

From knowledge-based programs to graded belief-based programs, part II: off-line reasoning

Noël Laverny

IRIT

Université Paul Sabatier

31062 Toulouse Cedex, France

laverny@irit.fr

Jérôme Lang

IRIT

Université Paul Sabatier

31062 Toulouse Cedex, France

lang@irit.fr

Abstract

Belief-based programs generalize knowledge-based programs [Fagin *et al.*, 1995] by allowing for incorrect beliefs, unreliable observations, and branching conditions that refer to implicit graded beliefs, such as in “while my belief about the direction to the railway station is not strong enough do ask someone”. We show how to reason off-line about the possible executions of a belief-based program, which calls for introducing second-order uncertainty in the model.

1 Introduction

Knowledge-based programs, or KBPs [Fagin *et al.*, 1995] are a powerful notion for expressing action policies in which branching conditions refer to knowledge (an agent acts according to what he knows), such as, typically,

if $\mathbf{K}\varphi$ then π else π'

where $\mathbf{K}$ is an epistemic (usually S5) modality, and π, π' are subprograms. However, branching conditions in KBPs cannot refer to possibly erroneous beliefs or to graded belief, such as in “repeat ask to someone about the way to the railway station until my belief about the direction to take is strong enough”. Recently, [Laverny and Lang, 2004] made a first step towards reasoning with *belief-based programs* (BBPs), where knowledge modalities of KBPs are replaced by graded belief modalities, whose semantics relies on belief states defined as ranking functions, which can be revised by observations and progressed by physical actions, which enables the agent to maintain his current belief state about the world while executing a BBP. Note that BBPs extend a restricted class of KBPs, in which (a) there is a single agent, (b) the agent has perfect recall and (c) branching conditions concern the agent’s knowledge about the present state of the world.

However, [Laverny and Lang, 2004] cannot deal with *off-line reasoning about the effects of a belief-based program*. Assume for instance that agent A is looking for the way to the railway station in a foreign town; her initial belief state is void, and she follows a plan consisting in sequentially asking to several pedestrians about the direction to follow, until she has acquired a sufficient level of certainty. Assume moreover

that answers, although not fully reliable, are normally correct. Each time A gets a direction confirming (resp. contradicting) her current belief, this belief becomes stronger (resp. weaker). Now, the assumption that answers are normally correct implies, for instance, that if A has already got 3 “right” and no “left” then the next answer is more likely to be “right” again than “left”: the plausibility of getting an observation depends on the current state.

Coping with this issue results in a natural way in *second-order uncertainty* when projecting a belief state by a BBP: in our example, the agent is able to predict beforehand something like “after asking 5 pedestrians, normally I’ll have a very firm belief about the direction to the station, although I cannot totally exclude the possibility that I’ll have only a weak, or even a totally void belief”. Such a complex belief state is a belief state about (future) belief states, that is, a second-order belief state. Thus, the main concern of this paper is how to do *a priori* reasoning about the possible states of belief produced by executing the program: uncertainty about which of these states of belief will result is itself represented as a ranking over states of belief.

After recalling some background in Section 2, belief-based programs are introduced in Section 3. Section 4 deals with *complex belief states* and their progression by actions and programs. In Section 5 we show how to compute progression syntactically. Related work is discussed in Section 6.

2 Background

In this Section we briefly recall some notions from [Laverny and Lang, 2004]. Let PS be a *finite* set of propositional symbols. L_{PS} is the (non-modal) propositional language generated from PS , the usual connectives and the Boolean constants $\top$ and $\perp$. $S = 2^{PS}$ is the set of *states* associated with PS . Formulas of L_{PS} are said *objective*. If $\varphi \in L_{PS}$ then $Mod(\varphi) = \{s \in S \mid s \models \varphi\}$. For $A \subseteq S$, $Form(A)$ is the objective formula (unique up to logical equivalence) such that $Mod(Form(A)) = A$.

Belief states are identified with *ordinal conditional functions* (OCF) [Spohn, 1988]: a belief state is a function $\kappa : S \rightarrow \overline{\mathbb{N}}$, where $\overline{\mathbb{N}} = \mathbb{N} \cup \{+\infty\}$, such that $\min_{s \in S} \kappa(s) = 0$. κ is lifted from S to L_{PS} by $\kappa(\varphi) = \min \{\kappa(s) \mid s \models \varphi\}$, with the convention $\min(\emptyset) = +\infty$ (we will use this convention throughout the paper without recalling it). $\kappa(s)$ can be seen as the *exceptionality degree* of s . In particular, $\kappa(s) = 0$

means that s is a normal state and $\kappa(s) = +\infty$ that s is totally impossible. For any $\varphi \in L_{PS}$, the belief state κ_φ is defined by $\kappa_\varphi(s) = \begin{cases} 0 & \text{if } s \models \varphi \\ +\infty & \text{if } s \not\models \varphi \end{cases}$. In particular, $\kappa_\top$ is the *void belief state*: $\forall s, \kappa_\top(s) = 0$.

Beliefs are expressed syntactically in a graded extension $KD45_G$ of $KD45$, whose language $\mathcal{L}_{PS}$ is defined as follows:

- (a) if $\varphi \in L_{PS}$ then $\mathbf{B}_1\varphi, \mathbf{B}_2\varphi, \dots, \mathbf{B}_\infty\varphi$ are in $\mathcal{L}_{PS}$;
- (b) if Φ and Ψ in $\mathcal{L}_{PS}$ then $\neg\Phi, \Phi \vee \Psi, \Phi \wedge \Psi$ in $\mathcal{L}_{PS}$.

Note that $\mathcal{L}_{PS}$ considers only *subjective* and *flat* formulas¹. Formulas of $\mathcal{L}_{PS}$ are denoted by capital Greek letters Φ, Ψ etc. while objective formulas are denoted by small Greek letters φ, ψ etc. $\mathbf{B}_i\varphi$ intuitively means that the agent believes φ with strength i . The larger i , the stronger the belief expressed by $\mathbf{B}_i$, and $\mathbf{B}_\infty$ is a *knowledge* (true belief) modality.

The truth of a formula of $\mathcal{L}_{PS}$ in a belief state κ is defined by:

- (a) if φ objective and $i \in \bar{\mathbb{N}}, \kappa \models \mathbf{B}_i\varphi$ iff $\kappa(\neg\varphi) \geq i$;
- (b) $\kappa \models \Phi \vee \Psi$ iff $\kappa \models \Phi$ or $\kappa \models \Psi$;
- (b) $\kappa \models \Phi \wedge \Psi$ iff $\kappa \models \Phi$ and $\kappa \models \Psi$;
- (c) $\kappa \models \neg\Phi$ iff $\kappa \not\models \Phi$.

Thus, $\kappa \models \mathbf{B}_i\varphi$ holds as soon as any countermodel of φ is exceptional at least to the degree i , or, equivalently, that all states such that $\kappa(s) < i$ are models of φ . In particular, $\mathbf{B}_1\varphi$ is satisfied when all normal states satisfy φ , and $\mathbf{B}_\infty\varphi$ is satisfied when all possible states (to any degree) are models of φ .

An *observation* is a belief state κ_{obs} , representing *all* we observe when getting the observation. Observations can be incomplete and partially unreliable (see [Laverny and Lang, 2004] for examples). The agent revises her current belief state by an observation by combining both: the revision of κ by κ_{obs} is undefined when $\min_S(\kappa + \kappa_{obs}) = \infty$, and otherwise is the belief state defined by

$$\forall s \in S, (\kappa \oplus \kappa_{obs})(s) = \kappa(s) + \kappa_{obs}(s) - \min_S(\kappa + \kappa_{obs})$$

In particular, $\kappa_\top \oplus \kappa_{obs} = \kappa_{obs}$ and $(\kappa \oplus \kappa_\varphi) = \kappa(\cdot|\varphi)$, where $\kappa(\cdot|\varphi)$ is Spohn's conditioning [Spohn, 1988].

A physical action α is a feedback-free action (that is, it possibly changes the state of the world but does not give any feedback), defined by a transition model consisting of a collection of belief states $\{\kappa_\alpha(\cdot|s), s \in S\}$. $\kappa_\alpha(s'|s)$ is the exceptionality degree of the outcome s' when performing α in state s . The *progression* of a belief state κ_0 by α is the belief state $\kappa \diamond \alpha = \kappa(\cdot|\kappa_0, \alpha)$ defined (cf. [Boutilier, 1998]) by

$$\forall s \in S, (\kappa \diamond \alpha)(s) = \kappa(s|\kappa_0, \alpha) = \min_{s' \in S} \{\kappa(s') + \kappa_\alpha(s|s')\}$$

A *positive formula* of $\mathcal{L}_{PS}$ is a formula where no $\mathbf{B}_i$ appears in the scope of negation. A *positive conjunctive* (PC) formula of $\mathcal{L}_{PS}$ is a formula of the form $\mathbf{B}_\infty\varphi_\infty \wedge$

$\mathbf{B}_n\varphi_n \wedge \dots \wedge \mathbf{B}_1\varphi_1$; without loss of generality we can assume $\varphi_i \models \varphi_{i+1}$, since $\mathbf{B}_\infty\varphi_\infty \wedge \mathbf{B}_n\varphi_n \wedge \dots \wedge \mathbf{B}_1\varphi_1$ is equivalent to $\mathbf{B}_\infty\varphi_\infty \wedge \mathbf{B}_n(\varphi_\infty \wedge \varphi_n) \wedge \dots \wedge \mathbf{B}_1(\varphi_\infty \wedge \varphi_n \wedge \dots \wedge \varphi_1)$. There is a one-to-one correspondance between belief states and satisfiable PC formulas (modulo logical equivalence): for each κ , the PC formula $G(\kappa) = \Phi_\kappa$ is defined as $\mathbf{B}_\infty\varphi_\infty \wedge \mathbf{B}_n\varphi_n \wedge \dots \wedge \mathbf{B}_1\varphi_1$, where $n = \max\{k < \infty, \exists s \text{ such that } \kappa(s) = k\}$, and for every $i \in \bar{\mathbb{N}}^*$, $\varphi_i = \text{Form}(\{s, \kappa(s) < i\})$. (Note that n is finite, because S is finite and $\min \kappa = 0$.) For instance, let $\kappa([a, b]) = 0$, $\kappa([a, \neg b]) = 1$, $\kappa([\neg a, b]) = 3$ and $\kappa([\neg a, \neg b]) = \infty$, then $\Phi_\kappa = \mathbf{B}_\infty(a \vee b) \wedge \mathbf{B}_3a \wedge \mathbf{B}_2a \wedge \mathbf{B}_1(a \wedge b)$ – which is equivalent to $\mathbf{B}_\infty(a \vee b) \wedge \mathbf{B}_3a \wedge \mathbf{B}_1b$. Conversely, for each satisfiable PC formula Ψ there is a belief state $\kappa_\Psi = H(\Psi)$ such that $G(H(\Psi)) \equiv \Psi$. $G(\kappa)$ represents *all the agent believes* in κ ². We will sometimes make the following slight abuse of notation: when a PC formula Ψ is equivalent to a shorter (but not PC) formula Ψ' , we write $\kappa_{\Psi'}$ instead of κ_Ψ . For instance, the PC formula $\Psi = \mathbf{B}_\infty\varphi_\infty \top \wedge \mathbf{B}_3\top \wedge \mathbf{B}_2r \wedge \mathbf{B}_1r$ is equivalent to $\mathbf{B}_2r$, therefore we write $\kappa_{\mathbf{B}_2r}$ instead of κ_Ψ .

3 Belief-based programs

Belief-based programs (BBP) are built up from a set of actions ACT and program constructors:

- the empty plan λ is a BBP;
- for any $\alpha \in ACT$, α is a BBP;
- if π and π' are BBPs then $(\pi; \pi')$ is a BBP;
- if π and π' are BBP and $\Phi \in \mathcal{L}_{PS}$, then $(\text{if } \Phi \text{ then } \pi \text{ else } \pi')$ and $(\text{while } \Phi \text{ do } \pi)$ are BBPs.

Thus, a BBP is a program *whose branching conditions are doxastically interpretable*: the agent can decide whether she believes to a given degree that a formula is true (whereas she is generally unable to decide whether a given objective formula is true in the actual world). For instance, the agent performing the BBP

$\pi = \text{while } \neg(\mathbf{B}_2r \vee \neg\mathbf{B}_2\neg r) \text{ do ask;}$
 $\text{if } \mathbf{B}_2r \text{ then } \text{goright} \text{ else } \text{goleft}$

performs the sensing action *ask* until she has a belief firm enough (namely of degree 2) about the way to follow (whether π is guaranteed to stop is a good question!).

Progression and revision in [Laverny and Lang, 2004] are used for maintaining the agent's current belief state *while the program is being executed*. However, predicting the future possible states resulting from the execution of a BBP *before* it has started to be executed (*off-line* evaluation) cannot be done in a satisfactory way. Consider $\pi = \text{ask; ask; ask; ask}$, consisting in asking in sequence to 4 pedestrians about the way to the station. Assume that each occurrence of *ask* can send

¹This restriction is made for the sake of simplicity; it would be possible to consider nested modalities, and then prove, as it is the case in $KD45$, that each formula is equivalent to a flat formula, but this issue has no relevance to the issues dealt with in this paper. Likewise, combinations of objective and subjective formulas do not play any role either for expressing belief-based programs.

²This could be formalized by extending our language with graded doxastic versions $\mathbf{O}_1, \dots, \mathbf{O}_\infty$ of the *only knowing* modality (e.g. [Levesque and Lakemeyer, 2000]), $\mathbf{O}_i\varphi$ meaning that *all the agent believes to the degree i is φ* . Due to space limitations we must omit the details.

back $obs_1 = \kappa_{B_1 r}$ or $obs_2 = \kappa_{B_1 \neg r}$, corresponding respectively to a pedestrian telling that the station is on the right (resp. left), taken with some limited reliability (for the sake of simplicity we exclude “don’t know” answers). Then all we can predict is that after doing π the agent will be in one of the 5 belief states $\kappa_{B_1 r}$, $\kappa_{B_1 \neg r}$, $\kappa_{B_2 r}$, $\kappa_{B_2 \neg r}$, $\kappa_{\top}$ ³. The point now is that obs_1 and obs_2 cannot always be considered as likely as each other: for instance, we may wish to express that accurate observations are more frequent than inaccurate ones. Therefore, observations should be ranked by their plausibility of occurrence given the current state and the sensing action performed. Then, the projection of an initial belief state by a program results in a *second-order* (or *complex*) belief state: in our example, one would expect to obtain that after asking to two persons, then normally the agent is the belief state $\kappa_{B_2 r}$ or in the belief state $\kappa_{B_2 \neg r}$, and exceptionally in the void belief state $\kappa_{\top}$. This issue is developed in next Section.

4 Complex belief states and progression

4.1 Complex belief states

Definition 1 Let B_S be the set of all belief states on S . A complex belief state (CBS) is an ordinal conditional function μ on B_S , i.e., a function $\mu : B_S \rightarrow \bar{\mathbb{N}}$ such that $\min_{\kappa \in B_S} \mu(\kappa) = 0$

μ is a second-order belief state expressing the beliefs, before executing some given program π , about the (future) possible belief states resulting from its execution.

Example 1 Let $S = \{r, \neg r\}$.

$$\mu : \left[\begin{array}{l} \kappa_0 : \left[\begin{array}{l} r : 0 \\ \neg r : 0 \end{array} \right] : \mu(\kappa_0) = 1 \\ \kappa_3 : \left[\begin{array}{l} r : 0 \\ \neg r : 2 \end{array} \right] : \mu(\kappa_3) = 0 \\ \kappa_4 : \left[\begin{array}{l} r : 2 \\ \neg r : 0 \end{array} \right] : \mu(\kappa_4) = 0 \end{array} \right]$$

is a CBS (by convention, for any belief state κ not mentioned we have $\mu(\kappa) = +\infty$); it intuitively represents a situation where the agent expects the resulting belief state to be either κ_0 , κ_3 or κ_4 , these last two being normal results and κ_0 being exceptional. Note that $\kappa_0 = \kappa_{\top}$, $\kappa_3 = \kappa_{B_2 r}$ and $\kappa_4 = \kappa_{B_2 \neg r}$.

We define μ_{κ} as the (degenerated) CBS defined by $\mu_{\kappa}(\kappa) = 0$ and $\mu_{\kappa}(\kappa') = \infty$ for all $\kappa' \neq \kappa$.

Note that since, unlike S , B_S is not finite, some CBSs cannot be finitely represented. We say that μ has a *finite support* iff only a finite number of belief states have a finite plausibility, i.e., $\{\kappa \in B_S \mid \mu(\kappa) < \infty\}$ is finite; in this case we define n_{μ} (or simply n where there is no ambiguity) as $\max\{i < \infty \mid \exists \kappa \text{ such that } \mu(\kappa) = i\}$.

4.2 Progression by sensing actions

Consider a finite set ACT_S of *sensing actions*, that send feedback to the agent, under the form of *observations*. Each sensing action is defined by a state-dependent plausibility assignment on possible observations.

³The notation κ_{Φ} has been introduced at the end of Section 2.

Definition 2 Let $OBS \subseteq B_S$ be a finite set of possible observations (recall that observations are belief states). An observation model is a collection of functions

$$\kappa_{OBS}(\cdot | s, \alpha) : OBS \rightarrow \bar{\mathbb{N}}$$

for every $\alpha \in ACT_S$ and every $s \in S$, such that:

1. for every $\alpha \in ACT_S$ and every $s \in S$,
 $\min_{obs \in OBS} \kappa_{OBS}(obs | s, \alpha) = 0$
2. for every $obs \in OBS$, if $obs(s) = \infty$ then for every $\alpha \in ACT_S$, $\kappa_{OBS}(obs | s, \alpha) = \infty$.

$\kappa_{OBS}(obs | s, \alpha)$ is the exceptionality degree of getting observation obs as feedback when executing the sensing action α in state s . This definition first appears in [Boutilier et al., 1998], and is similar in spirit to correlation probabilities between states and observations in partially observable Markov decision processes. Condition 1 expresses that there is always at least one normal observation; Condition 2 is a weak consistency condition between states and observations (an observation totally excluding a state cannot occur in that state).

Example 2

Let $S = \{r, \neg r\}$, $obs_1 = \kappa_{B_1 r}$, $obs_2 = \kappa_{B_1 \neg r}$, and let

$$\begin{array}{ll} \kappa_{OBS}(obs_1 | r, ask) = 0 & \kappa_{OBS}(obs_2 | r, ask) = 1 \\ \kappa_{OBS}(obs_2 | \neg r, ask) = 0 & \kappa_{OBS}(obs_1 | \neg r, ask) = 1 \end{array}$$

(all other observations being impossible, i.e., for $obs \neq obs_1, obs_2$, $\kappa(obs | s, \alpha) = \infty$; by convention we omit these when specifying $\kappa(\cdot | s, \alpha)$). This means that accurate observations are the normal ones, whereas incorrect observations are 1-exceptional.

Definition 3 Let κ_0 be a belief state and $\alpha \in ACT_S$. Given $obs \in OBS$, the plausibility of obtaining obs after α in belief state κ_0 is defined by

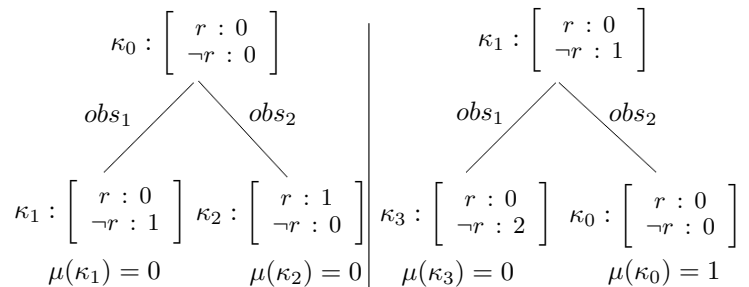
$$\kappa_{OBS}(obs | \kappa_0, \alpha) = \min_{s \in S} [\kappa_0(s) + \kappa_{OBS}(obs | s, \alpha)]$$

The progression of κ_0 by α is the complex belief state $prog(\kappa_0, \alpha) = \mu(\cdot | \kappa_0, \alpha)$ defined by: for all $\kappa \in B_S$,

$$\begin{aligned} & \mu(\kappa | \kappa_0, \alpha) \\ &= \min\{\kappa_{OBS}(obs | \kappa_0, \alpha) \mid obs \in OBS \text{ and } \kappa = \kappa_0 \oplus obs\} \end{aligned}$$

Thus, κ is all the more normal in the projected CBS $\mu(\cdot | \kappa_0, \alpha)$ as there exists a normal state s and a normal observation obs (given s) such that κ is the revision of κ_0 by obs . Condition 2 in Definition 2 guarantees that $\kappa_0 \oplus obs$ is defined whenever $\kappa_{OBS}(obs | \kappa_0, \alpha) < \infty$, which ensures that $\mu(\cdot | \kappa_0, \alpha)$ is a CBS.

Example 3 The figure on the left (resp. right) shows the progression of κ_0 (resp. κ_1) by ask .



$$\mu(\cdot|\kappa_0, \text{ask}) = \begin{bmatrix} \kappa_1 & : & 0 \\ \kappa_2 & : & 0 \end{bmatrix} \quad \mu(\cdot|\kappa_1, \text{ask}) = \begin{bmatrix} \kappa_3 & : & 0 \\ \kappa_0 & : & 1 \end{bmatrix}$$

4.3 Progression by physical actions

In addition to sensing actions we consider a finite set ACT_P of *physical, feedback-free actions*⁴. In Section 2, the progression of a belief state by a physical action was defined as a *belief state*. For the sake of homogeneity, we have now to define it as a CBS: the progression $\mu(\cdot|\kappa_0, \alpha)$ of a belief state κ_0 by $\alpha \in ACT_P$ is defined by $\mu(\cdot|\kappa_0, \alpha) = \mu_{\kappa_0 \diamond \alpha}$, i.e.,

$$\mu(\kappa|\kappa_0, \alpha) = \begin{cases} 0 & \text{if } \kappa = \kappa_0 \diamond \alpha \\ \infty & \text{otherwise} \end{cases}$$

4.4 Progression by belief-based programs

The progression of a belief state κ by a BBP π is the *complex belief state* $\mu(\cdot|\kappa, \pi)$ defined inductively by

- if $\pi = \lambda$ then $\mu(\cdot|\kappa, \pi) = \mu_\kappa$;
- if $\pi = \alpha$ then $\mu(\cdot|\kappa, \pi)$ is defined in Section 4.3 if $\alpha \in ACT_P$ and in Section 4.2 if $\alpha \in ACT_S$;
- if $\pi = (\pi_1; \pi_2)$ then $\mu(\kappa'|\kappa, \pi) = \min_{\kappa'' \in BS} (\mu(\kappa''|\kappa, \pi_1) + \mu(\kappa'|\kappa'', \pi_2))$
- if $\pi = \text{if } \Phi \text{ then } \pi_1 \text{ else } \pi_2$ then $\mu(\kappa'|\kappa, \pi) = \begin{cases} \mu(\kappa'|\kappa, \pi_1) & \text{if } \kappa \models \Phi \\ \mu(\kappa'|\kappa, \pi_2) & \text{otherwise} \end{cases}$
- if $\pi = \text{while } \Phi \text{ do } \pi'$ then $\mu(\kappa'|\kappa, \pi) = \begin{cases} \mu(\kappa'|\kappa, (\pi'; \pi)) & \text{if } \kappa \models \Phi \\ \mu_\kappa(\kappa') & \text{otherwise} \end{cases}$

If π contains no **while** construct then this definition is well-founded (it can be proved that it results a SBS which moreover does not depend on the order in which the above rules are applied). This is not so with **while** constructs, since the (recursive) definition leads to a fixpoint equation whose solution, when defined, is taken to be its least fixpoint when the latter is a CBS, which is not necessarily the case: consider $\kappa_0 = \kappa_\top$ and $\pi = \text{while } \top \text{ do ask}$; applying the above definition leads to $\mu(\kappa|\kappa_0, \pi) = \infty$ for all κ , which is not a CBS. Moreover, when $\mu(\cdot|\kappa_0, \pi)$ is defined, it does not necessarily have a finite support.

Example 4

- $\mu(\cdot|\kappa_\top, (\text{ask}; \text{ask})) = [\kappa_3 : 0; \kappa_4 : 0; \kappa_0 : 1]$;
- $\mu(\cdot|\kappa_\top, (\text{ask}; \text{ask}; \text{ask}; \text{ask})) = [\kappa_{\mathbf{B}_4 r} : 0; \kappa_{\mathbf{B}_4 \neg r} : 0; \kappa_3 : 1; \kappa_4 : 1; \kappa_0 : 2]$;
- $\pi_1 = (\text{ask}; \text{ask}; \text{if } \mathbf{B}_2 r \vee \mathbf{B}_2 \neg r \text{ then } \lambda \text{ else ask})$. Then $\mu(\cdot|\kappa_\top, \pi_1) = [\kappa_3 : 0; \kappa_4 : 0; \kappa_1 : 1; \kappa_2 : 1]$;
- $\pi_2 = \text{while } \neg(\mathbf{B}_2 r \vee \mathbf{B}_2 \neg r) \text{ do ask}$. Applying the definition gives a fixpoint equation whose solution is $\mu(\cdot|\kappa_0, \pi_2) = [\kappa_3 : 0; \kappa_4 : 0]$.⁵

⁴Such a partition between purely sensing and purely physical actions is usual and does not induce a loss of generality.

⁵The question whether this program can run forever is similar to that of whether it is possible that a fair coin tossed repeatedly always turns up heads, and this is thus related to an OCF version of the law of large numbers.

4.5 Two particular cases

An *unobservable environment* consists of a set of physical actions only ($ACT_S = \emptyset$).

A *fully observable environment* is somewhat harder to define formally because of the separation between physical and sensing actions, which prevents us to say that all actions send a full feedback. To cope with this, we assume that ACT_S contains one action $sense(x)$ for each propositional variable, which returns the truth value of x with full certainty, and we require that in a program, any physical action α should be followed by all actions $sense(x)$ – which means that after α and this sequence of sensing actions, the state is known with full certainty. Any program of this kind is said to be *admissible*. The initial state is also required to be known with certainty.

Proposition 1

1. in an unobservable environment, for any κ_0 and any program π for which $\mu(\cdot|\kappa_0, \pi)$ is defined, there exists a belief state κ such that $\mu(\cdot|\kappa_0, \pi) = \mu_\kappa$;
2. in a fully observable environment, for any belief state κ_0 and any admissible program π such that $\mu(\cdot|\kappa_0, \pi)$ is defined, $\mu(\kappa|\kappa_0, \pi) < \infty$ implies that κ is a precise belief state, i.e., $\kappa = \kappa_s$ for some $s \in S$.

5 Progression: syntactical computation

Applying progression as it is defined above has a prohibitive complexity, since it amounts at iterating over all belief states, states, and observations. In this Section we give a more friendly, syntactical way of computing progression, based on a compact representation of complex belief states.

So as to be able to reason about the resulting belief states, we now introduce a new family of modalities $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_\infty$ in addition to $\mathbf{B}_1, \dots, \mathbf{B}_\infty$. While the $\mathbf{B}_i$ modalities deal with uncertainty (*about the current state*), $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_\infty$ deal with second-order uncertainty, i.e., *uncertainty about the projected belief state* (with respect to some program).

$\mathcal{L}_{PS}^2$ is the language defined by:

- if Φ is a formula of $\mathcal{L}_{PS}$ then for all i , $\mathbf{P}_i \Phi$ is a formula of $\mathcal{L}_{PS}^2$;
- if Θ and Θ' are formulas of $\mathcal{L}_{PS}^2$ then $\Theta \wedge \Theta'$, $\Theta \vee \Theta'$ and $\neg \Theta$ are formulas of $\mathcal{L}_{PS}^2$.

Like for $\mathbf{B}_i$ modalities, we need not consider nested $\mathbf{P}_i$ modalities, neither heterogeneous combinations of $\mathbf{P}_i$ and $\mathbf{B}_i$ modalities (resp. objective formulas). Satisfaction of a $\mathcal{L}_{PS}^2$ formula by a CBS is defined by

- $\mu \models \mathbf{P}_i \Phi$ iff for all $\kappa \in BS$, $\kappa \models \neg \Phi$ implies $\mu(\kappa) \geq i$;
- $\mu \models \Theta \wedge \Theta'$ iff $\mu \models \Theta$ and $\mu \models \Theta'$ (and similarly for the other connectives).

Validity, satisfiability and logical consequence are defined in the usual way. Intuitively, $\mathbf{P}_i \Phi$ means that *all belief states that do not satisfy Φ are exceptional at least to the degree i* .

When reasoning about CBS, we are mainly concerned with inferring *positive formulas* – inferring negative formulas such as $\neg \mathbf{P}_i \Phi$ is somewhat derivative. We define a *positive $\mathcal{L}_{PS}^2$ formula* as follows:

- if Φ is a positive $\mathcal{L}_{PS}$ formula and $i \in \bar{\mathbb{N}}$ then $\mathbf{P}_i\Phi$ is a positive $\mathcal{L}_{PS}^2$ formula;
- if Θ_1 and Θ_2 are positive $\mathcal{L}_{PS}^2$ formulas then $\Theta_1 \wedge \Theta_2$ and $\Theta_1 \vee \Theta_2$ is a positive $\mathcal{L}_{PS}^2$ formula.

Moreover, a canonical $\mathcal{L}_{PS}^2$ formula is a $\mathcal{L}_{PS}^2$ formula of the form $\Theta = \mathbf{P}_1\Phi_1 \wedge \mathbf{P}_2\Phi_2 \wedge \dots \wedge \mathbf{P}_n\Phi_n \wedge \mathbf{P}_\infty\Phi_\infty$, where $\Phi_1, \dots, \Phi_n, \Phi_\infty$ are positive $\mathcal{L}_{PS}$ formulas⁶.

Given a CBS μ with finite support, the canonical $\mathcal{L}_{PS}^2$ formula $G^+(\mu) = \Theta_\mu$ associated with μ is defined by

$$\Theta_\mu = \bigwedge_{i=1}^{i=n} \mathbf{P}_i \left(\bigvee_{\mu(\kappa) < i} G(\kappa) \right) \wedge \mathbf{P}_\infty \left(\bigvee_{\mu(\kappa) < \infty} G(\kappa) \right)$$

where $n = n_\mu$ and $G(\kappa)$ is the canonical $\mathcal{L}_{PS}$ formula corresponding to κ (cf. Section 2).

Proposition 2 For any CBS μ with finite support and any positive conjunctive $\mathcal{L}_{PS}^2$ formula Θ , $\mu \models \Theta$ if and only if $\Theta_\mu \models \Theta$, that is, Θ_μ is the strongest positive conjunctive $\mathcal{L}_{PS}^2$ formula satisfied by μ .

Example 1 (continued)

$$\begin{aligned} \Theta_\mu &= \mathbf{P}_1(\mathbf{B}_2r \vee \mathbf{B}_2\neg r) \wedge \mathbf{P}_\infty(\mathbf{B}_2r \vee \mathbf{B}_2\neg r \vee \mathbf{B}_\infty\top) \\ &\equiv \mathbf{P}_1(\mathbf{B}_2r \vee \mathbf{B}_2\neg r) \end{aligned}$$

We are now in position to give a syntactical characterization of progression.

Definition 4 Let Φ be a positive conjunctive $\mathcal{L}_{PS}$ formula and α be any action. The progression of Φ by α is the canonical $\mathcal{L}_{PS}^2$ formula $Prog(\Phi, \alpha)$ corresponding to the CBS $\mu(\cdot | \kappa_\Phi, \alpha)$, i.e.,

$$Prog(\Phi, \alpha) = G^+(\mu(\cdot | \kappa_\Phi, \alpha)) = \Theta_{\mu(\cdot | \kappa_\Phi, \alpha)}$$

We now show how the formula $Prog(\Phi, \alpha)$ can be computed without first generating the corresponding μ .

Proposition 3

Let α be a sensing action and $\Phi = \mathbf{B}_1\varphi_1 \wedge \dots \wedge \mathbf{B}_p\varphi_p \wedge \mathbf{B}_\infty\varphi$ a positive conjunctive $\mathcal{L}_{PS}$ formula. We define:

- for any $obs \in OBS$ and $i \in \bar{\mathbb{N}}$,
 $\psi_{i,obs,\alpha} = Form\{s \in S \mid \mu_{OBS}(obs|s, \alpha) < i\}$;
- $X_{i,\Phi,\alpha} = \{obs \in OBS \mid (\varphi_1 \wedge \psi_{i,obs,\alpha}) \vee \dots \vee (\varphi_i \wedge \psi_{i,obs,\alpha}) \not\equiv \perp\}$;
- $X_{\infty,\Phi,\alpha} = \{obs \in OBS \mid \varphi \wedge \psi_{\infty,obs,\alpha} \not\equiv \perp\}$;
- n is the largest integer i such that $X_{i,\Phi,\alpha} \subsetneq X_{\infty,\Phi,\alpha}$.

Then

$$\begin{aligned} Prog(\Phi, \alpha) &= \bigwedge_{i=1}^{i=n} \mathbf{P}_i \left(\bigvee_{obs \in X_{i,\Phi,\alpha}} \Phi \otimes \Phi_{obs} \right) \\ &\quad \wedge \mathbf{P}_\infty \left(\bigvee_{obs \in X_{\infty,\Phi,\alpha}} \Phi \otimes \Phi_{obs} \right) \end{aligned}$$

where $\otimes$ is the syntactical revision operator (Proposition 1 of [Laverny and Lang, 2004]), which satisfies $\Phi \otimes \Phi_{obs} = G(H(\Phi) \oplus obs)$.

⁶Notice that canonical $\mathcal{L}_{PS}^2$ formulas are positive $\mathcal{L}_{PS}^2$ formulas in which there is no disjunction at the level of the $\mathbf{P}_i$ modalities (but disjunctions may appear in the scope of a $\mathbf{P}_i$ modality, and of course in the scope of a $\mathbf{B}_i$ modality too).

Example 2 (continued) Let $\Phi = \mathbf{B}_1r$. We get $X_{1,\Phi,ask} = \{obs_1\}$, $X_{2,\Phi,ask} = \{obs_1, obs_2\}$ and for all $n \geq 2$, $X_{n,\Phi,ask} = \{obs_1, obs_2\}$. Proposition 3 gives $Prog(\Phi, ask) = \mathbf{P}_1(\mathbf{B}_1r \otimes \mathbf{B}_1r) \wedge \mathbf{P}_\infty((\mathbf{B}_1r \otimes \mathbf{B}_1r) \vee (\mathbf{B}_1r \otimes \mathbf{B}_1\neg r)) \equiv \mathbf{P}_1\mathbf{B}_2r \wedge \mathbf{P}_\infty(\mathbf{B}_2r \vee \mathbf{B}_\infty\top) \equiv \mathbf{P}_1\mathbf{B}_2r$.

The characterization for physical actions is much easier.

Proposition 4 For any physical action α and any PC $\mathcal{L}_{PS}$ formula Φ , $Prog(\Phi, \alpha) \equiv \mathbf{P}_\infty G(\kappa_\Phi \diamond \alpha)$.

Moreover, $G(\kappa_\Phi \diamond \alpha)$ can be computed efficiently using Proposition 2 of [Laverny and Lang, 2004].

Lastly, the progression of a positive conjunctive $\mathcal{L}_{PS}$ formula Φ by a BBP π is the canonical $\mathcal{L}_{PS}^2$ formula $Prog(\Phi, \pi)$ defined inductively as follows:

- $Prog(\Phi, \lambda) = \mathbf{P}_\infty\Phi$;
- $Prog(\Phi, \alpha)$ is defined at Definition 4 if α is an action;
- $Prog(\Phi, \text{if } \Psi \text{ then } \pi_1 \text{ else } \pi_2)$
 $= \begin{cases} Prog(\Phi, \pi_1) & \text{if } \Phi \models \Psi \\ Prog(\Phi, \pi_2) & \text{otherwise} \end{cases}$
- $Prog(\Phi, \pi_1; \pi_2) =$

$$\begin{aligned} &\bigwedge_{i=1}^{i=n} \mathbf{P}_i \left(\bigvee_{u+v=i+1} [Prog([Prog(\Phi, \pi_1)]_u, \pi_2)]_v \right) \\ &\quad \wedge \mathbf{P}_\infty([Prog([Prog(\Phi, \pi_1)]_\infty, \pi_2)]_\infty) \end{aligned}$$

- $Prog(\Phi, \text{while } \Psi \text{ then } \pi') =$
 $\begin{cases} Prog(\Phi, \pi'; \pi) & \text{if } \Phi \models \Psi \\ Prog(\Phi, \lambda) & \text{otherwise} \end{cases}$

where

1. for any canonical $\mathcal{L}_{PS}^2$ formula Θ , $[\Theta]_i$ is the strongest positive $\mathcal{L}_{PS}$ formula Ψ (unique up to logical equivalence) such that $\Theta \models \mathbf{P}_i\Psi$.
2. for any positive conjunctive $\mathcal{L}_{PS}$ formulas Φ_1 and Φ_2 : $[Prog(\Phi_1 \vee \Phi_2, \pi)]_i = [Prog(\Phi_1, \pi)]_i \vee [Prog(\Phi_2, \pi)]_i$.

Proposition 5 $Prog(\Phi_\kappa, \pi) = \Theta_{\mu(\cdot | \kappa, \pi)}$

Example 5

Let $\pi = (ask; ask; \text{if } \mathbf{B}_2r \vee \mathbf{B}_2\neg r \text{ then } \lambda \text{ else } ask)$ and $\pi' = (\text{while } \neg(\mathbf{B}_2r \vee \mathbf{B}_2\neg r) \text{ do } ask)$.

- $Prog(\mathbf{B}_\infty\top, ask) \equiv \mathbf{P}_\infty(\mathbf{B}_1r \vee \mathbf{B}_1\neg r)$
- $Prog(\mathbf{B}_\infty\top, ask; ask) \equiv \mathbf{P}_1(\mathbf{B}_2r \vee \mathbf{B}_2\neg r)$.
- $Prog(\mathbf{B}_\infty\top, \pi) \equiv \mathbf{P}_1(\mathbf{B}_2r \vee \mathbf{B}_2\neg r) \wedge \mathbf{P}_\infty(\mathbf{B}_1r \vee \mathbf{B}_1\neg r)$
- $Prog(\mathbf{B}_\infty\top, \pi') \equiv \mathbf{P}_\infty(\mathbf{B}_2r \vee \mathbf{B}_2\neg r)$

6 Related work

Partially observable Markov decision processes

POMDPs are the dominant approach for acting under partial observability (including nondeterministic actions and unreliable observations). The relative plausibility of observations given states, as well as the notion of progressing a belief state by an action, has its natural counterparts in POMDPs. Now,

there are two important differences between POMDPs and our work.

First, in POMDPs policies, branching is conditioned by the observation sequence that has lead to the current belief state; the policy is therefore directly implementable, without the need for an on-line reasoning phase. In our framework, branching conditions are expressed logically, which may allow for much more compact policies than branching on observations. In this view, BBPs can be seen as high-level, compact specifications of POMDP policies (the policy being the implementation of the program). *Our work can thus be seen as a first step towards bridging knowledge-based programs and POMDPs.*

Second, we allow for second-order uncertainty whereas POMDPs get rid of it: if $p_{proj}(p|p_0, \pi)$ is the probability of obtaining the probabilistic belief state p after executing π in p_0 , this second-order belief state is systematically reduced into the first-order one $\hat{p}$, following the lottery reduction principle: $\hat{p}(s) = \sum_{p \in PB_S} p_{proj}(p) \cdot p(s)$ (where PB_S is the set of probability distributions on S). This is a loss of expressivity, as seen on the following example: consider the action α of tossing a coin and the action β of sensing it. The agent knows that after performing α only, she will be for sure in a belief state where $p(heads) = p(tails) = 0.5$. This differs from projecting the effects of $\alpha; \beta$: then she knows that he will reach either a belief state where she *knows heads* or a belief state where she *knows tails*, with equal probability. After reduction, these two plans and their projections can no longer be distinguished – whereas our CBS do distinguish them⁷.

Cognitive robotics

Another fairly close area is that of cognitive robotics, especially the work around *Golog* and the situation calculus (e.g., [Reiter, 2001]), which is concerned with logical specifications of actions and programs, including probabilistic extensions and partial observability. [Bacchus *et al.*, 1999] gives an account for the dynamics of probabilistic belief states when perceiving noisy observations and performing physical actions with noisy effectors. [Grosskreutz and Lakemeyer, 2000] considers probabilistic Golog programs with partial observability, with the aim of turning off-line nondeterministic plans into programs that are guaranteed to reach the goal with some given probability. In both works, branching conditions involve objective formulas and there is no account for second-order uncertainty. Bridging belief-based programs and the situation calculus (and Golog) is a promising issue.

Belief revision with unreliable observations

[Boutilier *et al.*, 1998] might be the closest work to ours, as far as the dynamics of belief states in the presence of noisy observations is concerned. Our notion of an observation model (Definition 3) owes a lot to theirs (which also makes use of ranking functions). Now, their objectives depart from ours, as they focus on a general ontology for belief

revision and do not consider physical actions nor programs, nor do they give a syntactical way of computing their revision functions.

Counterfactual belief-based programs

[Halpern and Moses, 2004] consider belief-based programs with counterfactuals whose semantics, like ours, is based on ranking functions. They do not allow for graded belief in branching conditions, nor unreliable observations (ranking functions are used for evaluating counterfactuals), but they allow for counterfactual branching conditions, possibly referring to future belief states, such as “if I believe that if I were not sending this message now then my partner might not get this important information, then I should send it”. Adding counterfactuals and beliefs about future states to our framework is worth considering for further research.

Acknowledgements

Thanks to the anonymous referees for many relevant and helpful suggestions.

References

- [Bacchus *et al.*, 1999] F. Bacchus, J. Halpern, and H. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111:171–208, 1999.
- [Boutilier *et al.*, 1998] C. Boutilier, N. Friedman, and J. Halpern. Belief revision with unreliable observations. In *Proceedings of AAAI-98*, pages 127–134, 1998.
- [Boutilier, 1998] C. Boutilier. A unified model of qualitative belief change: A dynamical systems perspective. *Artificial Intelligence*, 98(1-2):281–316, 1998.
- [Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [Grosskreutz and Lakemeyer, 2000] H. Grosskreutz and G. Lakemeyer. Turning high-level plans into robot programs in uncertain domains. In *Proceedings of ECAI-2000*, pages 548–552, 2000.
- [Halpern and Moses, 2004] J. Halpern and Y. Moses. Using counterfactuals in knowledge-based programming. *Distributed Computing*, 17(2):91–106, 2004.
- [Laverny and Lang, 2004] N. Laverny and J. Lang. From knowledge-based programs to graded belief based programs, part i: on-line reasoning. In *Proceedings of ECAI-04*, pages 368–372, 2004.
- [Levesque and Lakemeyer, 2000] H. Levesque and G. Lakemeyer. *The logic of knowledge bases*. MIT Press, 2000.
- [Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [Spohn, 1988] W. Spohn. Ordinal conditional functions: a dynamic theory of epistemic states. In William L. Harper and Brian Skyrms, editors, *Causation in Decision, Belief Change and Statistics*, volume 2, pages 105–134. Kluwer Academic Pub., 1988.

⁷Incidentally, we can reduce a CBS in a similar way: the reduction of the CBS μ of the is the belief state $\hat{\kappa}_\mu \in B_S$ defined by $\hat{\kappa}_\mu(s) = \min_{\kappa \in B_S} \mu(\kappa) + \kappa(s)$. It is readily checked that $\hat{\kappa}$ is a belief state. Reducing μ into $\hat{\kappa}_\mu$ results in a much simpler (and shorter) structure, which, on the other hand, is much less informative than μ .

A Model-Theoretic Counterpart of Loop Formulas

Joohyung Lee

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712, USA
appsmurf@cs.utexas.edu

Abstract

In an important recent paper, Lin and Zhao introduced the concept of a loop formula, and showed that the answer sets for a logic program are exactly the models of Clark's completion of the program that satisfy the loop formulas. Just as supported sets are a model-theoretic account of completion, "externally supported" sets, defined in this paper, are a model-theoretic counterpart of loop formulas. This reformulation of loop formulas shows that they are related to assumption sets (Saccá and Zaniolo) and to unfounded sets (Van Gelder, Ross and Schlipf; Leone, Rullo and Scarcello), invented many years earlier. Other contributions of this paper includes a simplification of the definition of a loop, extending it to programs with classical negation and infinite programs, and a generalization of the definition of a loop formula.

1 Introduction

The completion semantics [Clark, 1978] and the answer set semantics (also known as the stable model semantics) [Gelfond and Lifschitz, 1988] are two well-known proposals for defining the meaning of negation as failure in logic programs.¹ The former is attractive because it is simply a translation from logic programs to classical logic, but it sometimes gives unintuitive results [Przymusiński, 1989, Section 4.1]. It is well known that an answer set for a logic program is also a model of its completion, while the converse, generally, does not hold.

By adding so-called "loop formulas" to completion, Lin and Zhao [2004] defined a set of formulas whose models are exactly the answer sets. The idea comes from observing that "loops" are related to the mismatch between the models of the completion and the answer sets. Intuitively, every atom in an answer set can be "derived" from a program in a finite number of steps. The atoms in a loop that are not "supported from outside" cannot be derived in a finite number of steps, but they do not contradict the completion of the program.

¹As the answer set semantics is usually defined for propositional (grounded) logic programs only, in this paper we limit our attention to such programs, and refer to the propositional case of Clark's completion.

The Lin/Zhao theorem allows us to compute answer sets using SAT solvers, which has led to the implementation of SAT-based answer set solvers ASSAT² [Lin and Zhao, 2004] and CMODELS³ [Giunchiglia *et al.*, 2004]. These systems turn a logic program into a propositional theory and then call SAT solvers, such as CHAFF, SATO and RELSAT, to find its models, which in turn correspond to the answer sets. Thanks to recent progress with SAT solvers, in several benchmark examples they found answer sets faster than other answer set solvers such as SMOLELS⁴ and DLV⁵.

A program that has no loops is called "tight," and for tight programs the completion semantics and the answer set semantics are equivalent to each other. This fact was discovered earlier by Fages [1994], and then was generalized and extended by Erdem and Lifschitz [2003] to programs with nested expressions (in the sense of [Lifschitz *et al.*, 1999]) in the bodies of rules.

Lee and Lifschitz [2003] showed that program completion, loop formulas and the Lin/Zhao theorem can be extended to disjunctive logic programs and, more generally, to arbitrary programs with nested expressions. As a consequence, the concept of a tight program and Fages' theorem are extended to disjunctive programs as well.

Yet the previous work on loop formulas has been limited to *finite* programs *without classical negation*. In this paper, we lift these limitations. First, to account for programs that allow classical negation, we propose a model-theoretic counterpart of loop formulas—the concept of an "externally supported" set. It is similar to the model-theoretic counterpart of Clark's completion—the concept of a supported set introduced in [Apt *et al.*, 1988]. Interestingly, this reformulation of loop formulas shows that they are related to assumption sets [Saccá and Zaniolo, 1990] and to unfounded sets [Van Gelder *et al.*, 1991; Leone *et al.*, 1997] invented many years earlier. Second, we extend the concept of a loop formula to programs with nested expressions. Finally, we extend the notion of a loop and the theorem on loop formulas to infinite programs.

In the next section we introduce the concept of an exter-

²<http://assat.cs.ust.hk/> .

³<http://www.cs.utexas.edu/users/tag/cmodels/> .

⁴<http://www.tcs.hut.fi/Software/smodels/> .

⁵<http://www.dbai.tuwien.ac.at/proj/dlv/> .

nally supported set, extend the notion of a loop to infinite programs, state our main theorem, and discuss its relation to Fages' theorem about tight programs. In Section 3 we use the main theorem to define two transformations turning a logic program into an equivalent propositional theory. In Section 4, we relate externally supported sets to assumption sets and to unfounded sets, and compare our reformulation of loop formulas with the original definition by Lin and Zhao. The main theorem is generalized to arbitrary programs with nested expressions in Section 5.

2 Externally Supported Sets

2.1 Nondisjunctive Case

We begin with a review of the answer set semantics for nondisjunctive programs given in [Gelfond and Lifschitz, 1991, Section 2].⁶ The words *atom* and *literal* are understood here as in propositional logic; we call the negation sign $\neg$ in negative literals *classical negation*, to distinguish it from the symbol for negation as failure (*not*).

A *nondisjunctive rule* is an expression of the form

$$l_1 \leftarrow l_2, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n \quad (1)$$

($1 \leq m \leq n$) where all l_i are literals. We will often write (1) in the form

$$l_1 \leftarrow B, F \quad (2)$$

where B is $l_2, \dots, l_m$, and F is $\text{not } l_{m+1}, \dots, \text{not } l_n$, and we will sometimes identify B with the set $\{l_2, \dots, l_m\}$.

A *nondisjunctive (logic) program* is a set of rules of the form (1).

We say that a set X of literals *satisfies* the body B, F of rule (2) (symbolically $X \models B, F$) if $l_2, \dots, l_m \in X$, and $l_{m+1}, \dots, l_n \notin X$. We say that X *satisfies* a nondisjunctive program Π (symbolically $X \models \Pi$) if, for every rule (2) of that program, $l_1 \in X$ whenever X satisfies B, F .

The *reduct* Π^X of a nondisjunctive program Π with respect to a set X of literals is obtained from Π by deleting each rule (2) such that $X \not\models F$, and replacing each remaining rule (2) by $l_1 \leftarrow B$. A consistent set X of literals is an *answer set* for Π if X is minimal among the sets of literals that satisfy Π^X .

We say that a set Y of literals is *supported* by Π w.r.t. a set X of literals if there is a rule (2) in Π such that $l_1 \in Y$, and $X \models B, F$. Informally, such a rule (2) characterizes a source of support for the literal l_1 in Y .⁷

Now we make the notion of a supported set slightly stronger. We will say that a set Y of literals is *externally supported* by Π w.r.t. X if there is a rule (2) in Π such that $l_1 \in Y$, $X \models B, F$, and $B \cap Y = \emptyset$. Informally, the new, third condition ensures that the support for the literal l_1 in Y comes from *outside* of Y .

⁶Here we do not allow inconsistent answer sets.

⁷One may notice that this definition is slightly different from the usual definition of a supported set, which says that a set X of literals is supported by Π if, for every literal $l_1 \in X$, there is a rule (2) in Π such that $X \models B, F$ [Apt et al., 1988]. It is clear that our definition is more general: X is supported by Π under the usual definition iff every singleton subset of X is supported by Π w.r.t. X under the new definition.

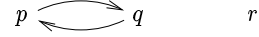


Figure 1: The dependency graph of Π_2

For example, consider the following program Π_1 :

$$p \leftarrow q \quad q \leftarrow p$$

Each of the sets $\{p\}$ and $\{q\}$ is externally supported (hence also supported) by Π_1 w.r.t. $\{p, q\}$, while $\{p, q\}$ is supported but not externally supported by Π_1 w.r.t. $\{p, q\}$. On the other hand, let Π_2 be the following program that adds two rules to Π_1 :

$$\begin{array}{ll} p \leftarrow q & q \leftarrow p \\ p \leftarrow \text{not } r & r \leftarrow \text{not } p. \end{array}$$

Then every nonempty subset of $\{p, q\}$ is externally supported by Π_2 w.r.t. $\{p, q\}$.

The *positive dependency graph* of Π is the directed graph such that

- its vertices are the literals occurring in Π , and
- its edges go from l_1 to $l_2, \dots, l_m$ for each rule (1) of Π .

When Π is finite, a nonempty set L of literals is called a *loop* of Π if, for every pair l_1, l_2 of literals in L , there exists a path from l_1 to l_2 in the positive dependency graph of Π such that all vertices in this path belong to L .⁸ In other words, a nonempty set L of literals is a loop of Π iff the subgraph of the positive dependency graph of Π induced by L is strongly connected. Note that, for every literal l that occurs in Π , the singleton set $\{l\}$ is a loop, according to this definition.⁹

The positive dependency graph of Π_2 , shown in Figure 1, has four loops: $\{p\}, \{q\}, \{r\}, \{p, q\}$.

Main Theorem for Finite Nondisjunctive Programs For any finite nondisjunctive program Π and any consistent set X of literals, the following conditions are equivalent:

- X is an answer set for Π .
- X satisfies Π , and every set of literals that has a common element with X is externally supported by Π w.r.t. X .
- X satisfies Π , and every loop of Π that is contained in X is externally supported by Π w.r.t. X .

Out of the three implications (a) to (b), (b) to (c), (c) to (a), the second is obvious, because every loop that is contained in X has a common element with X . Program Π_2 above has two answer sets: $\{p, q\}$ and $\{r\}$. Since (a) implies (b), it follows that every set of atoms that has a common element with $\{p, q\}$ is externally supported w.r.t. $\{p, q\}$, and every set of atoms that has a common element with $\{r\}$ is externally supported w.r.t. $\{r\}$. On the other hand, since (c) implies (a), to show that $\{p, q\}$ is an answer set we only need to check that every loop which is a subset of $\{p, q\}$ (that is, each of $\{p\}, \{q\}, \{p, q\}$) is externally supported w.r.t. $\{p, q\}$. Similarly, to show that $\{r\}$ is an answer set, we only need to check that $\{r\}$ is externally supported w.r.t. $\{r\}$.

⁸Note that we do allow paths of length 0.

⁹Thus our definition is slightly different from the definition by Lin and Zhao (See Section 4 for detail). The example of a singleton loop shows that a loop of Π does not necessarily correspond to a loop (or cycle) of the positive dependency graph of Π in the sense of graph theory.

The equivalence between (a) and (b) is a generalization of a theorem from [Saccá and Zaniolo, 1990] as we will discuss in Section 4.1, and the equivalence between (a) and (c) is a model-theoretic account of loop formulas as we will discuss in Section 3.

2.2 Extension to Programs in Canonical Form

We will extend the main theorem to finite programs with nested expressions [Lifschitz *et al.*, 1999]. In this section, for simplicity, instead of arbitrary rules with nested expressions, we consider rules of the form

$$l_1; \dots; l_k \leftarrow l_{k+1}, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n, \text{not not } l_{n+1}, \dots, \text{not not } l_p \quad (3)$$

($0 \leq k \leq m \leq n \leq p$) where all l_i are literals.¹⁰ We will call such rules *canonical*, and will often write (3) in the form

$$A \leftarrow B, F \quad (4)$$

where A is $l_1, \dots, l_k$, B is $l_{k+1}, \dots, l_m$, and F is

$$\text{not } l_{m+1}, \dots, \text{not } l_n, \text{not not } l_{n+1}, \dots, \text{not not } l_p.$$

We will sometimes identify A with the set $\{l_1, \dots, l_k\}$ and B with the set $\{l_{k+1}, \dots, l_m\}$.

A *canonical program* is a set of rules of the form (3).

The definition of satisfaction given in Section 2.1 is extended to canonical programs as follows. We say that a set X of literals *satisfies* the body B, F of rule (4) if $l_{k+1}, \dots, l_m \in X$, $l_{m+1}, \dots, l_n \notin X$, and $l_{n+1}, \dots, l_p \in X$. X *satisfies* a program if, for every rule (4) of that program, at least one of the literals in A belongs to X whenever X satisfies B, F .

The reduct Π^X of a canonical program Π with respect to a set X of literals is obtained from Π by deleting each rule (4) such that $X \not\models F$, and replacing each remaining rule (4) by $A \leftarrow B$. A consistent set X of literals is an *answer set* for Π if X is minimal among the sets of literals that satisfy Π^X .

Given a canonical program Π and a set X of literals, we will say that a set Y of literals is *externally supported* by Π w.r.t. X if there is a rule (4) in Π such that $A \cap Y \neq \emptyset$, $X \models B, F$, $B \cap Y = \emptyset$, and $X \cap (A \setminus Y) = \emptyset$. The last condition is suggested by the extension of the definition of a supported set to disjunctive programs proposed in [Baral and Gelfond, 1994] and [Inoue and Sakama, 1998].

The definition of a positive dependency graph is extended to the general case straightforwardly: for each rule (3) in Π , the edges of the graph go from each literal l_i ($0 < i \leq k$) to each literal l_j ($k < j \leq m$). The definition of a loop from Section 2.1 remains the same for arbitrary finite programs.

The theorem from Section 2.1 remains true if we replace “nondisjunctive” in its statement with “canonical.”

For example, let Π_3 be the program

$$p; s \leftarrow q \quad q \leftarrow p \quad p; r \leftarrow \text{not } s,$$

which has two answer sets: $\{p, q\}$ and $\{r\}$. The comments about these answer sets at the end of Section 2.2 apply to program Π_3 as well.

¹⁰Any program with nested expressions can be turned into an equivalent program whose rules have the form (3), or equivalently $l_1; \dots; l_k; \text{not } l_{n+1}; \dots; \text{not } l_p \leftarrow l_{k+1}, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n$ [Lifschitz *et al.*, 1999, Proposition 6 (iii)]. In Section 5, the main theorem is extended to arbitrary rules with nested expressions.

2.3 Extension to Infinite Programs

So far we restricted our attention to finite programs only. Indeed, unless we modify the definition of a loop, the theorem in Section 2.1 does not hold for infinite programs: conditions (a) and (b) are equivalent to each other even for infinite programs, but condition (c) turns out weaker than the others. For example, consider the following infinite program Π_4 :

$$p_i \leftarrow p_{i+1} \quad (i > 0),$$

where each p_i is an atom. The program has no loops (in the sense of Section 2.1) other than singletons. Besides $\emptyset$, which is the only answer set for Π_4 , there is one more set that satisfies condition (c): $\{p_1, p_2, \dots\}$.

Looking at the example, we observe that condition (c) is weaker than condition (b) in that it does not account for infinite paths in the positive dependency graph which do not correspond to loops, while condition (b) tells us that the sets that correspond to such infinite paths should also be externally supported. We propose to modify the definition of a loop as follows.

Given a (possibly infinite) program Π and a set L of literals, we say that L is *unbounded* if, for every literal $l \in L$, there exists an infinite path in L in the positive dependency graph of Π that starts from l and does not visit the same vertex more than once. A nonempty set L of literals is called a *loop* of Π if L is unbounded, or for every pair l_1, l_2 of literals in L , there exists a path from l_1 to l_2 in the positive dependency graph of Π such that all vertices in this path belong to L . When Π is finite, no set of literals that occur in Π is unbounded, so that this definition of a loop reduces to the earlier definition.

In program Π_4 above, every set $\{p_i, p_{i+1}, \dots\}$ ($i > 0$) is a loop according to this definition, since it is unbounded. But it is not externally supported by Π_4 w.r.t. $\{p_1, p_2, \dots\}$.

The theorem from Section 2.1 remains true if we replace “finite nondisjunctive” with “canonical”:

2.4 Tight Programs

The notion of a loop helps us simplify the definition of a “tight” program [Erdem and Lifschitz, 2003]. Let Π be any canonical program. We will say that a loop of Π is *trivial* if the loop consists of a single literal such that the positive dependency graph of Π does not contain an edge from the literal to itself. For example, in program Π_4 above, every singleton loop is trivial, while every loop $\{p_i, p_{i+1}, \dots\}$ ($i > 0$) is not. We say that Π is *absolutely tight* if every loop of Π is trivial. Let Π_X be the set of rules of Π whose heads and bodies are satisfied by X . Program Π is *tight on* a set X of literals if the subgraph of the positive dependency graph of Π_X induced by X has no loops other than trivial ones. These definitions are more general than the corresponding definitions in [Lee and Lifschitz, 2003] in that they are applicable to infinite programs and to programs with classical negation as well. We get the following corollaries to the main theorem, which generalizes Propositions 2 and 4 from [Lee and Lifschitz, 2003].

Corollary 1 *For any absolutely tight canonical program Π and any consistent set X of literals, X is an answer set for Π iff X satisfies Π and X is supported by Π .*

Corollary 2 For any canonical program Π and any consistent set X of literals such that Π is tight on X , X is an answer set for Π iff X satisfies Π and X is supported by Π .

3 Translating Logic Programs into Propositional Logic

In application to finite programs without classical negation, conditions (b) and (c) of the main theorem suggest ways to turn a program into an equivalent propositional theory. This translation is closely related to the Lin/Zhao theorem on loop formulas, as we will see in Section 4.

Consider a finite program Π whose rules have the form (4) where all l_i are atoms. For any set Y of atoms, the *external support formula* for Y is the disjunction of the conjunctions

$$B \wedge F \wedge \bigwedge_{p \in A \setminus Y} \neg p \quad (5)$$

for all rules (4) of Π such that $A \cap Y \neq \emptyset$, and $B \cap Y = \emptyset$. We will denote the external support formula by $ES_{\Pi,Y}$.

Condition (b) of the main theorem suggests the propositional theory $T_b(\Pi)$ which consists of

(i) the implications¹¹

$$B \wedge F \supset A \quad (6)$$

for all rules (4) in Π , and

(ii) the implications $\bigvee_{p \in Y} p \supset ES_{\Pi,Y}$ for all sets Y of atoms that occur in Π .

On the other hand, condition (c) of the main theorem suggests the propositional theory $T_c(\Pi)$, which consists of

(i) the implications (6) for all rules (4) in Π , and

(ii) the implications

$$\bigwedge_{p \in L} p \supset ES_{\Pi,L} \quad (7)$$

for all loops L of Π .

Formula (7) is called the *conjunctive loop formula* of L .

Corollary 3 For any finite program Π without classical negation and any set X of atoms, X is an answer set for Π iff X is a model of $T_b(\Pi)$ iff X is a model of $T_c(\Pi)$.

Note that the number of loops is exponential in the worst case. But we cannot do much about this: Lifschitz and Razborov [2004] showed that any equivalent translation from logic programs to propositional formulas involves a significant increase in size assuming a conjecture from the theory of computational complexity which is widely believed to be true.

4 Unfounded Sets and Disjunctive Loop Formulas

Since conditions (b) and (c) in the statement of the main theorem are equivalent to each other, any intermediate condition between the two also characterizes answer sets. In this section, we study two such conditions related to some earlier work.

¹¹We identify ‘not’ with ‘ $\neg$ ’, ‘ $\vee$ ’ with ‘ $\wedge$ ’, and ‘ $\wedge$ ’ with ‘ $\vee$ ’.

4.1 Relation to Unfounded Sets

The first condition is

(d) X satisfies Π , and every nonempty subset of X is externally supported by Π w.r.t. X .

Since (b) implies (d), (d) implies (c), and (a),(b),(c) are equivalent to each other, it is clear that all these conditions are equivalent to each other.

In fact, the equivalence between conditions (a) and (d) was established before for the special case when the rules of the program have the form (3) with $p = n$. It is related to the notion of an unfounded set originally introduced in [Van Gelder et al., 1991] to characterize negative conclusions under the well-founded semantics. Saccá and Zaniolo [1990] showed that answer sets for a nondisjunctive program can be characterized in terms of unfounded sets.¹² Leone et al. [1997] extended the notion of an unfounded set and the theorem by Saccá and Zaniolo to disjunctive programs.

Their definition can be further extended to programs with rules of the form (4) as follows. A set Y of literals is *unfounded* by a program Π w.r.t. a set X of literals if, for each rule (4) in Π such that $A \cap Y \neq \emptyset$, at least one of the following conditions holds: $X \not\models B, F$, $B \cap Y \neq \emptyset$, or $X \cap (A \setminus Y) \neq \emptyset$.

It is easy to see that Y is not unfounded w.r.t. X iff Y is externally supported w.r.t. X in the sense of Section 2.2.

A set X of literals is called *unfounded-free* if it does not have any nonempty subset that is unfounded w.r.t. X . The equivalence between conditions (a) and (d) stated above can be reformulated as follows: X is an answer set for Π iff X satisfies Π and X is unfounded-free. This is a generalization of Corollary 2 from [Saccá and Zaniolo, 1990] and Theorem 4.6 from [Leone et al., 1997] to programs with rules of the form (3).

For finite programs Π without classical negation, condition (d) suggests the following translation T_d into propositional logic, which consists of

(i) the implications (6) for all rules (4) in Π , and

(ii) the implications (7) for all nonempty sets L of atoms that occur in Π .

Note that $T_d(\Pi)$ is a superset of $T_c(\Pi)$. In (ii), L is not empty. If it were, then the implication would be unsatisfiable.

Corollary 4 For any finite program without classical negation, a set of atoms is an answer set for Π iff it is a model of $T_d(\Pi)$.

4.2 Relation to Disjunctive Loop Formulas

Here is another condition intermediate between (b) and (c):

(e) X satisfies Π , and, for every loop $L \subseteq X$, every set of literals that has a common element with L is externally supported by Π w.r.t. X .

Given a finite program Π whose rules have the form (4) where all l_i are atoms, the propositional theory $T_e(\Pi)$ consists of

¹²Their theorem refers to ‘assumption sets’ rather than ‘unfounded sets.’ But as the authors noted, the two notions coincide as far as the theorem is concerned.

- (i) the implications (6) for all rules (4) in Π , and
- (ii) the implications

$$\bigvee_{p \in L} p \supset ES_{\Pi, L} \quad (8)$$

for all loops L of Π .

Formula (8) is called the *disjunctive loop formula* of L .

Corollary 5 *For any finite program Π without classical negation, a set of atoms is an answer set for Π iff it is a model of $T_e(\Pi)$.*

In fact, if Π is nondisjunctive, (8) is essentially the loop formula defined by Lin and Zhao [2004]. In the rest of this section, we give a more precise description of the relationship between this special case of Corollary 5 and the Lin/Zhao theorem.

Note first that when Π is nondisjunctive, $T_e(\Pi)$ consists of

- (i) the implications
- $$B \wedge F \supset l_1 \quad (9)$$

for all rules (2) in Π , and

- (ii) the implications
- $$\bigvee_{p \in L} p \supset \bigvee_{\substack{l_1 \leftarrow B, F \in \Pi \\ l_1 \in L, B \cap L = \emptyset}} B \wedge F. \quad (10)$$

for all loops L of Π .

We now review the Lin/Zhao theorem. The completion of the same program Π , $Comp(\Pi)$, consists of the equivalences

$$l_1 \equiv \bigvee_{l_1 \leftarrow B, F \in \Pi} B \wedge F$$

for all atoms l_1 that occur in Π .

By $LF(\Pi)$ we denote the set of the disjunctive loop formulas (10) for all non-trivial loops L .

Theorem 1 [Lin and Zhao, 2004, Theorem 1] *For any finite nondisjunctive program Π without classical negation, a set of atoms is an answer set for Π iff it is a model of $Comp(\Pi) \cup LF(\Pi)$.*

To see why $Comp(\Pi) \cup LF(\Pi)$ is equivalent to $T_e(\Pi)$, observe first that $Comp(\Pi)$ can be rewritten as the set of implications “right-to-left”

$$\left(\bigvee_{l_1 \leftarrow B, F \in \Pi} B \wedge F \right) \supset l_1, \quad (11)$$

and “left-to-right”

$$l_1 \supset \bigvee_{l_1 \leftarrow B, F \in \Pi} B \wedge F. \quad (12)$$

Implications (11) can be further broken into implications (9). In implications (12), every singleton set $\{l_1\}$ is a loop. If the loop is non-trivial, then $LF(\Pi)$ contains the corresponding implication (10), which is stronger than (12). Consequently, in the presence of $LF(\Pi)$, implications (12) for all non-trivial loops $\{l_1\}$ can be dropped. On the other hand, if $\{l_1\}$ is trivial, then $l_1 \notin B$, so that (12) coincides with (10) (where $L = \{l_1\}$). To sum up, $Comp(\Pi) \cup LF(\Pi)$ can be equivalently rewritten as the set consisting of formulas (9), formulas (10) for trivial loops, and formulas (10) for non-trivial loops (they form $LF(\Pi)$). This set is exactly $T_e(\Pi)$.

5 Loop Formulas for Programs with Nested Expressions

Lifschitz *et al.*[1999] extended the answer set semantics to programs with nested expressions. “Formulas” defined in that paper allow negation as failure (*not*), conjunction ($\wedge$) and disjunction ($\vee$) to be nested arbitrarily. A program with nested expressions is a set of rules of which both heads and bodies are formulas. The following program Π_5 , for instance, is a program with nested expressions which is not in canonical form because the first rule has a disjunction in its body.

$$p \leftarrow q ; \text{not } r \quad q \leftarrow p \quad r \leftarrow \text{not } p.$$

For the semantics of such programs, along with the definition of satisfaction, we refer the reader to Section 2 of [Erdoğan and Lifschitz, 2004]. Recall that an occurrence of a formula F in a formula G is *singular* if the symbol before this occurrence is $\neg$; otherwise, the occurrence is *regular* [Lifschitz *et al.*, 1999]. For any formula G , by $poslit(G)$ we denote the set of all literals having a regular occurrence in G that is not in the scope of negation as failure. For instance, $poslit(p, \text{not } q, (\text{not not } r; \neg s)) = \{p, \neg s\}$.

For a set Y of literals, by $F_{\perp}^Y$ we denote the formula obtained from a formula F by replacing all regular occurrences of literals from Y that are not in the scope of negation as failure with $\perp$; by $\Pi_{\perp}^Y$ we denote the program obtained from a program Π by the same substitution. In application to canonical programs, this operation is closely related to the concept of external support:

Proposition 1 *Let Π be a canonical program, and X a set of literals satisfying Π . For any set Y of literals, Y is externally supported by Π w.r.t. X iff X does not satisfy $\Pi_{\perp}^Y$.*

For example, for program Π_2 in Section 2.1 we checked that $\{p, q\}$ is externally supported w.r.t. $\{p, q\}$. On the other hand, the program $(\Pi_2)_{\perp}^{\{p, q\}}$ is

$$\perp \leftarrow \perp \quad \perp \leftarrow \text{not } r \quad r \leftarrow \text{not } p,$$

and we can check that $\{p, q\}$ does not satisfy the program, in accordance with Proposition 1.

Proposition 1 shows that the main theorem in Section 2.3 can be stated without mentioning external support: in conditions (b) and (c), we can replace “ Y is externally supported by Π w.r.t. X ” with “ X does not satisfy $\Pi_{\perp}^Y$.” This fact can be used to generalize the main theorem to arbitrary programs.

First we need to generalize the definition of a positive dependency graph to programs with nested expressions. The *positive dependency graph* of a program Π with nested expressions is the directed graph G such that

- its vertices are the literals occurring in Π , and
- its edges go from each literal in $poslit(Head)$ to each literal in $poslit(Body)$ for each rule $Head \leftarrow Body$ of Π .

If Π is canonical, this definition reduces to the earlier definition. The positive dependency graph of Π_5 is the same as the positive dependency graph of Π_2 as shown in Figure 1.

Once we define a positive dependency graph, the definition of a loop given in Section 2.3 remains the same for programs with nested expressions.

Main Theorem For any program Π with nested expressions and any consistent set X of literals, the following conditions are equivalent:

- (a) X is an answer set for Π .
- (b) X satisfies Π , and, for every set Y of literals that has a common element with X , X does not satisfy $\Pi_{\perp}^Y$.
- (c) X satisfies Π , and, for every loop L of Π that is contained in X , X does not satisfy $\Pi_{\perp}^L$.

The translations T_b through T_e can be extended to finite programs Π with nested expressions that do not contain classical negation. For instance, the extended theory $T_b(\Pi)$ consists of (i) the implications $Body \supset Head$ for all rules $Head \leftarrow Body$ in Π , and (ii) the implications $\bigvee_{p \in Y} p \supset \neg \Pi_{\perp}^Y$ for all sets Y of atoms that occur in Π .

6 Conclusion

The following are the main contributions of this paper:

- We reformulated the definition of a loop formula so that loop formulas became a generalization of completion.
- We generalized the definition of a loop formula and the Lin/Zhao theorem to programs with nested expressions, not necessarily finite.
- We presented a model-theoretic account of loop formulas—the concept of an externally supported set—and showed that loop formulas are related to assumption sets and to unfounded sets.

It is interesting to note that the computational methods used in DLV and in SAT-based answer set solvers are related to each other, in view of our main theorem: DLV uses condition (d) [Leone *et al.*, 1997, Theorem 4.6] for finding answer sets for disjunctive programs, and ASSAT and CMODELS use condition (e) for nondisjunctive programs.

Acknowledgements

I am grateful to Vladimir Lifschitz and Fangzhen Lin for many useful discussions. Vladimir helped me develop the idea and improve the presentation significantly. I am also grateful to anonymous referees for their comments. Wolfgang Faber suggested that there may be a relationship between loop formulas and unfounded sets. Nicola Leone gave pointers to earlier work on unfounded sets and on assumption sets. This work was partially supported by NSF under Grant IIS-0412907.

References

- [Apt *et al.*, 1988] Krzysztof Apt, Howard Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, San Mateo, CA, 1988.
- [Baral and Gelfond, 1994] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994.
- [Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.
- [Erdem and Lifschitz, 2003] Esra Erdem and Vladimir Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 3:499–518, 2003.
- [Erdoğan and Lifschitz, 2004] Selim Erdoğan and Vladimir Lifschitz. Definitions in answer set programming. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 114–126, 2004.
- [Fages, 1994] François Fages. Consistency of Clark’s completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.
- [Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080, 1988.
- [Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [Giunchiglia *et al.*, 2004] Enrico Giunchiglia, Yuliya Lierler, and Marco Maratea. SAT-based answer set programming. In *Proc. AAAI-04*, pages 61–66, 2004.
- [Inoue and Sakama, 1998] Katsumi Inoue and Chiaki Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35:39–78, 1998.
- [Lee and Lifschitz, 2003] Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs. In *Proc. ICLP-03*, pages 451–465, 2003.
- [Leone *et al.*, 1997] Nicola Leone, Pasquale Rullo, and Francesco Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. *Information and Computation*, 135(2):69–112, 1997.
- [Lifschitz and Razborov, 2004] Vladimir Lifschitz and Alexander Razborov. Why are there so many loop formulas? *ACM Transactions on Computational Logic*, 2004. To appear.
- [Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.
- [Lin and Zhao, 2004] Fangzhen Lin and Yuting Zhao. AS-SAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157:115–137, 2004.
- [Przymusiński, 1989] Teodor Przymusiński. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, 5:167–205, 1989.
- [Saccà and Zaniolo, 1990] Domenico Saccà and Carlo Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proceedings of Symposium on Principles of Database Systems (PODS)*, pages 205–217, 1990.
- [Van Gelder *et al.*, 1991] Allen Van Gelder, Kenneth Ross, and John Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, 1991.

Planning with Loops

Hector J. Levesque*

Dept. of Computer Science
University of Toronto
Toronto, Ont. M5S 3H5
hector@cs.toronto.edu

Abstract

Unlike the case for sequential and conditional planning, much of the work on iterative planning (planning where loops may be needed) leans heavily on theorem-proving. This paper does the following: it proposes a different approach where generating plans is decoupled from verifying them; describes the implementation of an iterative planner based on the situation calculus; presents a few examples illustrating the sorts of plans that can be generated; shows some of the strengths and weaknesses of the approach; and finally sketches the beginnings of a theory, where validation of plans is done offline.

1 Introduction

The vast majority of the work in AI planning today deals with *sequential* planning, generating a sequence of actions to achieve a goal. A smaller community is concerned with *conditional* planning where plans can be tree-like structures, and an even smaller community is concerned with *iterative* planning, where plans can be graph-like structure with loops. The reason for this is clear: sequential planning admits interesting applications, and yet is already quite difficult, even under the assumption of complete knowledge about the initial state.

The bulk of the work on iterative planning is based on theorem-proving (see [Biundo, 1994] for a survey, but see also Section 4 below for some exceptions). From this perspective, plans are viewed as programs, and planning as a kind of program synthesis [Manna and Waldinger, 1980]. This is a notoriously difficult problem, and reasoning about the correctness of programs with loops, *e.g.* in terms of partial correctness and termination, requires mathematical induction and non-trivial algebra. The difficulty, in other words, is not at all like the difficulty with sequential planning where the size of the search space for long plans is the main problem; even short iterative programs can be quite difficult to reason

about. Stephan and Biundo [1996] say “Plan generation on this level is an interactive process with non-trivial inferences that in our opinion, which is shared by other authors as well [citations omitted], cannot be carried out in a fully automatic way.” As far as we can tell, they would say the same today.

So is fully automated iterative planning completely hopeless? Perhaps. But faced with an intractable reasoning problem, we can look for compromises. In this paper, we forego the strong guarantees of correctness provided by the theorem-proving approach. We consider a new way of generating iterative plans that does not traffic in loop invariants, non-negative decreasing expressions, or any of the other items associated with proving programs correct. The resulting plans will come with much weaker guarantees; plan validation will need to be done separately.

The application we have in mind is the sort of high-level programming typical of *cognitive robotics*, *e.g.* [Lespérance *et al.*, 1999]. Here we expect users to provide programs that tell a robot what to do at a very high-level, with considerable nondeterminism left to the robot to deal with at runtime. Part of the nondeterminism can be in the form of declarative goals to achieve. They will require planning on the part of the robot, but the plans are expected to be small relative to the overall mission of the robot. The main contribution of this paper is a new way of generating small plans with loops in this setting.

In the rest of this section, we describe a motivating example, define the class of plans we are searching for, and present the general approach. In Section 2, we discuss the implementation of a system called KPLANNER and the novel way it generates loops. In Section 3, we present three examples of KPLANNER in use. In Section 4, we discuss its limitations and related work. In Section 5, we present the beginnings of a theoretical foundation. In Section 6, we conclude.

1.1 A motivating example

The problem we wish to consider is that of chopping down a tree and putting away the axe [Sardina *et al.*, 2004]. We have at our disposal two primitive actions: *chop*, which hits the tree once with the axe, assuming the tree is up and the axe is available, and *store*, which puts the axe away, assuming it is available. We observe the following:

1. With no additional information, the problem is insoluble. There is no way to know when or if the tree will go down, and when to put away the axe.

*This research was done while I was away on sabbatical at York University in Toronto, RWTH in Aachen, Universita di Roma La Sapienza, the University of New South Wales in Sydney, and Simon Fraser University in Vancouver. I am very grateful to my hosts at these institutions for providing me with a wonderful environment for not doing teaching and administration. Sabbaticals rock!

2. If we are told that the tree will go down if it is hit 3 times, the problem is solved with the following sequential plan:
 chop ; chop ; chop ; store
3. If we are told that the tree will go down if it is hit *at most* 2 times, the problem is again insoluble. But if we are given a third action, *look*, which is a *sensing action* [Reiter, 2001] telling us whether the tree is down or up, then the problem can be solved with the following conditional plan:

```

CASE look OF
  -down: store
  -up: chop ;
    CASE look OF
      -down: store
      -up: chop ; store
    ENDC
ENDC

```

4. If all we are told is that the tree will *eventually* go down if it is hit repeatedly, and we have the *look* action, the problem can be solved with the following iterative plan:

```

LOOP
  CASE look OF
    -down: EXIT
    -up: chop ;
      NEXT
  ENDC
ENDL ;
store

```

Let us call this plan *TC* (TreeChop) for reference.

So in the most general case where the goal is achievable, an iterative plan like *TC* is necessary to achieve it. Moreover, *TC* is *general*, in that it handles all the cases where a conditional plan would be sufficient too.

1.2 Robot programs

Once we move beyond simple sequential plans, it becomes necessary to specify exactly what we mean by a plan. It is convenient here to use a variant of the *robot programs* of [Levesque, 1996] for this purpose. Assume we are given a set of primitive actions, some of which may be sensing actions each having a finite set of possible sensing results. Robot programs and their execution are defined by the following:

1. **nil** is a robot program executed by doing nothing;
2. for any primitive action *A* and robot program *P*, **seq**(*A*, *P*) is a robot program executed by first performing *A*, ignoring any result, and then executing *P*;
3. for any primitive action *A* with possible sensing results R_1 to R_k , and for any robot programs P_1 to P_k , **case**(*A*, [**if**(R_1 , P_1), ..., **if**(R_k , P_k)]) is a robot program executed by first performing *A*, and then on obtaining the sensing result R_i , continuing by executing P_i ;
4. if *P* and *Q* are robot programs, and *B* is the result of replacing in *P* some of the occurrences of **nil** by **exit** and the rest by **next**, then **loop**(*B*, *Q*) is a robot program, executed by repeatedly executing the body *B* until the execution terminates with **exit** (rather than **next**), and then going on by executing the continuation *Q*.

A formal definition of the execution of robot programs in the situation calculus is given in [Levesque, 1996]. The example plans shown earlier, including *TC*, are pretty-printed versions of robot programs. Although quite limited in form (no recursion, no logical expressions, no fluents, no variables), there is a precise sense in which robot programs (with the inclusion of five special primitive actions) are *universal* [Lin and Levesque, 1998]. So for us, planning is this: given a goal, find a robot program that achieves it.

1.3 The planning approach

We are going to try to solve planning problems like the tree chopping where some unknown quantity must be dealt with. Specifically, we assume that among the fluents we are dealing with, there will be a single distinguished one, which we will call the *planning parameter*, that has the following properties: (1) its value is not known or even bounded at plan time, and (2) no loops would be required to achieve the goal if its value were known. In other words, loops are required (if at all) only to deal with an unknown and unbounded planning parameter. For tree chopping, the planning parameter is the number of chops needed to fell the tree.

Given an application domain with a planning parameter *F*, rather than generating a plan and proving that it works for *all* values of *F* (e.g. by using partial correctness and termination), we will work with two bounds on *F* and try to find plans with loops using these bounds. In particular, we assume the following:

- the user will provide a constant N_1 (called the *generating bound*) and we *generate* a plan (possibly containing some number of loops) that is provably correct under the assumption that $F \leq N_1$;
- the user will provide a second larger constant N_2 (called the *testing bound*) and we *test* that the plan generated in the first step is also provably correct under the assumption that $F \leq N_2$.

So the only guarantee we get out of this is that the plan we return will be correct assuming that $F \leq N_2$. This is far from foolproof. For example, the conditional plan for tree chopping presented above works correctly for $N_1 = 1$ and $N_2 = 2$, but fails when $F \geq 3$. Suppose, however, that the user had specified $N_1 = 1$ and $N_2 = 100$. Then the smallest conditional plan that satisfies both bounds would have $3 \times 100 + 1$ actions in it. If we generate plans using N_1 only, and we *consider smaller plans before larger ones*, we will generate desired iterative plans well before we encounter undesired conditional ones (and see the theory in Section 5).

2 The planner

So the planning setup, then, is the following:

- the user provides a *problem specification* defined by a list of primitive actions and fluents, and formulas characterizing the initial and goal states, and for each action, its preconditions, effects, and sensing results;
- the user also identifies the parameter *F* and supplies the two numbers N_1 and N_2 ;

- we generate a plan that is correct for $F \leq N_1$; because N_1 is small, this search can be done reasonably efficiently, as we will see;
- we test the plan to see if it is correct for $F \leq N_2$; because this involves only testing a given plan, this can also be done reasonably efficiently, even for large N_2 .

In practice, and on the examples considered to date, even for very small N_1 and for very large N_2 , almost all of the time is spent finding a plan that works for $F \leq N_1$.

The planner, called KPLANNER, is written in Prolog, and consists of three main modules: a plan tester, a plan generator, and a formula evaluator used by the other two modules.¹

We will describe the formula evaluator in more detail when we look at some examples below. For now, it suffices to note that we need to be able to handle both the *projection* task (determining if a formula is true after a sequence of actions) and the *legality* task (determining if an action can be executed). We use regression for both [Reiter, 2001]. Moreover, we need to be able to incorporate the putative results of sensing, and use them in the evaluation. So instead of just keeping track of a sequence of actions performed to date, we maintain a *history* consisting of pairs of actions and sensing results. (For actions without sensing results, we use *ok* as the result.) Finally, we need to be able to determine when a sensing result cannot occur based on the history to date. For example, for tree chopping where we know that $F \leq 2$, the *look* action cannot produce a sensing result of *up* in a history that contains two legal chop actions.

2.1 Testing plans

Given a plan P and a testing bound N , we need to determine if P correctly achieves the goal assuming that $F \leq N$. Since we have a bound on F , we do not need to prove a theorem, but can simulate the execution of P for all possible results of the sensing actions and confirm that the goal is satisfied in all cases. We have the following: A plan P achieves a goal G starting in history H (initially empty) if

1. $P = \mathbf{nil}$ and G holds in history H ;
2. $P = \mathbf{seq}(A, P')$, the precondition of A holds in H , and P' achieves G in history $H \cdot (A, ok)$;
3. $P = \mathbf{case}(A, L)$, the precondition of A holds in H , and for each $\mathbf{if}(R_i, P_i) \in L$ such that R_i is a possible sensing result for A given that $F \leq N$, P_i achieves G in history $H \cdot (A, R_i)$;
4. P is a loop that unwinds to Q , and Q achieves G in history H .

The unwinding of $\mathbf{loop}(B, C)$ is B but with all occurrences of **exit** replaced by C and with all occurrences of **next** replaced by $\mathbf{loop}(B, C)$ itself.² Observe that unwinding a loop produces a plan that executes the same way as the original loop. This will be significant when it comes to generating plans.

¹The code for KPLANNER and a number of examples can be found at <http://www.cs.toronto.edu/cogrobo>.

²To guard against loops that run forever, such as $\mathbf{loop}(\mathbf{next}, \mathbf{nil})$, we also need an upper bound on the total number of unwindings that will be performed.

2.2 Generating plans

Given a generating bound N , we use a variant of *progressive planning* to produce a plan that correctly achieves the goal assuming that $F \leq N$. We have the following: To find a plan that achieves a goal G starting in history H (initially empty),

1. if G holds in H , return **nil**;
2. otherwise, select a primitive action A whose precondition is satisfied in H (and maybe other criteria too);
3. if A has no sensing results, do the following:
 - (a) find a plan P that achieves G starting in $H \cdot (A, ok)$;
 - (b) return the plan $\mathbf{seq}(A, P)$;
 if A has sensing results, then do the following:
 - (a) for each sensing result R_i that is possible in H given that $F \leq N$, find a plan P_i that achieves G starting in history $H \cdot (A, R_i)$;
 - (b) return the plan $\mathbf{case}(A, L)$, where L is the list of $\mathbf{if}(R_i, P_i)$ for the R_i and P_i from the previous step (the remaining sensing results R_i are impossible, and so the corresponding P_i are “don’t care”);
4. if the plan in the previous step is the unwinding of a loop, return the loop as well (as described below).

2.3 Generating loops

The key question is this: Where are plans with loops going to come from, if not from the proof of a general, universal theorem? As we suggested in the generation procedure above, they can come from sequential and conditional plans that have already been generated.

To see how this works, consider two conditional plans that are correct for tree chopping given that $F \leq 1$:

<pre> CASE look OF -down: store -up: chop ; store ENDC </pre>	<pre> CASE look OF -down: store -up: chop ; CASE look OF -down: store -up: don't care ENDC ENDC </pre>
---	--

The one on the right does a sensing action that is not needed since that *up* sensing result is impossible for $F \leq 1$. Moreover, that plan remains correct for $F \leq 1$ for any substitution of the “don’t care.” The key observation here is that there is a substitution for which that plan becomes the unwinding of a loop: if we replace the “don’t care” by the robot program $\mathbf{seq}(\mathbf{chop}, TC)$ where TC is the plan above, then the program that obtains is an unwinding of TC .³ The conclusion: TC is also correct for $F \leq 1$ and can be returned as a potential plan to be tested for the larger bound (in this case, successfully).

But for this idea to be practical, it must be possible to quickly check if a plan matches the unwinding of a loop (or the unwinding of an unwinding etc.). How can we do this without just guessing at the loop? Here is where using Prolog as our implementation language pays off: we can write an *unwind* predicate, so that $\mathbf{unwind}(P, Q)$ holds if loop

³More precisely, it is the unwinding of an unwinding of TC .

```

% P is a loop that unwinds to Q.
unw(P,Q) :- P=loop(B,C), sub(B,P,C,Q).

% sub(B,X,Y,Q) holds when Q is the result
% of replacing in B each 'exit' by Y and
% each 'next' by an unwinding of X
sub(_,-,-,Q) :- var(Q), !.
sub(exit,X,Y,Q) :-not X=loop(exit,-), Q=Y.
sub(next,X,-,Q) :-not X=loop(next,-), unw(X,Q).
sub(seq(A,P),X,Y,seq(A,Q)) :-sub(P,X,Y,Q).
sub(case(A,U),X,Y,case(A,V)) :-subl(U,X,Y,V).
sub(loop(G,P),X,Y,loop(G,Q)) :-sub(P,X,Y,Q).

subl([],_,_,[]).
subl([if(R,P)|U],X,Y,[if(R,Q)|V]) :-
    sub(P,X,Y,Q), subl(U,X,Y,V).

```

Figure 1: Prolog code for an unwind predicate

P unwinds to Q , but then call it by passing it a Q and having it return a P . Simplified code for doing this is in Figure 1. It is easy to confirm that when the Q is the conditional plan on the right above (with a Prolog variable as the “don’t care”), the first P it returns is the desired iterative plan TC .

This method of generating loops has turned out to be significant. The alternative of enumerating all possible plans containing loops is not practical even for very small plans.

3 The planner in action

We are now ready to consider KPLANNER in action on some problem specifications provided by the user. The representation we use is a variant of the one in INDIGOLOG [de Giacomo and Levesque, 1999; Sardina *et al.*, 2004] based on the situation calculus [McCarthy and Hayes, 1981; Reiter, 2001]. The user supplies the definition of nine predicates in Prolog, which we describe below.⁴

As in INDIGOLOG, all fluents in KPLANNER are functional. Unlike INDIGOLOG, fluents are interpreted *epistemically*, in that we take them to have more than one possible value, according to what is currently known [Vassos, 2004]. This allows us to reason about sensing without some of the disadvantages of possible worlds [Demolombe and Parra, 2000].

The predicates described below take as their arguments *conditions* which are logical formulas, closed under boolean operators and quantifiers. The atomic formulas are arbitrary Prolog goals, except that they may contain fluents. These are evaluated by replacing the fluents by their values and then calling Prolog on the result. We say that a formula is *possibly true* if the goal succeeds for some possible value of the fluents; the formula is *known to be true* if the goal succeeds for every possible value of the fluents.

The Prolog predicates defined by the user are (for action a , fluent f , sensing result r , condition c and arbitrary value v):

- `prim_fluent( $f$ )`, declares f as fluent;
- `prim_action( $a$ , [ $r_1, \dots, r_n$ ])`, declares a to have the r_i as possible sensing results; when $n = 1$, the action is considered to provide no sensing information;

⁴There are also directives to help control the search, which we do not describe further here.

```

prim_fluent(axe).
prim_fluent(tree).
prim_fluent(chops_max).

prim_action(chop,[ok]).
prim_action(look,[down,up]).
prim_action(store,[ok]).

poss(chop,and(axe=out,tree=up)).
poss(look,true).
poss(store,axe=out).

init(axe,out).
init(tree,up).
init(tree,down).

causes(store,axe,stored,true).
causes(chop,tree,down,true).
causes(chop,tree,up,true).
causes(chop,chops_max,X,X is chops_max-1).

settles(look,X,tree,X,true).
rejects(look,up,chops_max,0,true).
settles(look,down,chops_max,0,true).

parm_fluent(chops_max).
init_parm(generate,chops_max,1).
init_parm(test,chops_max,100).

```

Figure 2: A problem specification for tree chopping

- `poss( $a, c$ )`, c is the precondition for a ;
- `init( $f, v$ )`, v is a possible initial value for f ;
- `causes( $a, r, f, v, c$ )`, when c holds, a causes f to get value v ; more precisely, any v for which c is possible becomes a possible value of f ; (the r is optional)
- `settles( $a, r, f, v, c$ )`, when c is known, after doing a and getting result r , f is known to have value v ;
- `rejects( $a, r, f, v, c$ )`, when c is known, after doing a and getting result r , f is known not to have value v .

The final two predicates are not part of the action theory, but are used to specify the planning parameter, and its possible values for generating and for testing:

- `parm_fluent( $f$ )`, fluent f is the planning parameter;
- `init_parm( $w, f, v$ )`, where w is `generate` or `test`, v is a possible initial value for the planning parameter f .

3.1 The tree chopping example

In its most direct formulation, we would formalize tree chopping using a fluent *chops_needed* as the planning parameter. But then to handle a testing bound of 100 would require us to deal with 100 possible values for this fluent. Instead, it is sufficient to keep track of the maximum of these possible values, which we call *chops_max*.

The full problem specification in this language for the tree chopping example is in Figure 2. Since it is not known whether the tree is up or down initially, there are two possible initial values for the *tree* fluent, and similarly after a *chop* action. (Actions like this are sometimes called *nondeterministic*

```

prim_fluent(acc(N)) :- N=1 ; N=2.
prim_fluent(input).    % the unknown fluent

prim_action(incr_acc(N),[ok]) :- N=1 ; N=2.
prim_action(test_acc(1),[same,diff]).

poss(incr_acc(_),true).
poss(test_acc(1),true).

causes(incr_acc(N),acc(N),V,V is acc(N)+1).
settles(test_acc(1),same,input,V,V=acc(1)).
rejects(test_acc(1),diff,input,V,V=acc(1)).

init(acc(_),0).
parm_fluent(input).
init_parm(generate,input,V) :- V=1 ; V=2.
init_parm(test,input,V) :- V=1 ; V=2; V=3.

```

Figure 3: A counting example

in the literature.) The *look* action is what settles its value. In addition, if *look* reports *up*, then *chops_max* cannot be 0. So if we know that *chops_max* is 0 (as a result of having performed some *chop* actions), the *up* result is impossible. We will show the output of a run of KPLANNER on the next example. For this one, suffice it to say that it finds *TC* in .11 seconds.⁵

3.2 A counting example

We turn now to a very different example involving some simple counting. The problem is this: We have two accumulators and some unknown integer input k , where $k > 0$. The primitive actions are: *incr_acc(n)*, increment accumulator n (both start at 0); and *test_acc(1)*, sense if the first accumulator has the same value as the input. The goal is to make the second accumulator have the value $2k - 1$.⁶

The complete problem specification is in Figure 3. There are three fluents, *acc(1)*, *acc(2)*, and *input*, the last of which is the planning parameter. A run of KPLANNER with the output it produces is in Figure 4. KPLANNER works by iterative deepening, and there are numbers in the output to indicate the level. An 'x' indicates a generated plan that was sufficient for the generating bound, but that did not work for the testing bound (many of which were omitted from the figure). Note the multiplication in the goal. Nothing in the specification said anything about multiplication. Although progressive planning has serious disadvantages, one advantage it does have is that all we need to be able to do is test if a goal condition is satisfied; we do not need to reason about the goal in a more analytic way.

3.3 A searching example

We now turn to a much more complex example, that of searching an unbounded binary tree for a target node.⁷ More precisely the primitive actions are (1) *check_node_type*: sense

⁵All runs were done in Eclipse Prolog version 5.7 on a Mac G5 single 1.6GHz processor with 512MB memory.

⁶This is an abstract variant of the problem of building two copies of a stack of coloured blocks.

⁷This can be thought of as a simplified version of searching for a file in a Unix directory of arbitrary depth.

```

The goal:  acc(2) is 2 * input - 1
0 1 2 3 4 5xx ... 6xx ... 7xx ...
A plan is found after 1.42 seconds.
-----
incr_acc(1) ;
LOOP
CASE test_acc(1) OF
  -same: EXIT
  -diff:
    incr_acc(1) ;
    incr_acc(2) ;
    incr_acc(2) ;
    NEXT
ENDC
ENDL ;
incr_acc(2)

```

Figure 4: Doing the arithmetic

if the current node is a target node, a non-target leaf node, or a non-target internal node (having left and right children); (2) *push_down_to(x)*: go down from an internal node to the left or right, x ; and (3) *pop_up_from*: return from a child node. A few moments thought should convince the reader that this problem cannot be solved without additional storage, and so we assume that *push_down_to(x)* pushes the direction x onto an internal *stack*, and that *pop_up_from* pops the stack and produces the popped value as a sensing result.

The rest of the specification is too large to display here. We use a fluent *depth_max* much like *chops_max* so that we cannot get a node type of *internal* when *depth_max* is 0.

But when do we know that we cannot get a node type of *leaf*? This is a bit trickier. The answer is: when there are no more nodes left to explore! For example, assuming we search left branches before right ones, then when we are on the rightmost branch of a tree, we can only get node types of *internal* or *target*. This is reflected in the following:

```

rejects(check_node_type, leaf, stack, S,
all(x, member(x, S), last_dir(x))).

```

Getting a result of *leaf* rejects any stack of moves whose members are all for the last direction available. Without this constraint, the problem is insoluble; with it, we get the very nice plan shown in Figure 5. Note that KPLANNER generates a nested loop; we believe that this nesting is required here.

4 Discussion and related work

Because of the loops, the three examples presented here are clearly beyond the scope of existing sequential and conditional planners. As far as we know, no planner based on fully automated theorem-proving can generate the three plans either. Two other camps not based on theorem-proving have considered an interesting special case of iterative planning, for what might be called *repeated attempt* problems.

First, with probabilities. Consider an action a that has a non-zero probability of making some G true. If we assume repetitions of a have independent outcomes, then an iterative plan like *TC* will achieve G with probability 1. This is just right for trying to pick up a block [Ngo and Haddawy, 1995] or to find a good egg [Bonet and Geffner, 2001]. However, independence is untenable for tree chopping, and it is not

```

The goal: current = target
0 1 2 3 4 5 6xxxxx
A plan is found after 0.07 seconds.
-----
LOOP
  CASE check_node_type OF
    -target: EXIT
    -leaf:
      LOOP
        CASE pop_up_from OF
          -left: EXIT
          -right: NEXT
        ENDC
      ENDL ;
      push_down_to(right) ;
      NEXT
    -internal:
      push_down_to(left) ;
      NEXT
  ENDC
ENDL

```

Figure 5: Searching a binary tree

clear what to replace it with. It is even less clear how to exploit probabilities in the counting or search examples above.

A related approach is the model-checking of [Cimatti *et al.*, 2003], where planning problems are cast as finite state systems. For tree chopping, if we are willing to ignore the planning parameter, this works out perfectly: they would get four states and generate (the equivalent of) *TC* almost instantly. However, without that information about the planning parameter, they are forced to conclude that there is no “strong” solution to the problem, only a “strong cyclic” one. Indeed they can never generate a strong solution when loops are required. The counting and search examples above, which depend on the planning parameter in a more direct way, would thus appear to be outside their scope as well.

KPLANNER has its own limitations, however. In particular, it does not scale at all well as the search space grows, even for seemingly easy problems. Consider the problem of getting some good eggs into a bowl [Bonet and Geffner, 2001]. For just one egg, we need a plan with a single loop like in Figure 6. To get three good eggs, we would need a plan consisting of three copies of this plan, strung together sequentially. Without an additional sensing action to tell us if we have enough good eggs, there is not a more compact solution. So with this we can force KPLANNER to generate long plans.

And how well does it do? Here are some timing results:

Number of good eggs	Size of plan (unwound)	Number of backtracks	Running time in seconds
1	6	1	.004
2	9	6	.02
3	12	42	1.41
4	15	702	1681.
5	18	?????	> 3 weeks

As is very clear, unless the search space can be limited in some way, *e.g.* by forward filtering [Bacchus and Kabanza, 2000], KPLANNER is practical only for small plans, in keep-

```

LOOP
  break_next_egg_into_dish ;
  CASE sniff_dish OF
    -good_egg: EXIT
    -bad_egg:
      discard_dish_contents ;
      NEXT
  ENDC
ENDL ;
transfer_dish_contents_to_bowl

```

Figure 6: Getting one good egg into a bowl

ing with the cognitive robotics application sketched above. We might say that it is better suited for small but difficult problems than for large but easy ones. For example, it quickly finds a nice solution to the more general problem of getting an arbitrary number of good eggs in the bowl, where a second sensing action now determines when there are enough.

5 Towards a theory

KPLANNER does a good job of synthesizing small plans with loops, but without a strong guarantee of correctness. While our approach has been to construct a plan in a way that does not require simultaneously proving its correctness, it would of course be very useful to know that the plans were nonetheless correct. Are there conditions under which a plan that works for values of the planning parameter F up to the testing bound will be guaranteed to work for *all* values of F ? Without looking too hard for a free lunch, we can sketch a promising direction for further research.⁸

Let $W(k, \sigma)$ stand for the proposition that if we start in any initial state where $F = k$, and we perform the action sequence σ , we end up in a goal state. Let us call a planning problem *simple* wrt action A if it satisfies the following:

1. Sensing can only tell us whether or not the initial value of F was equal to the number of A actions done so far.
2. If an action sequence is legal starting in one initial state but not in another, then at some point in the sequence, the sensing results would be different.
3. If $W(k, \alpha \cdot \beta)$ and $W(k + m, \alpha \cdot \sigma \cdot \beta)$ and $W(k', \alpha \cdot \gamma)$, where $k' > k$, then $W(k' + m, \alpha \cdot \sigma \cdot \gamma)$.

The last condition says that if it was sufficient to add σ in going from k to $k + m$, then that σ can also be used for any larger k' . This ensures that loops depend only on the F .

Both the tree chopping and the counting examples can be shown to be simple (wrt *chop* and *incr_acc(1)* respectively). The search example, on the other hand, is nowhere near simple. We get the following theorem:

Theorem 1 *Suppose we have a planning problem that is simple with respect to action A , and a robot program P that contains N occurrences of A . If P is a correct plan for all values of $F \leq N + 2$, then P is correct for all values of F .*

This theorem is proved by adapting ideas from the Pumping Lemma of classical automata theory. It suggests a variant

⁸This is joint work with Patrick Dymond and independently with Giuseppe de Giacomo.

of KPLANNER: instead of having the user specify a testing bound, we *compute* a testing bound once a plan P has been generated by counting the occurrences of A in P . Then any plan that passes the test is guaranteed by the theorem to be correct. This applies to simple planning problems only; but we believe that theorems like the above can be found for less restrictive classes of problems. We should not expect to be able to compute testing bounds in this way for *all* planning problems however, since armed with suitable primitive actions, robot programs have the power of Turing machines.

6 Conclusion

We have presented a new way of generating a plan with loops that is not tied to proving a theorem about its correctness. The method involves generating a plan that is correct for a given bound, determining if the plan is the unwinding of a plan with loops, and testing if another unwinding of the plan with loops would also be correct for a larger bound.

Other than the handling of loops, these steps reduce to traditional non-iterative planning, and would benefit from being performed by a more efficient conditional planner, such as the one by Petrick and Bacchus [2004]. Similarly, the winding and unwinding of loops is performed by KPLANNER in a fairly obvious way. Anything that would reduce the number of legal plans considered would speed things up considerably, since for iterative deepening, *all* legal plans of one size must be considered before going on to the next.

On the more theoretical side, we presented a theorem showing that this method of planning is correct for a certain class of simple planning problems. It remains to be seen whether the theorem can be strengthened to include more complex planning problems for which the planner does appear to work, such as the search example presented here.

Acknowledgements

Thanks to Gerhard Lakemeyer and Fahiem Bacchus for helpful comments on an earlier version of the text.

References

- [Bacchus and Kabanza, 2000] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116, 2000.
- [Biundo, 1994] S. Biundo. Present-day deductive planning. In C. Bäckström and E. Sandewell, editors, *Procs. of the 2nd European Workshop on Planning (EWSP-93)*, pages 1–5, Vadstena, Sweden, 1994. IOS Press (Amsterdam).
- [Bonet and Geffner, 2001] B. Bonet and H. Geffner. Gpt: a tool for planning with uncertainty and partial information. In *Proc. IJCAI-01 Workshop on Planning with Uncertainty and Partial Information*, pages 82–87, 2001.
- [Cimatti et al., 2003] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2):35–84, 2003.
- [de Giacomo and Levesque, 1999] G. de Giacomo and H. Levesque. An incremental interpreter for high-level programs with sensing. In H. Levesque and F. Pirri, editors, *Logical Foundation for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 86–102. Springer, Berlin, 1999.
- [Demolombe and Parra, 2000] R. Demolombe and M. Pozos Parra. A simple and tractable extension of situation calculus to epistemic logic. *Lecture Notes in Computer Science*, 1932, 2000.
- [Lespérance et al., 1999] Y. Lespérance, H. Levesque, and R. Reiter. A situation calculus approach to modeling and programming agents. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*. Kluwer, 1999.
- [Levesque, 1996] H. Levesque. What is planning in the presence of sensing? In *Procs. of the 13th National Conference, AAAI-96*, pages 1139–1146, Portland, Oregon, 1996.
- [Lin and Levesque, 1998] F. Lin and H. Levesque. What robots can do: Robot programs and effective achievability. *Artificial Intelligence*, 101:201–226, 1998.
- [Manna and Waldinger, 1980] Z. Manna and R. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980.
- [McCarthy and Hayes, 1981] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Webber and N. Nilsson, editors, *Readings in Artificial Intelligence*, pages 431–450. Kaufmann, Los Altos, 1981.
- [Ngo and Haddawy, 1995] L. Ngo and P. Haddawy. Representing iterative loops for decision theoretic planning. In *Extending Theories of Action: Papers from the 1995 AAAI Spring Symposium*, pages 151–156. AAAI Press, Menlo Park, 1995.
- [Petrick and Bacchus, 2004] R. Petrick and F. Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 2–11, Menlo Park, CA, June 2004. AAAI Press.
- [Reiter, 2001] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [Sardina et al., 2004] S. Sardina, G. de Giacomo, Y. Lespérance, and H. Levesque. On the semantics of deliberation in IndiGolog – from theory to implementation. *Annals of Mathematics and Artificial Intelligence*, 41(2-4):259–299, Aug 2004.
- [Stephan and Biundo, 1996] W. Stephan and S. Biundo. Deduction based refinement planning. In B. Drabble, editor, *Procs. of AIPS-96*, pages 213–220. AAAI Press, 1996.
- [Vassos, 2004] S. Vassos. A feasible approach to disjunctive knowledge in situation calculus. Master’s thesis, Department of Computer Science, University of Toronto, 2004.

Discovering Classes of Strongly Equivalent Logic Programs*

Fangzhen Lin

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Yin Chen

Software Institute
Sun Yat-Sen University
Guangzhou, China

Abstract

We report on a successful experiment of computer-aided theorem discovery in the area of logic programming with answer set semantics. Specifically, with the help of computers, we discovered exact conditions that capture the strong equivalence between a set of a rule and the empty set, a set of a rule and another set of a rule, a set S of two rules and a subset of S with one rule, a set of two rules and a set of another rule, and a set S of three rules and a subset of S with two rules. We prove some general theorems that can help us verify the correctness of these conditions, and discuss the usefulness of our results in program simplification.

1 Introduction

This paper makes two contributions. First, it reports on a successful experiment of computer-aided theorem discovery in logic programming with answer set semantics. Second, it contributes to the theory and practice of logic programming in that the discovered theorems that capture certain classes of strongly equivalent logic programs are new, non-trivial, and lead to new program simplification rules that preserve strong equivalence.

Theorem discovery is a highly creative human process. Generally speaking, we can divide it into two steps: (i) conjecture formulation, and (ii) conjecture verification, and computers can help in both of these two steps. For instance, machine learning tools can be used in the first step, i.e. in coming up with reasonable conjectures, and automated deduction tools can be used in the second step, i.e. in verifying the correctness of these conjectures.

While theorem discovery may make use of learning, these two tasks are fundamentally different. Theorem discovery starts with a theory, and aims at finding *interesting* consequences of the theory, while learning is mostly about induction, i.e. it starts with examples/consequences, and aims at finding a theory that would explain the given examples/consequences.

*Our thanks to Yan Zhang for his comments on an earlier version of this paper. This work was supported in part by HK RGC CERG HKUST6170/04E.

Using computers to discover theorems is an old aspiration. There have been many success stories. For instance, AM [Lenat, 1979] was reported to be able to come up with some interesting concepts and theorems in number theory, and the remarkable system described in [Petkovsek *et al.*, 1996] automates the discovery and proofs of identities, especially hypergeometric identities involving sums of binomial coefficients that are important for the analyses of algorithms. Yet another example where “interesting” theorems can be discovered almost fully automatically is a recent work by Lin [2004] on discovering state invariants in planning domains. Lin showed that there are ways to classify potentially interesting constraints according to their syntactic properties, and these constraints can be easily enumerated for most domains. Furthermore, for many of these constraints whether they are invariants can be checked automatically. As a result, the system described in [Lin, 2004] could discover many common invariants in planning domains, and for the logistics domain, it could even discover a set of “complete” state invariants.

In this paper, we consider the problem of discovering theorems about strongly equivalent logic programs under answer set semantics.

The notion of strongly equivalent logic programs is interesting for a variety of reasons. For instance, as Lifschitz *et al.* [2001] noted, if two sets of rules are strongly equivalent, then one can be replaced by the other in any logic program regardless of the context. Thus identifying strongly equivalent sets of rules is a useful exercise that may have applications in program simplification, and the purpose of this paper is to discover conditions under which a set of rules is strongly equivalent to another. It is important that these conditions need to be computationally effective as in general checking if two sets of rules are strongly equivalent is coNP-complete (c.f. [Lin, 2002]).

To discover these conditions, we follow the methodology of [Lin, 2004] by looking at domains of small sizes first. For instance, to discover for what kinds of rules r_1 and r_2 we have that $\{r_1\}$ is strongly equivalent to $\{r_2\}$, we consider a language with, say three atoms, and enumerate all possible pairs of rules in this language that are strongly equivalent. We then conjecture a condition that would capture exactly this set of pairs of rules, and then try to verify if this conjecture is true in the general case. In [Lin, 2004], a general theorem is proved to automate the verification part. We try to do the

same here by proving some general theorems that will make the verification part easier, almost semi-automatic.

This paper is organized as follows. In the next section, we briefly review the basic concepts of logic programming under answer set semantics. Then in section 3 we state in more precise terms the type of theorems that we want to discover. In section 4 we prove some general theorems that will help us prove these theorems, and in section 5, we describe some of the theorems that we discovered. We then discuss an application to logic program simplification in section 6, and finally we conclude this paper in section 7.

2 Logic programming with answer set semantics

Let L be a propositional language, i.e. a set of atoms. In this paper we shall consider logic programs with rules of the following form:

$$h_1; \dots; h_k \leftarrow p_1, \dots, p_m, \text{not } p_{m+1}, \dots, \text{not } p_n \quad (1)$$

where h_i 's and p_i 's are atoms in L . So a logic program here can have default negation (*not*), constraints (when $k = 0$), and disjunctions in the head of its rules. In the following, if r is a rule of the above form, we write Hd_r to denote the set $\{h_1, \dots, h_k\}$, Ps_r the set $\{p_1, \dots, p_m\}$, and Ng_r the set $\{p_{m+1}, \dots, p_n\}$. Thus a rule r can also be written as $Hd_r \leftarrow Ps_r, \text{not } Ng_r$. The semantics of these programs are given by answer sets as defined in [Gelfond and Lifschitz, 1991]. To save space, we do not give the definition here.

Two such logic programs P_1 and P_2 are said to be *equivalent* if they have the same answer sets, and *strongly equivalent* [Lifschitz et al., 2001] (under the language L) if for any logic program P in L , $P \cup P_1$ and $P \cup P_2$ are equivalent. For example, $\{a \leftarrow b\}$ and $\{a \leftarrow c\}$ are equivalent, but not strongly equivalent. It can be shown that $\{a \leftarrow \text{not } a\}$ and $\{\leftarrow \text{not } a\}$ are strongly equivalent.

Lifschitz et al. [2001] showed that checking for strong equivalence between two logic programs can be done in the logic of here-and-there, a three-valued non-classical logic somewhere between classical logic and intuitionistic logic. Lin [2002] provided a mapping from logic programs to propositional theories and showed that two logic programs are strongly equivalent iff their corresponding theories in propositional logic are equivalent. This result will be used here both for generating example pairs of strongly equivalent logic programs, and for verifying a conjecture. We repeat it here.

Let P_1 and P_2 be two finite logic programs, and L the set of atoms in them.

Theorem 1 [Lin, 2002] P_1 and P_2 are strongly equivalent iff in propositional logic, the following sentence is valid:

$$\left(\bigwedge_{p \in L} p \supset p' \right) \supset \left[\bigwedge_{r \in P_1} \delta(r) \equiv \bigwedge_{r \in P_2} \delta(r) \right], \quad (2)$$

where for each $p \in L$, p' is a new atom, and for each rule r of the form (1), $\delta(r)$ is the conjunction of the following two sentences:

$$p_1 \wedge \dots \wedge p_m \wedge \neg p'_{m+1} \wedge \dots \wedge \neg p'_n \supset h_1 \vee \dots \vee h_k, \quad (3)$$

$$p'_1 \wedge \dots \wedge p'_m \wedge \neg p'_{m+1} \wedge \dots \wedge \neg p'_n \supset h'_1 \vee \dots \vee h'_k. \quad (4)$$

Notice that if $m = n = 0$, then the left sides of the implications in (3) and (4) are considered to be *true*, and if $k = 0$, then the right sides of the implications in (3) and (4) are considered to be *false*.

3 The problem

As we mentioned above, one possible use of the notion of strongly equivalent logic programs is in program simplification. For instance, given a logic program, for each rule r in it, we may ask whether it can be deleted without knowing what other rules are in P , i.e. whether $\{r\}$ is strongly equivalent to the empty set. Or we may ask whether a rule r in P can be deleted if one knows that another rule r' is already in P , i.e. whether $\{r, r'\}$ is strongly equivalent to $\{r'\}$. In general, we may ask the following k - m - n question: Is $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ strongly equivalent to $\{r_1, \dots, r_k, v_1, \dots, v_n\}$? Thus our theorem discovery task is to come up, for a given k - m - n problem, a computationally effective condition that holds if and only if the answer to the k - m - n question is positive.

Now suppose we have such a condition C , and suppose that when $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ is strongly equivalent to $\{r_1, \dots, r_k, v_1, \dots, v_n\}$, it is better to replace $\{u_1, \dots, u_m\}$ by $\{v_1, \dots, v_n\}$ in the presence of $r_1, \dots, r_k$ for the purpose of, say computing the answer sets of a program. One way to use this result to simplify a given program P is to first choose k rules in P , and for any other m rules in it, try to find n rules so that the condition C holds, and then replace the m rules in P by the simpler n rules.

However, even if checking whether C holds would take a negligible constant time, using the above procedure to simplify a given logic program will be practical only when k, m, n are all very small or when k is almost the same as the number of the rules in the given program, and m and n are very small. Thus it seems to us that it is worthwhile to solve the k - m - n problem only when k, m, n are small. In particular, in this paper, we shall concentrate on the 0-1-0 problem (whether a rule can always be deleted), the 0-1-1 problem (whether a rule can always be replaced by another one), the 1-1-0 problem (in the presence of a rule, whether another rule can be deleted), the 2-1-0 problem (in the presence of two rules, whether a rule can always be deleted), and the 0-2-1 problem (if a pair of rules can be replaced by a single rule).

An example of theorems that we want to discover about these problems is as follows:

For any rule r , $\{r\}$ is strongly equivalent to the empty set $\emptyset$ iff $(Hd_r \cup Ng_r) \cap Ps_r \neq \emptyset$. (*)

4 Some General Theorems

In this section, we prove some general theorems that will help us verify whether an assertion like (*) above is true.

Let L be a propositional language, i.e. a set of atoms. From L , construct a first-order language F_L with equality, two unary predicates H_1 and H_2 , three unary predicates Hd_r , Ps_r , and Ng_r for each logic program rule r in L (we assume that each rule in L has a unique name), and three unary predicates X_i , Y_i , and Z_i for each positive number i .

Notice that we have used Hd_r , Ps_r , and Ng_r to denote sets of atoms previously, but now we overload them as unary predicates. Naturally, the intended interpretations of these unary predicates are their respective sets.

Definition 1 Given a set L of atoms, an intended model of F_L is one whose domain is L , and for each rule r in L , the unary predicates Ps_r , Hd_r , and Ng_r are interpreted by their corresponding sets of atoms, Ps_r , Hd_r , and Ng_r , respectively.

Conditions on rules in L , such as $Ps_r \cap Ng_r \neq \emptyset$, will be expressed by special sentences called *properties* in F_L .

Definition 2 A sentence of F_L is a property about n rules if it is constructed from equality and predicates X_i , Y_i , and Z_i , $1 \leq i \leq n$. A property Φ about n rules is true (holds) on a sequence $P = [r_1, \dots, r_n]$ of n rules if $\Phi[P]$ is true in an intended model of F_L , where $\Phi[P]$ is obtained from Φ by replacing each X_i by Hd_{r_i} , Y_i by Ps_{r_i} , and Z_i by Ng_{r_i} .

Notice that since $\Phi[P]$ does not mention predicates X_i , Y_i , Z_i , H_1 , and H_2 , if it is true in one intended model, then it is true in all intended models.

As we have mentioned above, we are interested in capturing the strong equivalence between two programs by a computationally effective condition. More specifically, for some small k , m , and n , we are interested in finding a property Φ about $k+m+n$ rules such that for any sequence of $k+m+n$ rules, $P = [r_1, \dots, r_k, u_1, \dots, u_m, v_1, \dots, v_n]$,

$\{r_1, \dots, r_k, u_1, \dots, u_m\}$ and $\{r_1, \dots, r_k, v_1, \dots, v_n\}$ are strongly equivalent if and only if Φ is true on P . (5)

We shall now prove some general theorems that can help us verify the above assertion for a class of formulas Φ .

First of all, Theorem 1 can be reformulated in F_L as follows by reading $H_1(p)$ as “ p holds”, and $H_2(p)$ as “ p' holds”:

Theorem 2 P_1 and P_2 are strongly equivalent in L iff the following sentence

$$\forall x(H_1(x) \supset H_2(x)) \supset [\bigwedge_{r \in P_1} \gamma(r) \equiv \bigwedge_{r \in P_2} \gamma(r)] \quad (6)$$

is true in all intended models of F_L , where $\gamma(r)$ is the conjunction of the following two sentences:

$$[\forall x(Ps_r(x) \supset H_1(x)) \wedge \forall x(Ng_r(x) \supset \neg H_2(x))] \supset \exists x(Hd_r(x) \wedge H_1(x)), \quad (7)$$

$$[\forall x(Ps_r(x) \supset H_2(x)) \wedge \forall x(Ng_r(x) \supset \neg H_2(x))] \supset \exists x(Hd_r(x) \wedge H_2(x)). \quad (8)$$

In first order logic, if a prenex formula of the form $\exists \vec{x} \forall \vec{y} B$ is satisfiable, then it is satisfiable in a structure with n elements, where n is the length of $\vec{x}$ if it is non-empty, and 1 when $\vec{x}$ is empty. We can prove a similar result for our first-order languages and their intended models here.

From this and Theorem 2, we can show the following theorem which will enable us to automate the verification of the “if” part of (5) when the property Φ is in the prenex format.

Theorem 3 Without loss of generality, suppose $m \geq n$. If Φ is a property about $k+m+n$ rules of the form $\exists \vec{x} \forall \vec{y} Q$, where $\vec{x}$ is a tuple of w variables, and Q a formula that does not have any quantifiers, then the following two assertions are equivalent:

- (a) For any sequence of $k+m+n$ rules, $P = [r_1, \dots, r_k, u_1, \dots, u_m, v_1, \dots, v_n]$, if Φ is true on P , then $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ is strongly equivalent to $\{r_1, \dots, r_k, v_1, \dots, v_n\}$.
- (b)(b.1) If $n > 0$, then for any sequence $P = [r_1, \dots, r_k, u_1, \dots, u_m, v_1, \dots, v_n]$ of rules with at most $w + 2(k + m)$ atoms, if Φ is true on P , then $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ is strongly equivalent to $\{r_1, \dots, r_k, v_1, \dots, v_n\}$.
- (b.2) If $n = 0$, then for any sequence $P = [r_1, \dots, r_k, u_1, \dots, u_m]$ of rules with at most K atoms, if Φ is true on P , then $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ is strongly equivalent to $\{r_1, \dots, r_k\}$, where K is $w + 2k$ if $w + 2k > 0$, and $K = 1$ otherwise.

The “only if” part of (5) can often be proved with the help of the following theorem.

Theorem 4 Let L_1 and L_2 be two languages, and f a function from L_1 to L_2 . If P_1 and P_2 are two programs in L_1 that are strongly equivalent, then $f(P_1)$ and $f(P_2)$ are two programs in L_2 that are also strongly equivalent. Here $f(P)$ is obtained from P by replacing each atom p in it by $f(p)$.

Proof: By Theorem 1 and the fact that in propositional logic, if φ is a tautology, and f a function from L_1 to L_2 , then $f(\varphi)$ is also a tautology, where $f(\varphi)$ is the formula obtained from φ by replacing each atom p in it by $f(p)$. ■

For an example of using the theorems in this section for proving assertions of the form (5), see Section 5.1.

5 Computer aided theorem discovery

Given a k - m - n problem, our strategy for discovering theorems about it is as follows:

1. Choose a small language L ;
2. Generate all possible triples

$$(\{r_1, \dots, r_k\}, \{u_1, \dots, u_m\}, \{v_1, \dots, v_n\}) \quad (9)$$

of sets of rules in L such that $\{r_1, \dots, r_k, u_1, \dots, u_m\}$ is strongly equivalent to $\{r_1, \dots, r_k, v_1, \dots, v_n\}$ in L ;

3. Formulate a conjecture on the k - m - n problem that holds in the language L , i.e. a condition that is true for a triple of the form (9) iff it is generated in Step 2;
4. Verify the correctness of this conjecture in the general case.

This process may have to be iterated. For instance, a conjecture came up at Step 3 may fail to generalize in Step 4, so the whole process has to be repeated. Or we may start with a language L with, say three atoms but have to increase it to five or six atoms later on.

Ideally, we would like this process to be automatic. However, it is difficult to automate Steps 3 and 4 - the number of possible patterns that we need to examine in order to come up with a good conjecture in Step 3 is huge, and we do not have a general theorem that enables us to automate the verification part in Step 4: while Theorem 3 enables us to automate

the proof of the sufficient part of the assertion (5) for a class of formulas Φ , we do not have a similar result for the necessary part. Theorem 4 helps, but it is not fully automatic yet. Nonetheless, computers play a crucial role in all steps, and in the following we report some of the theorems discovered using the above procedure.

5.1 The 0-1-0 problem

This problem asks if a given rule is strongly equivalent to the empty set, thus can always be deleted from any program. We have the following experimental result:

Lemma 1 *If a rule r mentions at most three distinct atoms, then $\{r\}$ is strongly equivalent to $\emptyset$ iff $(Hd_r \cup Ng_r) \cap Ps_r \neq \emptyset$.*

Using Theorem 4, we can show the following result:

Lemma 2 *If there is a rule r of the form (1) such that $\{r\}$ is strongly equivalent to $\emptyset$ and $(Hd_r \cup Ng_r) \cap Ps_r \neq \emptyset$ is not true, then there is such a rule that mentions at most three atoms.*

Proof: Suppose $\{r\}$ is strongly equivalent to $\emptyset$, $Hd_r \cap Ps_r = \emptyset$, and $Ps_r \cap Ng_r = \emptyset$. Suppose L is the set of atoms in r , and a, b, c are three new atoms. Let

$$f(p) = \begin{cases} a & p \in Hd_r \\ b & p \in Ps_r \\ c & \text{otherwise} \end{cases}$$

By Theorem 4, $\{f(r)\}$ is also strongly equivalent to $\emptyset$. By the construction of f , we also have $Hd_{f(r)} \cap Ps_{f(r)} = \emptyset$, and $Ps_{f(r)} \cap Ng_{f(r)} = \emptyset$, and that $f(r)$ mentions at most three distinct atoms. ■

Theorem 5 (The 0-1-0 problem) *Lemma 1 holds in the general case, i.e. without any restriction on the number of atoms in r .*

Proof: We notice that the condition in Lemma 1, $(Hd_r \cup Ng_r) \cap Ps_r \neq \emptyset$, is equivalent to the following property

$$\exists x.(X_1(x) \vee Z_1(x)) \wedge Y_1(x)$$

being true on $[r]$. Thus the “if” part follows from Theorem 3 and Lemma 1. The “only if” part follows from Lemma 1 and Lemma 2. ■

The “if” part of the theorem is already well-known, first proved by Osorio *et al.* [2001]. To the best of our knowledge, the “only if” part is new.

We notice here that there is no need to consider the 0- n -0 problem for $n > 1$, because for any n , $\{r_1, \dots, r_n\}$ is strongly equivalent to $\emptyset$ iff for each $1 \leq i \leq n$, $\{r_i\}$ is strongly equivalent to $\emptyset$.

5.2 The 1-1-0 and the 0-1-1 problems

The 1-1-0 problem asks if a rule can always be deleted in the presence of another rule, and the 0-1-1 problem asks if a rule can always be replaced by another one. We first solve the 1-1-0 problem, and the solution to the 0-1-1 problem will come as a corollary.

We have the following experimental result for the 1-1-0 problem:

Lemma 3 *For any two rules r_1 and r_2 that mentions at most three atoms, $\{r_1, r_2\}$ and $\{r_1\}$ are strongly equivalent iff one of the following two conditions is true:*

1. $\{r_2\}$ is strongly equivalent to $\emptyset$.
2. $Ps_{r_1} \subseteq Ps_{r_2}$, $Ng_{r_1} \subseteq Ng_{r_2}$, and $Hd_{r_1} \subseteq Hd_{r_2} \cup Ng_{r_2}$.

Lemma 4 *If there are two rules r_1 and r_2 such that $\{r_1, r_2\}$ and $\{r_2\}$ are strongly equivalent, but none of the two conditions in Lemma 3 hold, then there are two such rules that mention at most three atoms.*

Theorem 6 (The 1-1-0 problem) *Lemma 3 holds in the general case, without any restriction on the number of atoms in r_1 and r_2 .*

Proof: The condition in Lemma 3 is equivalent to the following property

$$\begin{aligned} & [\exists x.(X_2(x) \vee Z_2(x)) \wedge Y_2(x)] \vee \\ & \{[\forall x.Y_1(x) \supset Y_2(x)] \wedge [\forall x.Z_1(x) \supset Z_2(x)] \wedge \\ & [\forall x.X_1(x) \supset (X_2(x) \vee Z_2(x))]\} \end{aligned}$$

being true on $[r_1, r_2]$. Thus the “if” part follows from Theorem 3 and Lemma 3, by noticing that the above property can be written as $\exists x \forall \vec{y}. Q$ as required by Theorem 3. The “only if” part follows from Lemma 3 and Lemma 4. ■

Thus if a rule r_2 cannot be deleted on its own but can be deleted in the presence of another rule r_1 , then it must be the case that r_2 is redundant given r_1 : if the body of r_2 is satisfied, then the body of r_1 is satisfied as well; furthermore, r_2 can entail no more than what can be entailed by r_1 ($Hd_{r_1} \subseteq Hd_{r_2} \cup Ng_{r_2}$).

Osorio *et al.* [2001] proved that $\{r_1, r_2\}$ and $\{r_1\}$ are strongly equivalent if either $Ps_{r_1} \cup Ng_{r_1} = \emptyset$ and $Hd_{r_1} \subseteq Ng_{r_2}$ or $Ps_{r_1} \subseteq Ps_{r_2}$, $Ng_{r_1} \subseteq Ng_{r_2}$, and $Hd_{r_1} \subseteq Hd_{r_2}$. More recently, Eiter *et al.* [2004] showed that $\{r_1, r_2\}$ and $\{r_1\}$ are strongly equivalent if r_1 *s-implies* r_2 [Wang and Zhou, 2005], i.e. if there exists a set $A \subseteq Ng_{r_2}$ such that $Hd_{r_1} \subseteq Hd_{r_2} \cup A$, $Ng_{r_1} \subseteq Ng_{r_2} \setminus A$, and $Ps_{r_1} \subseteq Ps_{r_2}$.

As one can see, these are all special cases of the “if” part of Theorem 6. Our result is actually more general. For instance, these special cases do not apply to $\{(c \leftarrow b, \text{not } c), (\leftarrow b, \text{not } c)\}$ and $\{c \leftarrow b, \text{not } c\}$, but one can easily show that these two sets are strongly equivalent using our theorem.

From our solution to the 1-1-0 problem, we can derive a solution to the 0-1-1 problem.

Theorem 7 (The 0-1-1 problem) *For any two rules r_1 and r_2 , $\{r_1\}$ and $\{r_2\}$ are strongly equivalent iff one of the following two conditions is true:*

1. $\{r_1\}$ and $\{r_2\}$ are both strongly equivalent to $\emptyset$.
2. $Ps_{r_1} = Ps_{r_2}$, $Ng_{r_1} = Ng_{r_2}$, and $Hd_{r_1} \cup Ng_{r_1} = Hd_{r_2} \cup Ng_{r_2}$.

Proof: By Theorem 1, it is easy to see that $\{r_1\}$ and $\{r_2\}$ are strongly equivalent iff $\{r_1, r_2\}$ and $\{r_1\}$ are strongly equivalent and $\{r_1, r_2\}$ and $\{r_2\}$ are strongly equivalent. ■

Thus two rules r_1 and r_2 can always be interchanged if either both of them can be deleted (strongly equivalent to the

empty set) or they have the same body, and the same consequences when the body is true. For instance, $\{a \leftarrow B, \text{not } a\}$ and $\{\leftarrow B, \text{not } a\}$ are strongly equivalent no matter what B is, because the two rules have the same body, and when the body is true, the same consequence - a contradiction. As another example, $\{a; b \leftarrow \text{not } a\}$ and $\{b \leftarrow \text{not } a\}$ are strongly equivalent because the two rules have the same body, and, when the body is true, the same consequence, b .

5.3 The 2-1-0, 0-2-1, and 0-2-2 problems

The 2-1-0 problem asks if a rule can be deleted in the presence of another two rules, the 0-2-1 problem asks if two rules can be replaced by a single rule, and the 0-2-2 problem asks if two rules can be replaced by another two rules. Similar to the previous subsection, the solution to the 0-2-1 and 0-2-2 problems will follow from a solution to the 2-1-0 problem.

We have the following experimental result for the 2-1-0 problem:

Lemma 5 *For any three rules r_1, r_2 and r_3 that mentions at most six atoms, $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$ are strongly equivalent iff one of the following four conditions is true:*

1. $\{r_3\}$ is strongly equivalent to $\emptyset$.
2. $\{r_1, r_3\}$ is strongly equivalent to $\{r_1\}$.
3. $\{r_2, r_3\}$ is strongly equivalent to $\{r_2\}$.
4. There is an atom p such that:
 - 4.1 $p \in (Ps_{r_1} \cup Ps_{r_2}) \cap (Hd_{r_1} \cup Hd_{r_2} \cup Ng_{r_1} \cup Ng_{r_2})$
 - 4.2 $Hd_{r_i} \setminus \{p\} \subseteq Hd_{r_3} \cup Ng_{r_3}$ and $Ps_{r_i} \setminus \{p\} \subseteq Ps_{r_3}$ and $Ng_{r_i} \setminus \{p\} \subseteq Ng_{r_3}$, where $i = 1, 2$
 - 4.3 If $p \in Ps_{r_1} \cap Ng_{r_2}$, then $Hd_{r_1} \cap Hd_{r_3} = \emptyset$
 - 4.4 If $p \in Ps_{r_2} \cap Ng_{r_1}$, then $Hd_{r_2} \cap Hd_{r_3} = \emptyset$

Notice that in principle, given a language L , every subset of L can be the Hd , Ps , or Ng of a rule. Thus when the size of L is six, there are in principle $(2^6)^3 - 1 = 262,143$ possible rules, and $262,143^3$ triples of them. Thus at first glance, it seems that verifying Lemma 5 experimentally using the currently available computers would be impossible. However, we can cut the numbers down significantly with the results that we already have proved. First, we only have to consider rules that do not have common elements in Hd , Ps , and Ng : if either Hd and Ps or Ps and Ng have a common element, then by Theorem 5, this rule can be deleted; if Hd and Ng have common elements, then according to Theorem 7, we obtain a strongly equivalent rule by deleting the common elements in Hd . Second, we do not have to consider isomorphic rules: if there is a one-to-one onto function from L to L that maps $\{r_1, r_2, r_3\}$ to $\{r'_1, r'_2, r'_3\}$, then these two sets of rules are essentially the same except for the names of atoms in them. By using a certain normal form for triples of rules that avoids these redundant cases, we ended up with roughly 120 million triples of rules to consider for testing Lemma 5, which took about 10 hours on a Solaris server consisting of 8 Sun Ultra-SPARC III 900Mhz CPUs with 8GB RAM.

The following lemma is the reason why we need to consider a language with six atoms for this problem.

Lemma 6 *If there are three rules r_1, r_2 and r_3 such that $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$ are strongly equivalent, but none of the four conditions in Lemma 5 hold, then there are three such rules that mention at most six atoms.*

Theorem 8 (The 2-1-0 problem) *Lemma 5 holds in the general case, without any restriction on the number of atoms in r_1, r_2, r_3 .*

The conditions in Lemma 5 (Theorem 8) are rather complex, and the reason why it is difficult to automate Step 3 of the procedure at the beginning of the section. These conditions capture all possible cases when r_3 is “subsumed” by r_1 and r_2 , and are difficult to describe concisely by words. We give some examples.

Consider the following three rules: $r_1: (a_2 \leftarrow a_1)$, $r_2: (a_3 \leftarrow \text{not } a_1)$, and $r_3: (a_3 \leftarrow \text{not } a_2)$. We have that $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$ are strongly equivalent because the condition (4) in Lemma 5 holds.

However, if we change r_3 into $r'_3: a_2 \leftarrow \text{not } a_3$, then $P_1 = \{r_1, r_2, r'_3\}$ and $P_2 = \{r_1, r_2\}$ are not strongly equivalent: one could check that condition (4.3) in Lemma 5 does not hold, and indeed, while $P_2 \cup \{a_1 \leftarrow a_2\}$ has a unique answer set $\{a_3\}$, $P_1 \cup \{a_1 \leftarrow a_2\}$ has two answer sets $\{a_3\}$ and $\{a_1, a_2\}$.

It is also easy to show by Theorem 8 that $a_3 \leftarrow \text{not } a_2$ is “subsumed” by $\{(a_1; a_2; a_3 \leftarrow), (a_2; a_3 \leftarrow a_1)\}$, and $a_2; a_3 \leftarrow$ is “subsumed” by $\{(a_2 \leftarrow a_1), (a_3 \leftarrow \text{not } a_1)\}$.

With the results that we have, the following theorem will yield a solution to the 0-2-1 problem.

Theorem 9 (the 0-2-1 problem) *For any three rules r_1, r_2 and r_3 , $\{r_1, r_2\}$ and $\{r_3\}$ are strongly equivalent iff the following three conditions are true:*

1. $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$ are strongly equivalent.
2. $\{r_1, r_3\}$ and $\{r_3\}$ are strongly equivalent.
3. $\{r_2, r_3\}$ and $\{r_3\}$ are strongly equivalent.

For example, $\{(a_2 \leftarrow a_1, \text{not } a_3), (a_1; a_2 \leftarrow \text{not } a_3)\}$ is strongly equivalent to $\{a_2 \leftarrow \text{not } a_3\}$. While $\{(\leftarrow a_2, a_3), (\leftarrow a_3, \text{not } a_2)\}$ is strongly equivalent to $\{\leftarrow a_3\}$, $\{(a_1 \leftarrow a_2, a_3), (a_1 \leftarrow a_3, \text{not } a_2)\}$ is not strongly equivalent to $\{a_1 \leftarrow a_3\}$. Similarly, we have the following theorem

Theorem 10 (the 0-2-2 problem) *For any four rules r_1, r_2, r_3, r_4 , $\{r_1, r_2\}$ and $\{r_3, r_4\}$ are strongly equivalent iff the following four conditions are true:*

1. $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$ are strongly equivalent.
2. $\{r_1, r_2, r_4\}$ and $\{r_1, r_2\}$ are strongly equivalent.
3. $\{r_3, r_4, r_1\}$ and $\{r_3, r_4\}$ are strongly equivalent.
4. $\{r_3, r_4, r_2\}$ and $\{r_3, r_4\}$ are strongly equivalent.

6 Program simplification

We have mentioned that one possible use of the notion of strongly equivalent logic programs is in simplifying logic programs: if P and Q are strongly equivalent, and that Q is “simpler” than P , we can then replace P in any program that contains it by Q .

Most answer set programming systems perform some program simplifications. However, only Smodels [Niemelä *et al.*, 2000] has a stand-alone front-end called *lparse* that can be used to ground and simplify a given logic program. It seems that *lparse* simplifies a grounded logic program by computing first its well-founded model. It does not, however, perform any program simplification using the notion of strong equivalence. For instance, *lparse*-1.0.13, the current version of *lparse*, did nothing to the following set of rules: $\{(a \leftarrow \text{not } b), (b \leftarrow \text{not } a), (a \leftarrow a)\}$. Nor does it replace the first rule in the following program $\{(a \leftarrow \text{not } a), (a \leftarrow \text{not } b), (b \leftarrow \text{not } a)\}$ by the constraint $\leftarrow \text{not } a$.

It is unlikely that anyone would be intentionally writing rules like $a \leftarrow a$ or $b \leftarrow a, \text{not } a$. But these type of rules can arise as a result of grounding some rules with variables. For instance, the following is a typical recursive rule used in logic programming encoding of the Hamiltonian Circuit problem [Niemelä, 1999; Marek and Truszczyński, 1999]:

$$\text{reached}(X) \leftarrow \text{arc}(Y, X), \text{hc}(Y, X), \text{reached}(Y).$$

When instantiated on a graph with cyclic arcs like $\text{arc}(a, a)$, this rule generates cyclic rules of the form $\text{reached}(X) \leftarrow \text{hc}(X, X), \text{reached}(X)$. Unless deleted explicitly, these rules will slow down many systems, especially those based on SAT.

It is thus useful to consider using the results that we have here for program simplification. Indeed, transformation rules such as deleting those that contain common elements in their heads and positive bodies have been proposed [Brass and Dix, 1999], and studied from the perspective of strong equivalence [Osorio *et al.*, 2001; Eiter *et al.*, 2004]. Our results add new such transformation rules. For instance, by Theorem 7, we can delete those elements in the head of a rule that also appear in the negation-as-failure part of the rule. Theorems 6, 8, and 9 can also be used to define some new transformation rules.

7 Concluding remarks and future work

Donald Knuth, in his Forward to [Petkovsek *et al.*, 1996], said

“Science is what we understand well enough to explain to a computer. Art is everything else we do. ...Science advances whenever an Art becomes a Science. And the state of the Art advances too, because people always leap into new territory once they have understood more about the old.”

We hope that with this work, we are one step closer to making discovering classes of strongly equivalent logic programs a Science.

We have mentioned that the methodology used in this paper is similar to that in [Lin, 2004]. In both cases, plausible conjectures are generated by testing them in domains of small sizes, and general theorems are proved to aid the verification of these conjectures in the general case. However, while plausible conjectures are generated automatically in [Lin, 2004], they are done manually here. While the verifications of most conjectures in [Lin, 2004] are done automatically as well, they are done only semi-automatically here. Overcoming these two weaknesses is the focus of our future

work. Specifically, we would like to make Step 3 of the procedure in Section 5 automatic, and prove a theorem similar to Theorem 3 to automate the proofs of the “only if” parts of theorems like Theorems 5 - 8, in the same way that Theorem 3 makes the proofs of the “if” parts of these theorems automatic. This way, we would be able to discover more interesting theorems in this area, and more easily!

References

- [Brass and Dix, 1999] S. Brass and J. Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *J. Log. Program.* 40(1), pages 1–46, 1999.
- [Eiter *et al.*, 2004] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. In *LPNMR*, pages 87–99, 2004.
- [Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [Lenat, 1979] D. B. Lenat. On automated scientific theory formation: A case study using the AM program. *Machine Intelligence* 9, pages 251–283, 1979. Jean Hayes, Donald Michie, and L. I. Mikulich, eds.
- [Lifschitz *et al.*, 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
- [Lin, 2002] F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. KR’02*, pages 170–176, 2002.
- [Lin, 2004] F. Lin. Discovering state invariants. In *Proc. KR’04*, pages 536–544, 2004.
- [Marek and Truszczyński, 1999] V. W. Marek and M. Truszczyński. Stable logic programming - an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*. K.R. Apt, V.W. Marek, M. Truszczyński, D.S. Warren, eds, Springer-Verlag, 1999.
- [Niemelä *et al.*, 2000] I. Niemelä, P. Simons, and T. Syrjänen. Smodels: a system for answer set programming. In *Proc NMR-2000*. (CoRR: arXiv:cs.AI/0003033) <http://www.tcs.hut.fi/Software/smodels/>.
- [Niemelä, 1999] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. and AI*, 25(3-4):241–273, 1999.
- [Osorio *et al.*, 2001] M. Osorio, J. A. Navarro, and J. Arrazola. Equivalence in answer set programming. In *LOPSTR 2001*, pages 57–75, 2001.
- [Petkovsek *et al.*, 1996] M. Petkovsek, H. S. Wilf, and D. Zeilberger. *A = B*. Wellesley, Mass. : A K Peters, 1996.
- [Wang and Zhou, 2005] K. Wang and L. Zhou. Comparisons and computation of well-founded semantics for disjunctive logic programs. *ACM Transactions on Computational Logic*, 2005. To appear.

Tractable Reasoning with Incomplete First-Order Knowledge in Dynamic Systems with Context-Dependent Actions

Yongmei Liu and Hector J. Levesque

Department of Computer Science

University of Toronto

Toronto, ON, Canada M5S 3G4

{yliu, hector}@cs.toronto.edu

Abstract

A basic reasoning problem in dynamic systems is the projection problem: determine if a formula holds after a sequence of actions has been performed. In this paper, we propose a tractable¹ solution to the projection problem in the presence of incomplete first-order knowledge and context-dependent actions. Our solution is based on a type of progression, that is, we progress the initial knowledge base (KB) wrt the action sequence and answer the query against the resulting KB. The form of reasoning we propose is always logically sound and is also logically complete when the query is in a certain normal form and the agent has complete knowledge about the context of any context-dependent actions.

1 Introduction

In the area of Knowledge Representation and Reasoning, there is a well-known tradeoff between the expressiveness of the representation language and the computational tractability of the associated reasoning task. At one extreme, we have databases for which queries can be efficiently evaluated. But databases are too limited for many AI applications because they require complete knowledge about the domains. Levesque [1998] proposes a generalization of a database called a *proper KB*, which allows a limited form of incomplete knowledge, equivalent to a (possibly infinite) consistent set of ground literals. Since the deduction problem for proper KBs is undecidable, Levesque proposes an evaluation-based reasoning procedure called V that is logically sound and, when the query is in a certain *normal form* called $\mathcal{NF}$, also logically complete. Moreover, later Liu and Levesque [2003] show that despite the incomplete knowledge, database techniques can be used to implement V efficiently.

In this paper, we apply the procedure V to reasoning in *dynamic systems* where the state of the world changes as a result of the actions of agents. For such applications, a basic reasoning problem is the so-called *projection problem*: given an action theory that specifies the preconditions and effects of actions, and an initial KB, determine whether or not a formula

holds after a sequence of actions is performed. Two settings where this problem arises naturally are for planning and for high-level program execution [Levesque *et al.*, 1997]. A prerequisite to planning is the ability to determine if a goal is satisfied after a sequence of actions. To execute a high level robotic program such as “while there is a block on the table, pick up a block and put it away”, one needs to determine after various sequences of actions whether there is still a block on the table.

In practice, there are two ways to deal with projection: we can *progress* the initial KB wrt the action sequence and answer the query against the resulting KB; or we can *regress* the query wrt the action sequence and answer the resulting query against the initial KB. Progression has at least two advantages: First, it avoids a duplication of effort when multiple queries need to be answered wrt the same action sequence, and especially when that sequence is long. Second, in a robotics setting, a robot can use its “mental idle time” to compute a progression while it is busy performing physical actions. Projection via progression has three main computational requirements: the new KB must be efficiently computed, its size should be at most linear in the size of the initial KB (to allow for iterated progression), and it must be possible to answer the query efficiently from the new KB. Lin and Reiter [1997] give a formal study of progression. They show that progression is not always first-order definable, and identify a few important cases where progression is first-order definable and computationally tractable. However, the third requirement is not addressed in their paper.

In this paper, we propose a tractable, sound, and sometimes complete solution to the projection problem in the presence of incomplete first-order knowledge and context-dependent actions. We restrict our attention to actions with only “local” effects, and where incomplete knowledge is in the form of a proper KB. We define a version of progression where a proper KB remains proper afterward, and where applying V to the progressed KB and the query returns the same value as applying V to the initial KB and a regressed query. We prove that when the query is in $\mathcal{NF}$ and the initial KB has complete knowledge about the context of any context-dependent actions, our solution is logically complete. It is also logically complete when the query is in $\mathcal{NF}$ and there are sensing actions that provide dynamic information about the context of the relevant context-dependent actions.

¹By “tractable” we mean “solvable in polynomial time”.

2 Preliminaries

In this section, we review proper KBs, V , and $\mathcal{NF}$. Also, we briefly review the situation calculus, and formally define local-effect action theories and regression for them.

2.1 Proper KBs, V , and the normal form $\mathcal{NF}$

We use a standard first-order logical language $\mathcal{L}$ with equality, a countably infinite set of constants $\mathcal{C} = \{c_1, c_2, \dots\}$, and no other function symbols. We restrict our attention to *standard interpretations*, where equality is identity, and there is a bijection between the set of constants and the domain of discourse. This restriction can be captured by a set of axioms $\mathcal{E}$, consisting of the axioms of equality and the set of formulas $\{c_i \neq c_j \mid i \neq j\}$. Since we treat equality separately, when we say “predicate”, “atom” or “literal”, we exclude equality.

We use ρ to range over atoms whose arguments are distinct variables. We use e to range over *ewffs*, that is, quantifier-free formulas with only equalities (i.e. no predicates). We use $\forall\phi$ to denote the universal closure of ϕ . We write ϕ_c^x to denote ϕ with all free occurrences of x replaced by constant c . We write $\Sigma \models_{\mathcal{E}} \phi$ to denote $\mathcal{E} \cup \Sigma \models \phi$.

Definition 1 A KB Σ is *proper* if $\mathcal{E} \cup \Sigma$ is consistent and Σ is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$.

It is not hard to see that the problem of determining whether a sentence is logically entailed by a proper KB is undecidable, since when the KB is empty, this reduces to classical validity. Levesque [1998] proposes an evaluation-based reasoning procedure called V instead. Given a proper KB and a query, V returns one of three values 0 (known false), 1 (known true), or $\frac{1}{2}$ (unknown) as follows:

1. $V[\Sigma, P(\vec{c})] = \begin{cases} 1 & \text{if there is a } \forall(e(\vec{x}) \supset P(\vec{x})) \\ & \text{in } \Sigma \text{ such that } V[\Sigma, e(\vec{c})] = 1 \\ 0 & \text{if there is a } \forall(e(\vec{x}) \supset \neg P(\vec{x})) \\ & \text{in } \Sigma \text{ such that } V[\Sigma, e(\vec{c})] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$
2. $V[\Sigma, c = c'] = 1$ if c is identical to c' , and 0 otherwise;
3. $V[\Sigma, \neg\phi] = 1 - V[\Sigma, \phi]$;
4. $V[\Sigma, \phi \vee \psi] = \max\{V[\Sigma, \phi], V[\Sigma, \psi]\}$;
5. $V[\Sigma, \exists x\phi] = \max_{c \in H^+} V[\Sigma, \phi_c^x]$, where H^+ is the union of the constants in Σ or ϕ , and an extra constant.

This V procedure is logically sound and, when the query is in a certain normal form called $\mathcal{NF}$, also logically complete:

Theorem 1 (Levesque, 1998) *Let Σ be proper. Then*

1. *for every $\phi \in \mathcal{L}$, if $V[\Sigma, \phi] = 1$ then $\Sigma \models_{\mathcal{E}} \phi$; and if $V[\Sigma, \phi] = 0$ then $\Sigma \models_{\mathcal{E}} \neg\phi$.*
2. *for every $\phi \in \mathcal{NF}$, $V[\Sigma, \phi] = 1$ iff $\Sigma \models_{\mathcal{E}} \phi$; and $V[\Sigma, \phi] = 0$ iff $\Sigma \models_{\mathcal{E}} \neg\phi$.*

For the interested readers, the following is the definition of $\mathcal{NF}$ from [Levesque, 1998]:

Definition 2 A set Γ of sentences is logically separable iff for every consistent set of ground literals L , if $L \cup \Gamma$ has no standard model, then $L \cup \{\phi\}$ is inconsistent for some $\phi \in \Gamma$.

Definition 3 The normal form $\mathcal{NF}$ is the least set such that

1. if ϕ is a ground atom or ewff, then $\phi \in \mathcal{NF}$;
2. if $\phi \in \mathcal{NF}$, then $\neg\phi \in \mathcal{NF}$;
3. if $\phi_1, \dots, \phi_n \in \mathcal{NF}$, and $\{\phi_1, \dots, \phi_n\}$ is logically separable, then $\bigwedge \phi_i \in \mathcal{NF}$;
4. if $\Gamma \subseteq \mathcal{NF}$, Γ is logically separable, and for some ϕ , $\Gamma = \{\phi_c^x \mid c \in \mathcal{C}\}$, then $\forall x\phi \in \mathcal{NF}$.

The intuition behind $\mathcal{NF}$ is that different parts of a formula must be logically independent. A simple example of a formula not in $\mathcal{NF}$ is $(p \vee \neg p)$, where p is atomic. In the propositional case, a CNF formula is in $\mathcal{NF}$ if its clauses are non-tautologous and closed under resolution.

Liu and Levesque [2003] show that V can be implemented efficiently using database techniques (projections, joins, etc). Here we present a cleaner variant of this result.

Let $\mathcal{L}^k$ denote the set of formulas from $\mathcal{L}$ that use at most k different variables. Let $R = \{\vec{c}_1, \dots, \vec{c}_m\}$ be a finite set of n -tuples. We use $\vec{x} \in R$ to denote $\vec{x} = \vec{c}_1 \vee \dots \vee \vec{x} = \vec{c}_m$.

Definition 4 Let L be P or $\neg P$ for some predicate P . The *ewff defining L* in a proper Σ , denoted by ξ_L , is the disjunction of all e such that $\forall(e(\vec{x}) \supset L(\vec{x})) \in \Sigma$. We can write ξ_L in the form of $\vec{x} \in I_L \vee e_L \wedge \vec{x} \notin O_L$ so that I_L and O_L are finite relations with as many tuples as possible, and e_L is an ewff. The *e -size* of Σ is the maximum size of an e_L in Σ .

Then a corollary to Theorem 4.8 in [Liu and Levesque, 2003]:

Corollary 2 *Let Σ be proper, and let $\phi \in \mathcal{L}^k$. Then $V[\Sigma, \phi]$ can be computed in time $O(lmn^k)$, where l is the size of ϕ , m is the e -size of Σ , and n is the size of Σ .*

Although the time complexity scales exponentially with k , this is typical even of queries over ordinary databases, and so is perhaps as good as can be expected.

2.2 Situation calculus

Our account of action and change is formulated in the language of the situation calculus [McCarthy and Hayes, 1969; Reiter, 2001]. We will not go over the language here except to note the following components: there are three disjoint sorts for actions, situations, and objects; there is a special constant S_0 denoting the *initial situation*, namely the one in which no actions have yet occurred; there is a distinguished binary function $do(a, s)$ denoting the successor situation to s resulting from performing action a ; relations whose truth values vary from situation to situation, are called (relational) *fluents*, and are denoted by predicate symbols taking a situation term as their last argument; and there is a special predicate $Poss(a, s)$ stating that action a is executable in situation s .

We relate the language of the situation calculus to $\mathcal{L}$ as follows: There is a set of constants of sort object which are constants of $\mathcal{L}$. The situation-independent predicates and relational fluents are predicates from $\mathcal{L}$. That is, if $P(\vec{x})$ is a situation-independent predicate, and $F(\vec{x}, s)$ is a relational fluent, then $P(\vec{x})$ and $F(\vec{x})$ are predicates from $\mathcal{L}$.

We extend the language $\mathcal{L}$ to $\mathcal{L}^+$ by allowing equalities involving action functions. Let $\phi \in \mathcal{L}^+$, and let τ be a situation term. We use $\phi[\tau]$ to denote the situation calculus formula obtained from ϕ by taking τ as the situation arguments of all fluents mentioned by ϕ . We use α to range over ground actions, and we use δ to range over sequences of ground actions.

Let $\delta = \langle \alpha_1, \dots, \alpha_n \rangle$. We use $do(\delta, S_0)$ to denote the end situation of δ , that is, $do(\alpha_n, do(\alpha_{n-1}, \dots do(\alpha_1, S_0) \dots))$.

A particular domain of application will be specified by a basic action theory of the following form:²

$$\mathcal{D} = \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}, \text{ where}$$

1. $\mathcal{D}_{ap}$ is a set of action precondition axioms, one for each action function A , with form $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x})[s]$,³ where $\Pi_A(\vec{x}) \in \mathcal{L}$.
2. $\mathcal{D}_{ss}$ is a set of successor state axioms (SSAs), one for each fluent, of the form $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a)[s]$, where $\Phi_F(\vec{x}, a) \in \mathcal{L}^+$. Usually, $\Phi_F(\vec{x}, a)$ has the form

$$\gamma_F^+(\vec{x}, a) \vee (F(\vec{x}) \wedge \neg \gamma_F^-(\vec{x}, a)).$$

SSAs take the place of the so-called effect axioms, and provide a solution to the frame problem.

3. $\mathcal{D}_{una}$ is the set of unique names axioms for actions:

$$A(\vec{x}) \neq A'(\vec{y}), \text{ and } A(\vec{x}) = A(\vec{y}) \supset \vec{x} = \vec{y},$$

where A and A' are distinct action functions.

4. $\mathcal{D}_{S_0}$ is of the form $\{\phi[S_0] \mid \phi \in \Sigma_0\}$, where $\Sigma_0 \subseteq \mathcal{L}$. Σ_0 is called the initial KB.

In this setting, the projection task can be defined as follows: determine if $\mathcal{D} \models_{\varepsilon} \phi[do(\delta, S_0)]$, where $\phi \in \mathcal{L}$, and δ is a sequence of ground actions.

As a running example, we will use a simple blocks world.⁴ We use a single action, $move(x, y, z)$, moving a block x from block y to block z (treating the table as just another block). We use three fluents: $clear(x, s)$, block x has no blocks on top of it; $on(x, y, s)$, block x is on block y ; $eh(x, s)$, the height of block x is even. We have the following action precondition axiom and successor state axioms:

$$\begin{aligned} Poss(move(x, y, z), s) &\equiv clear(x) \wedge on(x, y) \wedge clear(z). \\ clear(x, do(a, s)) &\equiv (\exists y, z) a = move(y, x, z) \vee \\ &\quad clear(x, s) \wedge \neg(\exists y, z) a = move(y, z, x); \\ on(x, y, do(a, s)) &\equiv (\exists z) a = move(x, z, y) \vee \\ &\quad on(x, y, s) \wedge \neg(\exists z) a = move(x, y, z); \\ eh(x, do(a, s)) &\equiv (\exists y, z)[a = move(x, y, z) \wedge \neg eh(z, s)] \vee \\ &\quad eh(x, s) \wedge \neg(\exists y, z)[a = move(x, y, z) \wedge eh(z, s)]. \end{aligned}$$

2.3 Local effect action theories and regression

Actions in many dynamic domains have only *local effects* in the sense that if an action $A(\vec{c})$ changes the truth value of an atom $F(\vec{d})$, then $\vec{d}$ is contained in $\vec{c}$. This contrasts with actions having *universal effects* such as exploding a bomb, which kills all those near it. We can define this as follows:

Definition 5 A successor state axiom is *local-effect* if both $\gamma_F^+(\vec{x}, a)$ and $\gamma_F^-(\vec{x}, a)$ are disjunctions of formulas of the form $\exists \vec{z}[a = A(\vec{y}) \wedge \phi(\vec{y})]$, where A is an action function, $\vec{y}$ contains $\vec{x}$, $\vec{z}$ is the remaining variables of $\vec{y}$, and ϕ (called a *context formula*) is a quantifier-free formula from $\mathcal{L}$. An action theory is local-effect if each SSA is local-effect.

²We use slightly different notation from that in [Reiter, 2001].

³We omit the leading universal quantifiers.

⁴To justify the concerns for the tractability of reasoning, the reader should imagine there being a very large number of blocks.

Our blocks world example above is clearly local-effect.

The notion of a successor state axiom being local-effect is a generalization of that of being strictly context-free defined by Lin and Reiter [1997]. An SSA is *strictly context-free* if $\gamma_F^+(\vec{x}, a)$ and $\gamma_F^-(\vec{x}, a)$ are disjunctions of formulas of the form $\exists \vec{z}[a = A(\vec{y})]$, where A , $\vec{y}$, and $\vec{z}$ are as above. For instance, the SSA for fluent *on* is strictly context-free, while that for fluent *eh* is not.

By using the unique names axioms, the instantiation of a local-effect SSA on a ground action can be significantly simplified. Suppose the SSA for F is local-effect. Let $\alpha = A(\vec{c})$ be a ground action, and let $*$ be $+$ or $-$. Then $\gamma_F^*(\vec{x}, \alpha)$ is equivalent to a formula of the following form:

$$\vec{x} = \vec{d}_1 \wedge \psi_1 \vee \dots \vee \vec{x} = \vec{d}_n \wedge \psi_n,$$

where $\vec{d}_i$ is a vector of constants contained in $\vec{c}$, and ψ_i is a sentence. We will use $\gamma_F^*(\alpha)(\vec{x})$ to denote the above formula, and we will write $(\vec{d}, \psi) \in \gamma_F^*(\alpha)$ to mean that $\vec{x} = \vec{d} \wedge \psi$ is one of the disjuncts. Also, we will use $\Phi_F(\alpha)(\vec{x})$ to denote $\gamma_F^+(\alpha)(\vec{x}) \vee (F(\vec{x}) \wedge \neg \gamma_F^-(\alpha)(\vec{x}))$. In the case of our blocks world, instances of the SSAs can be simplified as follows:

$$\begin{aligned} clear(x, do(move(c_1, c_2, c_3), s)) &\equiv x = c_2 \vee \\ &\quad clear(x, s) \wedge \neg(x = c_3). \\ on(x, y, do(move(c_1, c_2, c_3), s)) &\equiv x = c_1 \wedge y = c_3 \vee \\ &\quad on(x, y, s) \wedge \neg(x = c_1 \wedge y = c_2). \\ eh(x, do(move(c_1, c_2, c_3), s)) &\equiv x = c_1 \wedge \neg eh(c_3, s) \vee \\ &\quad eh(x, s) \wedge \neg(x = c_1 \wedge eh(c_3, s)). \end{aligned}$$

An important computational mechanism for reasoning about actions is regression [Reiter, 2001]. Here we define a one-step regression operator for local-effect action theories.

Definition 6 Let $\phi \in \mathcal{L}$. We use $\mathcal{R}_\alpha(\phi)$ to denote the formula obtained from ϕ by replacing each fluent atom $F(\vec{t})$ with $\Phi_F(\alpha)(\vec{t})$. We call $\mathcal{R}_\alpha(\phi)$ the *regression* of ϕ wrt α .

Note that $\mathcal{R}_\alpha(\phi)$ remains in $\mathcal{L}$. Let $\delta = \langle \alpha_1, \dots, \alpha_n \rangle$. We use $\mathcal{R}_\delta$ to denote $\mathcal{R}_{\alpha_1} \circ \dots \circ \mathcal{R}_{\alpha_n}$. We now state a simple form of the regression theorem [Reiter, 2001]. Recall that Σ_0 is the initial KB of $\mathcal{D}$.

Theorem 3 (The Regression Theorem)

For every $\phi \in \mathcal{L}$, $\mathcal{D} \models_{\varepsilon} \phi[do(\delta, S_0)]$ iff $\Sigma_0 \models_{\varepsilon} \mathcal{R}_\delta(\phi)$.

This theorem shows that regression is a sound and complete solution to the projection problem. In this paper, we prove all our results about progression by using regression as a bridge.

3 Progression of Proper KBs

In this section, we define a variant of classical progression, and show how to compute it for local-effect action theories.

First consider *classical progression*. Suppose we have a KB Σ . Let M be a possible state of Σ , that is, a model of Σ . Let α be a ground action. Then the *successor state* of M wrt α is the model M' such that for any ground fluent atom $F(\vec{c})$, $M' \models F(\vec{c})$ iff $M \models \Phi_F(\alpha)(\vec{c})$. A KB Σ' is a progression of Σ wrt α if the models of Σ' are exactly the successor states of models of Σ wrt α . A basic property of progression is: Suppose that Σ' is a progression of Σ wrt α . Then for every $\phi \in \mathcal{L}$, $\Sigma' \models_{\varepsilon} \phi$ iff $\Sigma \models_{\varepsilon} \mathcal{R}_\alpha(\phi)$. It is in this sense that we say classical progression preserves classical entailment.

It would be nice if the classical progression of a proper KB were proper, so that we could use it and V to solve the projection problem. However, this is unfortunately not the case even for very simple action theories. Consider the following example from [Petrick and Levesque, 2002]:

$$F(do(a, s)) \equiv a = A \wedge G(s) \vee F(s); \quad G(do(a, s)) \equiv G(s).$$

Then any progression of the empty KB (which is proper) wrt action A results in disjunctive information, $(F \vee \neg G)$, and hence is no longer proper. So what we will propose is a variant of classical progression where the progression of a proper KB does remain proper and the progression preserves V instead of preserving classical entailment.

Definition 7 Let Σ and Σ' be proper. We say that Σ' is a (weak) progression of Σ wrt a ground action α if for every $\phi \in \mathcal{L}$, $V[\Sigma', \phi] = V[\Sigma, \mathcal{R}_\alpha(\phi)]$.

We now show that for local-effect action theories, it is easy to compute a weak progression of a proper KB.

Definition 8 Let $\mathcal{D}$ be local-effect and Σ be proper. We define $\mathcal{P}_\alpha(\Sigma)$ as the set of the following sentences:

$$\begin{aligned} &\forall [\vec{x} \in A_F \vee \xi_F(\vec{x}) \wedge \vec{x} \notin D_F \supset F(\vec{x})], \\ &\forall [\vec{x} \in (A_{-F} - D_{-F}) \vee \xi_{-F}(\vec{x}) \wedge \vec{x} \notin D_{-F} \supset \neg F(\vec{x})], \end{aligned}$$

where F ranges over fluents, ξ_F (resp. ξ_{-F}) is the ewff defining F (resp. $\neg F$) in Σ (c.f. Definition 4), and

1. $A_F = \{\vec{d} \mid (\vec{d}, \psi) \in \gamma_F^+(\alpha) \text{ and } V[\Sigma, \psi] = 1\},$
 $D_{-F} = \{\vec{d} \mid (\vec{d}, \psi) \in \gamma_F^+(\alpha) \text{ and } V[\Sigma, \psi] \neq 0\};$
2. $A_{-F} = \{\vec{d} \mid (\vec{d}, \psi) \in \gamma_F^-(\alpha) \text{ and } V[\Sigma, \psi] = 1\},$
 $D_F = \{\vec{d} \mid (\vec{d}, \psi) \in \gamma_F^-(\alpha) \text{ and } V[\Sigma, \psi] \neq 0\}.$

Then we get the following:

Theorem 4 Let $\mathcal{D}$ be local-effect and Σ be proper. Then $\mathcal{P}_\alpha(\Sigma)$ is a weak progression of Σ wrt α .

Let $\delta = \langle \alpha_1, \dots, \alpha_n \rangle$. We use $\mathcal{P}_\delta$ to denote $\mathcal{P}_{\alpha_n} \circ \dots \circ \mathcal{P}_{\alpha_1}$. By a simple induction, we have that for every $\phi \in \mathcal{L}$, $V[\mathcal{P}_\delta(\Sigma), \phi] = V[\Sigma, \mathcal{R}_\delta(\phi)]$.

The intuition behind A_F and D_{-F} is simple. For $\vec{d} \in A_F$, $F(\vec{d})$ will become true in every possible successor state, so we add $F(\vec{d})$ to Σ . For $\vec{d} \in D_{-F}$, $F(\vec{d})$ may become true in some possible successor state, so we delete $\neg F(\vec{d})$ from Σ . Now consider our blocks world example. Let $\Sigma = \{on(c_1, c_2), clear(c_1), clear(c_3), eh(c_1)\}$. After action $move(c_1, c_2, c_3)$ is performed, we add $clear(c_2)$, $\neg clear(c_3)$, $on(c_1, c_3)$, and $\neg on(c_1, c_2)$ to Σ , and delete $clear(c_3)$, $\neg clear(c_2)$, $on(c_1, c_2)$, $\neg on(c_1, c_3)$, $eh(c_1)$, and $\neg eh(c_1)$ from Σ . We delete $eh(c_1)$ because if $eh(c_3)$ holds in the current state, $eh(c_1)$ will become false in the successor state; similarly, we delete $\neg eh(c_1)$.

We now define a reasoning procedure PV to solve the projection task using weak progression and V as follows:

Definition 9 Let $\mathcal{D}$ be a local-effect action theory with a proper Σ_0 . We define $PV[\delta, \phi]$ as $V[\mathcal{P}_\delta(\Sigma_0), \phi]$.

Now suppose that $PV[\delta, \phi] = 1$. Then $V[\Sigma_0, \mathcal{R}_\delta(\phi)] = V[\mathcal{P}_\delta(\Sigma_0), \phi] = 1$. By soundness of V , $\Sigma_0 \models_{\mathcal{E}} \mathcal{R}_\delta(\phi)$. By the Regression Theorem, $\mathcal{D} \models_{\mathcal{E}} \phi[do(\delta, S_0)]$. Similarly, if

$PV[\delta, \phi] = 0$, then $\mathcal{D} \models_{\mathcal{E}} \neg \phi[do(\delta, S_0)]$. Thus PV is logically sound for projection. It is easy to see that $\mathcal{P}_\alpha(\Sigma)$ can be computed in $O(n)$ time, where n is the size of Σ . By Corollary 2, we have the following tractability result:

Theorem 5 Let $\mathcal{D}$ be a local-effect action theory with a proper Σ_0 , and let $\phi \in \mathcal{L}^k$. Then $PV[\delta, \phi]$ can be computed in time $O(pn + lmn^k)$, where p is the length of δ , l the size of ϕ , m the e-size of Σ_0 , and n the size of Σ_0 .

Thus PV provides an efficient and logically sound solution to the projection problem despite the incomplete knowledge. In the next two sections, we will explore under what conditions, PV is also logically complete.

4 A Completeness Result

Since PV uses V , it is not surprising that we need a query to be in normal form for logical completeness. In this section, we will show that the only other thing we need is for the initial KB to have complete knowledge of the context of any context-dependent actions.

More precisely, we say that a KB Σ is *complete* wrt a set G of ground atoms if for all $l \in G$, either $\Sigma \models_{\mathcal{E}} l$ or $\Sigma \models_{\mathcal{E}} \neg l$. A KB Σ is complete wrt a predicate P if it is complete wrt all ground atoms of P . Now let Σ be proper, and ϕ a quantifier-free sentence such that Σ is complete wrt all atoms of ϕ . Then it is easy to see that $V[\Sigma, \phi]$ is either 0 or 1.

Definition 10 A KB Σ is context-complete (wrt $\mathcal{D}$) if it is complete wrt every predicate appearing in every γ_F^+ and γ_F^- .

So Σ is context-complete if it has complete knowledge about the predicates in the context of any context-dependent actions. For example, in our blocks world, a Σ is context-complete if it is complete wrt eh ; it may be incomplete wrt $clear$ and on . So context-completeness still allows incomplete knowledge.

There are two useful special cases where we get context-completeness. An SSA is *equality-only* if no predicate appears in γ_F^+ or γ_F^- . Obviously, any Σ is context-complete wrt equality-only SSAs. Indeed, many SSAs we come across are equality-only. An SSA is *context-free* if no fluent appears in γ_F^+ or γ_F^- . It is reasonable to assume that an agent has complete knowledge about situation-independent predicates. Under such an assumption, any Σ is context-complete wrt context-free SSAs.

The logical completeness of PV is obtained by showing that progression preserves context-completeness and that under context-completeness, our progression coincides with classical progression.

Theorem 6 Let Σ be context-complete. Then

1. $\mathcal{P}_\alpha(\Sigma)$ is context-complete too;
2. $\mathcal{P}_\alpha(\Sigma)$ is a classical progression of Σ .

Proof: (2) We prove that for every model M' , $M' \models \mathcal{P}_\alpha(\Sigma)$ iff there is a model M s.t. $M \models \Sigma$ and M' is the successor state of M wrt α . For the only-if direction, we construct M as follows: for every fluent atom $F(\vec{c})$, if $V[\Sigma, F(\vec{c})] = 1$, then $M \models F(\vec{c})$; if $V[\Sigma, F(\vec{c})] = 0$, then $M \models \neg F(\vec{c})$; otherwise, $M \models F(\vec{c})$ iff $M' \models F(\vec{c})$. The proof uses the fact that $V[\Sigma, \gamma_F^*(\alpha)(\vec{c})] \in \{0, 1\}$, where $*$ is + or -. ■

So under context-completeness, our progression preserves classical entailment. Now let Σ_0 be context-complete. By a simple induction, we have: for every $\phi \in \mathcal{L}$, $\mathcal{P}_\delta(\Sigma_0) \models_\varepsilon \phi$ iff $\Sigma_0 \models_\varepsilon \mathcal{R}_\delta(\phi)$. Now let $\phi \in \mathcal{NF}$. By completeness of V for $\mathcal{NF}$, $V[\mathcal{P}_\delta(\Sigma_0), \phi] = 1$ iff $\mathcal{P}_\delta(\Sigma_0) \models_\varepsilon \phi$ iff $\Sigma_0 \models_\varepsilon \mathcal{R}_\delta(\phi)$ iff $\mathcal{D} \models_\varepsilon \phi[do(\delta, S_0)]$. Thus when the initial KB is context-complete and the query is in normal form, PV is logically complete for projection.

5 Incorporating Sensing

In many applications, it is asking too much to require complete knowledge in the initial KB about the context of the context-dependent actions. In this section, we follow de Giacomo and Levesque [1999] and relax this restriction in two ways: first, we only need context-completeness relative to the sequence of actions and the query in question; second, we can achieve this local context-completeness dynamically by resorting to sensing actions, that is, actions that get knowledge from outside the system. In other words, we show that when a history of actions and sensing results is “just-in-time” for a normal form query, PV is once again logically complete.

We first extend our account of action and change to incorporate sensing. Assume that in addition to ordinary actions that change the world, we also have binary sensing actions that do not change the world but tell the agent whether some condition ϕ holds in the current situation. We use the predicate $SF(a, s)$ to characterize what the sensing action tells the agent about the world. Now our basic action theory has an extra component $\mathcal{D}_{sf}$, which is a set of sensed fluent axioms (SFAs), one for each action, of the form $SF(A(\vec{x}), s) \equiv \phi_A(\vec{x})[s]$, where $\phi_A \in \mathcal{L}$. We say that $\mathcal{D}_{sf}$ is atomic if each ϕ_A is an atom.

For instance, we may add three sensing actions to the blocks world example: $sense_{clear}(x)$, $sense_{on}(x, y)$, and $sense_{eh}(x)$. The axiom $SF(sense_{eh}(x), s) \equiv eh(x, s)$ says that the action $sense_{eh}(x)$ tells the agent if $eh(x, s)$ holds.

To describe a sequence of actions and sensing results, we use the notion of a *history*, that is, a sequence of pairs (α, μ) where α is a ground action and $\mu \in \{0, 1\}$ is the sensing result: when α is an ordinary action, we simply let $\mu = 1$. We use $end(\sigma)$ to denote the end situation of history σ , and $Sensed(\sigma)$ to denote the situation calculus formula stating all sensing results of σ . Formally,

- $end(\varepsilon) = S_0$, where ε is the empty history;
 $end(\sigma \cdot (\alpha, \mu)) = do(\alpha, end(\sigma))$.
- $Sensed(\varepsilon) = True$;
 $Sensed(\sigma \cdot (\alpha, 1)) = Sensed(\sigma) \wedge SF(\alpha, end(\sigma))$;
 $Sensed(\sigma \cdot (\alpha, 0)) = Sensed(\sigma) \wedge \neg SF(\alpha, end(\sigma))$.

Naturally, we are only interested in consistent histories, that is, histories with reasonable sensing results. Formally,

Definition 11 A history σ is consistent if $\mathcal{E} \cup \mathcal{D} \cup \{Sensed(\sigma)\}$ is a consistent theory.

Now the projection problem including sensing is formulated as deciding if $\mathcal{D} \cup \{Sensed(\sigma)\} \models_\varepsilon \phi[end(\sigma)]$, where $\phi \in \mathcal{L}$, and σ is a consistent history.

In the rest of this section, we assume that $\mathcal{D}_{sf}$ is atomic. To prepare for the definition of just-in-time-history, we first

extend our regression and progression operators to incorporate sensing. The ideas are quite simple. For example, if we regress the formula $clear(c_1) \wedge eh(c_1)$ wrt $(sense_{eh}(c_1), 1)$, we should obtain $clear(c_1)$. If we progress a proper KB Σ wrt $(sense_{eh}(c_1), 1)$, we should obtain $\Sigma \cup \{eh(c_1)\}$.

Let $\alpha = A(\vec{c})$ be a ground action, and let $\mu \in \{0, 1\}$. We define $\mathcal{R}_{(\alpha, \mu)}(\phi)$ as follows: If α is an ordinary action, then $\mathcal{R}_{(\alpha, \mu)}(\phi) = \mathcal{R}_\alpha(\phi)$. Otherwise, let the SFA be $SF(A(\vec{x}), s) \equiv F(\vec{x}, s)$. Then $\mathcal{R}_{(\alpha, \mu)}(\phi)$ is the formula obtained from ϕ by replacing each atom $F(\vec{t})$ with $\vec{t} = \vec{c} \vee F(\vec{t})$ when $\mu = 1$ and with $F(\vec{t}) \wedge \vec{t} \neq \vec{c}$ when $\mu = 0$. We have the following extended regression theorem:

Theorem 7 Let σ be a consistent history. Then $\mathcal{D} \cup \{Sensed(\sigma)\} \models_\varepsilon \phi[end(\sigma)]$ iff $\Sigma_0 \models_\varepsilon \mathcal{R}_\sigma(\phi)$.

Now we turn to progression with sensing. Let Σ be proper. Let $\alpha = A(\vec{c})$ be a ground action, and let $\mu \in \{0, 1\}$. We define $\mathcal{P}_{(\alpha, \mu)}(\Sigma)$ as follows: If α is an ordinary action, then $\mathcal{P}_{(\alpha, \mu)}(\Sigma) = \mathcal{P}_\alpha(\Sigma)$. Otherwise, let the SFA be $SF(A(\vec{x}), s) \equiv F(\vec{x}, s)$. Then $\mathcal{P}_{(\alpha, 1)}(\Sigma) = \Sigma \cup \{F(\vec{c})\}$, and $\mathcal{P}_{(\alpha, 0)}(\Sigma) = \Sigma \cup \{\neg F(\vec{c})\}$. We have the following extended progression theorem:

Theorem 8 Let σ be a consistent history. Then

1. $\mathcal{E} \cup \mathcal{P}_\sigma(\Sigma_0)$ is consistent. Hence $\mathcal{P}_\sigma(\Sigma_0)$ is proper.
2. For every $\phi \in \mathcal{L}$, $V[\mathcal{P}_\sigma(\Sigma_0), \phi] = V[\Sigma_0, \mathcal{R}_\sigma(\phi)]$.

Another concept we need is dependency set.

Definition 12 The dependency set of a formula ϕ wrt an ordinary action α , denoted by $DS_\alpha(\phi)$, is the set of ground atoms that appear in $\gamma_F^+(\alpha)$ or $\gamma_F^-(\alpha)$ for some fluent F in ϕ .

For example, let $\alpha = move(c_1, c_2, c_3)$. Then $DS_\alpha(clear(x))$ is the empty set, and $DS_\alpha(eh(x)) = \{eh(c_3)\}$.

Definition 13 Let σ be a consistent history, and $\phi \in \mathcal{L}$. We say that σ is a just-in-time-history (JIT-history) for ϕ if for every division $\sigma_1 \cdot (\alpha, \mu) \cdot \sigma_2$ of σ such that α is an ordinary action, $\mathcal{P}_{\sigma_1}(\Sigma_0)$ is complete wrt $DS_\alpha(\mathcal{R}_{\sigma_2}(\phi))$.

Intuitively, σ is a JIT-history for ϕ if whenever performing an ordinary action α , the agent has complete knowledge about the context of α wrt fluents related to ϕ . This complete knowledge may come from the sensing actions preceding α . For example, let $\Sigma_0 = \{clear(c_1), on(c_1, c_2), clear(c_3)\}$. Then the history $(sense_{eh}(c_3), 1) \cdot (move(c_1, c_2, c_3), 1)$ is a JIT history for the formula $\exists x[clear(x) \wedge eh(x)]$. Note that here the agent has incomplete knowledge about both fluents $clear$ and eh . Thus a JIT history does not require complete knowledge about the component fluents of the query.

By a proof essentially the same as the one in the previous section, we get the following result:

Theorem 9 Let σ be a JIT-history for ϕ . Then $\mathcal{P}_\sigma(\Sigma_0) \models_\varepsilon \phi$ iff $\Sigma_0 \models_\varepsilon \mathcal{R}_\sigma(\phi)$.

Now let σ be a JIT-history for $\phi \in \mathcal{NF}$. Then we have $V[\mathcal{P}_\sigma(\Sigma_0), \phi] = 1$ iff $\mathcal{P}_\sigma(\Sigma_0) \models_\varepsilon \phi$ iff $\Sigma_0 \models_\varepsilon \mathcal{R}_\sigma(\phi)$ iff $\mathcal{D} \cup \{Sensed(\sigma)\} \models_\varepsilon \phi[end(\sigma)]$. Thus when a history is just-in-time for a normal form query, PV is again logically complete for projection.

6 Related Work

As mentioned in the introduction, Lin and Reiter [1997] give a systematic study of classical progression. As a part of their study, they view STRIPS as a mechanism for computing progression and thus provide a logical semantics for STRIPS. In this respect, they consider strictly context-free SSAs and initial KBs in the form of relational databases or sets of ground literals. These are special cases of local-effect SSAs and proper KBs, and our weak progression coincides with classical progression in these cases. Son and Baral [2001] propose the so-called 0-approximation semantics for an extension of action language $\mathcal{A}$. They define an a-state (approximate state) as a consistent set of fluent literals, and define a transition function which maps an a-state and an action into the next a-state. So 0-approximation is essentially a kind of approximate progression. However, their work is restricted to the propositional case, and our progression coincides with theirs therein. Amir and Russell [2003] present efficient algorithms for (approximate) logical filtering, where filtering means updating an agent's belief state in response to actions and observations. So logical filtering is essentially progression. But again, their work is restricted to the propositional case. De Giacomo and Mancini [2004] study how to exploit relational database technology to implement progression, but only when the initial KB has complete knowledge. We get to use database techniques in the incomplete case via the results in [Liu and Levesque, 2003].

The idea of progression is widely used in planning under incomplete knowledge. Most systems use propositional representations, for example, BDDs [Cimatti and Roveri, 2000], and clauses [Brafman and Hoffmann, 2004]. Being propositional makes it possible for them to consider arbitrary incomplete knowledge and perform classical progression. Although techniques are employed so that the systems can achieve reasonable performance in practice, there is no theoretical guarantee of the tractability of their solutions to projection. The PKS system of Petrick and Bacchus [2002] uses a first-order representation. The form of incomplete knowledge they consider is mainly a set of ground literals but with some other features. The general idea behind their progression is similar to ours, but without a semantical characterization of what it preserves. Moreover, they do not address the issue of the restrictions they need to get completeness.

The idea of JIT histories in this paper comes from [De Giacomo and Levesque, 1999]. They use JIT histories to obtain complete knowledge about the component fluents of the query. However, we use JIT histories only to obtain complete knowledge about the context of actions to be performed.

7 Conclusions

In this paper, we have proposed a tractable, sound, and sometimes complete solution to the projection problem in the presence of context-dependent actions and incomplete first-order knowledge in the form of a proper KB. Our solution is via a version of progression that preserves properness and V .

For simplicity of presentation, in this paper we require actions to have local effects only, and we make the extra requirement that context formulas (c.f. Definition 5) be quantifier-

free. However, the soundness and completeness results in this paper will still hold if we relax these two requirements. The tractability result will also hold if context formulas use a bounded number of variables. As for local effects, the tractability result only needs them to ensure that the progressed KB is not much larger than the original KB. We believe that there are other ways of doing this that would include a substantial class of actions with universal effects.

For the future, we would like to conduct experimental evaluation of our solution to projection, and apply it to first-order planning systems. Also, we would like to extend our work here to deal with functional fluents and disjunctive incomplete knowledge.

References

- [Amir and Russell, 2003] E. Amir and S. Russell. Logical filtering. In *Proc. IJCAI-03*, pages 75–82, 2003.
- [Brafman and Hoffmann, 2004] R. Brafman and J. Hoffmann. Conformant planning via heuristic forward search. In *Proc. ICAPS-04*, pages 355–364, 2004.
- [Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.
- [De Giacomo and Levesque, 1999] G. De Giacomo and H. J. Levesque. Projection using regression and sensors. In *Proc. IJCAI-99*, pages 160–165, 1999.
- [De Giacomo and Mancini, 2004] G. De Giacomo and T. Mancini. Scaling up reasoning about actions using relational database technology. In *Proc. AAAI-04*, pages 245–250, 2004.
- [Levesque *et al.*, 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.
- [Levesque, 1998] H. J. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. KR-98*, pages 14–23, 1998.
- [Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1–2):131–167, 1997.
- [Liu and Levesque, 2003] Y. Liu and H. J. Levesque. A tractability result for reasoning with incomplete first-order knowledge bases. In *Proc. IJCAI-03*, pages 83–88, 2003.
- [McCarthy and Hayes, 1969] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502, 1969.
- [Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. AIPS-02*, pages 212–222, 2002.
- [Petrick and Levesque, 2002] R. Petrick and H. J. Levesque. Knowledge equivalence in combined action theories. In *Proc. KR-02*, pages 303–314, 2002.
- [Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [Son and Baral, 2001] T. C. Son and C. Baral. Formalizing sensing actions – A transition function based approach. *Artificial Intelligence*, 125(1–2):19–91, 2001.

Cognitive Modelling of Event Ordering Reasoning in Imagistic Domains¹

Laura S. Mastella¹, Mara Abel¹, Luís C. Lamb¹, Luiz F. De Ros²

¹Instituto de Informática and ²Instituto de Geociências

Universidade Federal do Rio Grande do Sul

Porto Alegre, RS 91501-970, Brazil

{mastella,marabel,lamb,lfders}@inf.ufrgs.br

Abstract

The inference of temporal information from past event occurrences in imagistic domains is relevant in several applications in knowledge engineering. In such applications, the order in which events have happened is imprinted in the domain as visual-spatial relations among its elements. Therefore, the interpretation of the relative ordering in which those events have occurred is essential for understanding the domain evolution. We propose a cognitive model for event ordering reasoning within domains whose elements have been modified by past events. From the analysis of cognitive abilities of experts we propose new ontology constructs for knowledge modelling associated to Problem-Solving Methods. We illustrate the effectiveness of the model by means of an application to an imagistic domain.

1 Introduction

The inference of temporal information from past event occurrences in imagistic domains is relevant in several applications in knowledge engineering [Thagard and Shelley, 1997]. There are several approaches to model temporal reasoning in intelligent systems such as algebras, logics, calculi and effective mechanisms to reason about time [Allen, 1983; Gabbay *et al.*, 1994]. Nonetheless, the development of ontological languages providing adequate support to formalise event-based temporal inferences in such domains is still incipient. In addition, most proposals consider an absolute notion of time, which can be of limited use in a number of applications.

However, here we aim at deriving relative temporal information from another dimension (the visual-spatial rela-

tions between the elements of the domain), which is foreign to most proposals developed so far. We are interested in proposing a cognitive model of the reasoning of an expert about an imagistic domain, when deriving a temporal relation from a visual-spatial relation.

Imagistic domains are characterised by visual-spatial relations requiring from the problem-solver the ability of applying visual recognition of objects, and from this initial recognition, to start the search and analytical methods in order to interpret these objects. Such abilities define a form of *imagistic reasoning* [Yip and Zhao, 1996], which cannot be described only in terms of the geometrical properties of objects, but by the meaning and significance that these objects have in selecting a particular reasoning path in a visual-spatial domain.

In some domains, the recognition of key features of elements and the identification of the visual-spatial relations among them is an important research issue [Ericsson and Smith, 1991]. For instance, a geologist identifies visual-spatial relations among rock constituents (called paragenetic relations), as does a physician when analysing medical images to identify a pathology.

In order to build an ontology and appropriate problem-solving methods for such domains, a long process of knowledge acquisition from experts was carried out. The analysis of the cognitive abilities of the expert led to the development of new ontology constructs for knowledge modelling associated to Problem-Solving Methods (PSM). These constructs and methods are then shown to be capable of modelling the expert's reasoning when deriving the sequence of events which led to the visual-spatial organisation of the domain under analysis.

Section 2 presents the basics of ontologies and their extensions with temporal constructs and a brief description of the PSMs that shall be used. Section 3 defines the main constructs and primitives of the cognitive model for event reasoning and PSMs to infer sequences of events. Section 4 describes the application of the developed models to an imagistic domain, namely *sedimentary petrography*. Section 5

¹ This work is supported by the Brazilian Research Council CNPq through the CTPetro programme grant 502009/2003-9. L.C. Lamb is partly supported by CAPES and CNPq.

presents an initial validation of the model. Section 6 concludes the paper and discusses directions for future research.

2 Preliminaries

An ontology is a formal explicit specification of a shared conceptualisation [Studer *et al.*, 1998]. Usually, domain ontologies represent static or declarative knowledge about a domain: the main concepts, their attributes, the relationships between them, axioms, rules, etc [Gómez-Pérez *et al.*, 2004]. [Gruber, 1993] has identified five types of ontological constructs: *classes*, which represent elements of a domain defined by a set of attributes and their possible values; *relations*, which represent the types of association between domain elements; *functions*, special relations that map one or several elements of the domain to a unique element; *axioms*, which are statements that are always true about the domain; and *instances*, the individuals of the ontology.

While ontologies describe the structural and static part of knowledge, the dynamic part is described using a PSM [Gómez-Pérez and Benjamins, 1999]. A PSM describes the reasoning process of a knowledge-based system (KBS), specifying the knowledge and data required by an inference process at a more abstract and structured level. This reasoning pattern is modelled by three related parts: (i) a *competence specification* related to the solution of a task, (ii) an *operational specification* described by high-level modelling primitives and (iii) *requirements/assumptions* of the method in terms of domain knowledge.

Recently, some authors have aimed at augmenting the expressive power of ontologies representing temporal information. A causal time ontology is proposed by [Kitamura *et al.*, 1997], in which an event represents instantaneous changes of qualitative values of parameters and their resulting values at a time point. The work of [Zhou and Fikes, 2000] treats both time points and time intervals as primitive elements of a time line, and the classes, relations, axioms and instances of the ontology are built upon those primitives. The extension of DAML to DAML-Time aims to develop an ontology of time that expresses temporal aspects common to any formalisation of time, such as temporal relations on instants and intervals [Hobbs and Pustejovsky, 2003]. [Bennett and Galton, 2004] propose a highly expressive language for representing temporal relationships and events, called VEL (Versatile Event Logic). A single VEL formula can contain both time-point and interval variables. In most works, the addition of an explicit temporal dimension to ontology constructs is typically defined by an absolute time stamp associated with objects of the ontology, i.e., time points or time intervals.

However, in several application domains, events are not to be interpreted as having time stamps labelling them. It is the *relative ordering* in which events have occurred that is of fundamental importance in understanding, analysing, and inferring the evolution of elements in such domains. The aim of our approach is on representing how the domain evolves through a sequence of events that act as operators transforming the state-space of a domain.

The cognitive model proposed here shall be applied for modelling domains that present evolving elements which have been modified by events that have occurred in an un-planned order. The order in which the events happened is imprinted in the domain as complex visual-spatial relations among the elements of the domain. The cognitive models we introduce here intend to represent how to identify the events from the domain elements' characteristics, and how to find the order in which the events have occurred out of the visual-spatial relations in such domains.

We deal with visual-spatial knowledge translated into a symbolic representation, instead of using typical numerical approaches for image processing. This follows a line of research outlined in [Kosslyn, 1994] which describes that some features (and types) of visual-spatial knowledge are dealt by the human brain as symbolic entities, not image representations; therefore we believe that these features can be represented by means of a symbolic approach. Moreover, the proposed cognitive models are not intended to recognise the elements of the domain, but to infer new information from their symbolic description made by the user of the expert system.

3 A Cognitive Model for Event Ordering

Let us now define the required constructs to deal with event ordering and the associated problem-solving methods.

3.1 An Ontology of Events

We propose an extension to the constructs of ontological representation - classes, relations, inference rules, axioms and instances - for evolving domains in order to capture the meaning of *events* and *temporal relations between them*. Such proposed constructs shall be applied for modelling domains that need to infer past sequences of occurrences to provide a better comprehension of the current state of the domain. Further, we aim at understanding how the domain elements were produced and by which events, and also at formalising the explanation through a representation of an event sequence. We define the new constructs as follows:

- *Events* are class-transforming constructs. They represent phenomena that generate or modify the elements of the domain. They are characterised by specific domain-dependent attributes, but not by a time stamp. Events are also described by rules that associate them to their products. Events are associated to each other by an ordering relation.
- *Temporal relation*: a construct proposed to represent the ordering relation between events. We have defined the binary relations *before*, *after* and *during* in order to reflect the ordering between events.

Furthermore, in our domain ontology, we extend the notion of *inference rules* to make them more expressive. They express functions between instances of relations, in addition to functions between instances of classes.

Most knowledge representation structures used in domain ontologies define rules as expressions about the *attributes* of objects, as in the following rule:

if classA.attribute1 = value-x then classB.attribute2 = value-y

We use this type of rule to model functions that identify events from their produced elements. The characteristics of the elements (expressed by class attributes in the ontology) are used to indicate the event that originated or modified the element. The instances of these rules define that when a particular value for an attribute is found in the description of an element then a determined event is indicated as the generating event. These inference rules are called *event indication rules*. In addition, we define an extended type of inference rule in knowledge representation ontologies. The purpose is to express functions between instances of *relations*, instead of classes. This second type of rule, referred to as *temporal implication rule*, is defined to allow the inference of binary *temporal relations* between events from *visual-spatial relations* between the elements, as in

if visual_relation(A,B) then temporal_relation(A,B)

Next, we present a PSM to model inferences about the ordering of events in ontologies built with these new types of constructs and inference rules.

3.2 A PSM for Ordering Events

In order to show that the proposed constructs are adequate and sufficient for inferring event ordering, the reasoning method was modelled as a PSM, abstracted from the heuristics used by a highly-trained expert in the domain. The PSM uses the temporal constructs to identify and order the events that originated the current characteristics of the domain.

The *competence* of our PSM takes a description of domain elements as input, identifies the events that acted over these elements and infers the ordering in which such events have occurred. The *assumptions* of our PSM are: (1) the characteristics imprinted in the domain elements indicate the event that generated the element; (2) the way elements are spatially disposed reflects the order in which they were originated in the domain. The *requirements* are the inference rules that associate characteristics of domain elements to their generating events and the inference rules that associate visual-spatial and ordering relations. The input of the PSM is a description of the visual-spatial features without any temporal reference. The sequence of events is inferred as the PSM output. The *operational specification* describes the inferences. The inferences are applied to the elements in the domain knowledge (represented as knowledge roles) in order to derive new information. The inferences used in our PSM can be detailed as follows:

- *Abstract*: select attributes that describe the relevant aspects of the element for problem-solving.
- *Relation abstract*: takes an element as input and outputs relevant relations between such element and others.
- *Specify*: takes a relation as input and outputs the related element (the second argument of the relation).
- *Cover*: given a set of rules and objects that match the premise of the rules, outputs the conclusion of the rule.
- *Order*: takes a pair of events and the ordering relation between them and produces an ordered pair of events.

The *event ordering model* is shown in Figure 1, as a graphic model that presents the knowledge roles, the information flow, and the inferences. It is explained as follows.

The knowledge role (label 1) represents elements in the application domain that is under analysis. The elements of the domain are related to each other through complex visual-spatial relations (label 2), which are abstracted from the element description. As visual-spatial relations are binary, the other argument of the relation can be specified as the related element (label 3), which is also a domain element. The result of these steps is a pair of domain elements associated by a visual-spatial relation.

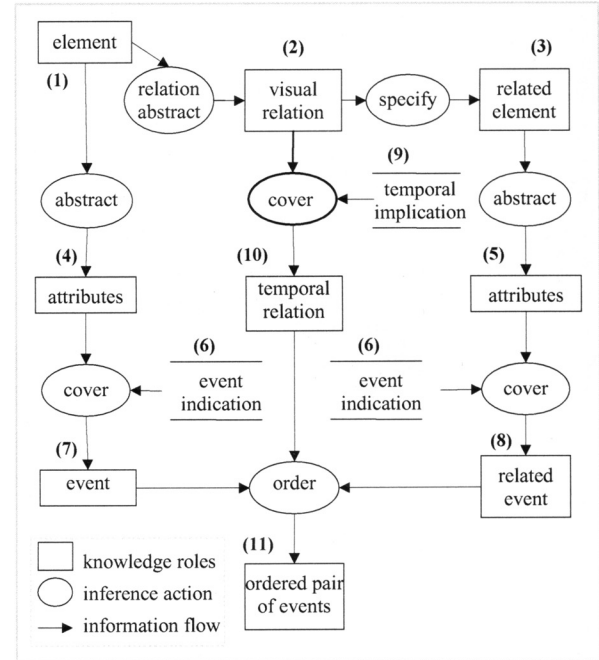


Figure 1: Problem-Solving Method for Event Ordering.

Every domain element is described by a set of attributes which defines its generating event. Relevant attributes (labelled as 4 and 5) are abstracted from the set of attributes. Event indication rules (label 6) cover the generating events (labelled as 7 and 8) from the attributes of the element.

The visual-spatial relation between domain elements is used to infer a temporal relation (label 10) using the temporal implication rules (label 9). From the temporal relation it is possible to order the events, producing an ordered pair of events (label 11). A detailed example of application of this reasoning method is presented in Section 5.

Thus, the above PSM can be used in domain ontologies modelled using the proposed temporal constructs with the objective of inferring temporal relations from visual-spatial relations in an imagistic domain.

4 Modelling an Imagistic Domain

We are now in a position to illustrate the application of the proposed models to an imagistic domain scenario.

4.1 Describing the Petrographic Domain

We consider *sedimentary petrography*, which aims at evaluating the economic prospects of oil fields and reservoirs by means of interpretation techniques from descriptions of rock samples. One particular technique widely used to determine the quality of a reservoir is called *diagenetic sequence interpretation* [Anjos *et al.*, 2000], which aims to infer the order in which diagenetic events occurred in a rock. *Diagenetic events* are physical-chemical processes that act over the sediments transforming them into solid rocks and modifying the porosity and permeability of a potential oil-reservoir. The visual-spatial relations among rock constituents reflect the changes undergone by the rock as a result of diagenetic events. In Figure 2 we show an example of rock sample and the visual-spatial relations between minerals (called *paragenetic relations*) that can be identified.

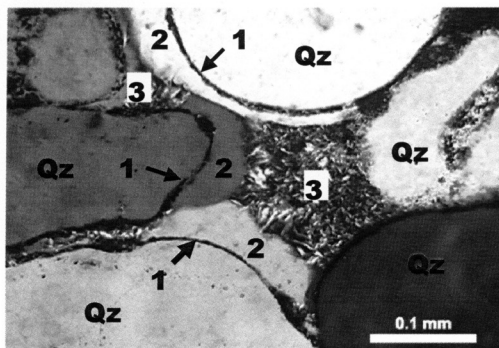


Figure 2: Detail view of a rock sample with (1) hematite covering quartz grains (Qz); (2) quartz covering the hematite; (3) quartz being covered by illite, which is covering the hematite.

Even though the arrangement is said to be spatial, they cannot be translated into simple spatial relations, since they are by themselves a result of a visual interpretation of particular rock aspects. This interpretation requires previous experience in mineral recognition that goes beyond the analysis of geometrical and topological characteristics. Figure 2 illustrates examples of paragenetic relations.²

The expert in petrography points out the ordering of events by observing how the constituents are spatially and visually related to each other, based on his extensive previous knowledge. Using a simple example, if one mineral appears to be on top of other mineral, it means that the former was generated in the rock later than the latter.

Some interpretations techniques used for the evaluation of oil reservoirs had already been modelled in the *Petro-Grapher* system, an intelligent database application to support the description and interpretation of sedimentary rock samples [Abel *et al.*, 2004]. The vocabulary of petrography was elicited as a result of previous work on the domain

[Abel *et al.*, 2003] and modelled as a domain ontology mostly structured as a partonomy³ of classes. The system implements interpretation tasks as PSMs over the domain ontology.

Although the petrography domain ontology had already been modelled, temporal aspects of the domain needed in the interpretation of diagenetic sequences were still lacking. These aspects had to be elicited later, by means of knowledge acquisition (KA) sessions with an expert in sedimentary petrography. In order to acknowledge the heuristics of the expert when performing the diagenetic sequence interpretation, several knowledge elicitation techniques have been applied [Cooke, 1994]. Firstly, we applied interviews and observation techniques to elicit the key vocabulary used in the interpretation, to determine the task's purpose and to understand how the interpretation is realised. Then, in order to reveal the expert's implicit knowledge (i.e. knowledge used by the expert that is not easily verbalised [Polanyi, 1974]), we applied card sorting and repertory grids techniques. Thus, the main results obtained in the KA session were insights about temporal aspects that needed to be modelled (such as diagenetic events and ordering relations) and an abstraction of the expert's reasoning steps when solving the interpretation task.

Next, we shall model the task of diagenetic sequence interpretation (i.e. how to identify the order in which events have occurred) by using the constructs proposed in Section 3 above, so as to represent the temporal aspects identified in the domain. We shall also show how to instantiate the PSM for event ordering of Figure 1 to infer the sequence of diagenetic events that occurred in a rock.

4.2 From the Event Ontology to the Petrography Domain

Let us now show how the petrography domain ontology was extended with temporal aspects using the new temporal constructs.

a) Rock Constituents (modelled as *classes*). Rock constituents are minerals and pores that form a rock. They are represented in the ontology using the *class* construct, since they are elements of the domain. The characteristics of the constituents are described as *attributes* of the class that represents it. It is important to represent these features since they reflect how this constituent was produced or modified, indicating its generating diagenetic events.

b) Paragenetic Relations (modelled as *relations*). Paragenetic relations are visual-spatial arrangements among constituents. Since a paragenetic relation is defined between a pair of constituents, the construct used in the ontology is a *binary relation*.

c) Diagenetic Events (modelled as *events*). These events are physical-chemical phenomena of rock consolidation and are represented in the model as an *event*. The expert does

² There are several kinds of paragenetic relations that can be identified such as *covering*, *replacing*, *dissolving*. Each particular paragenetic relation creates visual-spatial relations between different types of constituents. For instance, the relation *dissolution* occurs only between a pore and a diagenetic constituent.

³ Partonomy is a domain organization based on a part-of relation rather than on a kind-of (is-a) relation [Martínez-Béjar and Fernández-Breis, 2000]. E.g., a foot is part of a leg which is part of a body.

not take into account the absolute period of time the event has happened, but only the order in which the diagenetic event has occurred in relation to other events.

d) Ordering Relation (modelled as a temporal relation). A rock is formed as a result of several different diagenetic events, which can happen in a simultaneous or in a sequential way. To simplify the computational treatment of the sequence, we treat the ordering of events in pairs, as does the expert. The relations between pairs of events were represented using the *temporal relation* construct to model the temporal relations *after*, *before*, and *during*.

e) Inference rules (modelled as functions). As explained in Section 3.1, the expert is able to indicate the generating events by analysing the characteristics of the constituents. For instance, when the attribute *modifier* of a constituent holds the value *deformed*, it is possible to conclude that the event that transformed the constituent is *compaction*. Hence, it was necessary to represent this knowledge as *event indication rules*. These inference rules define an association between *constituents* and *diagenetic events*, e.g. (R1) below:

if constituent.modifier = deformed
then event.event_name = compaction R1

After identifying the events, the expert is able to infer the order in which they occurred by means of the visual-spatial (paragenetic) relations among the constituents. For instance, when a mineral appears to be *covering* (on top of) another mineral, the expert says that the event that formed the first mineral occurred later than the event that formed the latter. The first part of this particular expert's rule is assuming a *paragenetic relation* between constituents. The second part is defining an *ordering relation* between events. Thus, we need to represent this knowledge as *temporal implication rules*, defined in Section 3.1. An example of this type of rule is the following (R2):

if covering(constituent1, constituent2)
and produced_by(constituent1, event1)
and produced_by(constituent2, event2) R2
then after(event1, event2)

If *constituent1* is covering *constituent2* and the events that produced them are, respectively, *event1* and *event2*, then the conclusion is that *event1* happened after *event2*.

5 Initial Validation of the Cognitive Models

The temporal constructs and the proposed PSM have been applied to the petrography domain. They were implemented as an inference module within the *PetroGrapher* system, called *diagenetic sequence interpretation module*. Real rock samples were described by the expert in the *PetroGrapher* system and he also provided a previous interpretation of the sequence of diagenetic events. Thus, the interpretation produced by the algorithm was compared to the expert's interpretation of the sequence. In order to illustrate this initial validation, a step-by-step interpretation of a sample is described in what follows.

As input to the algorithm we have a description of a rock sample, which is composed by 132 attributes that detail the description of the constituents of the rock. The algorithm

analyses each constituent individually. For example, the description of one constituent is as follows:

[quartz - intergranular discontinuous pore-lining - covering <diagenetic-constituents> - hematite]

In the PSM depicted in Figure 1, this line corresponds to the knowledge role labelled as *element* (1). The next steps are: abstracting the paragenetic relation (*covering <diagenetic-constituents>*) from the description (role *visual relation* - label 2) and specifying the constituent visually related to the former (role *related element* - label 3). Thus, the related constituent is *hematite*, described as follows:

[hematite - coating - intergranular continuous pore lining - within intergranular primary porosity - macroporosity intergranular]

The result of these steps is a pair of constituents associated by a paragenetic relation (*quartz - covering - hematite*).

A rock constituent is characterised by some attributes which provide an indication of the generating event of the constituent. The algorithm abstracts such attributes from the description of the constituent. The attributes are the roles labelled as 4 and 5 in Figure 1. The relevant attributes are used as premises for the event indication rules (label 6), returning the event that produced or modified the constituent. These last inference steps are performed for both constituents, resulting in a pair of diagenetic events (labelled as 7 and 8). For the constituents considered in this example, the following events are inferred by the rules: *precipitation of quartz* and *precipitation of hematite*.

The paragenetic relation between constituents (label 2) is used as premise for the temporal inference rules (label 9). The result is the ordering relation (label 10) (e.g. given in rule R2, where the paragenetic relation *covering* implies the temporal relation *after*). The ordering relation and the pair of events are used as inputs for the inference named *order*, which results an ordered pair of events (label 11), which, in this case is

after(Cementation of Quartz, Cementation of Hematite)

The algorithm that mechanizes the PSM is implemented as a loop that stops when there are no more constituents to be evaluated. The algorithm outputs a list of ordered pairs, e.g. the following list for the sample being evaluated:

after(Cementation of Hematite, Deposition of Detrital Quartz)
after(Cementation of Quartz, Cementation of Hematite)
before(Cementation of Quartz, Cementation of Illite)

We use a directed graph to create a sequence of events from the list of ordered pairs. An edge from the event that comes before (in the source vertex) to the event that comes later in time (in the target vertex) defines the ordering relation. For instance, the result of mapping the list above to a directed graph is represented in Figure 3.

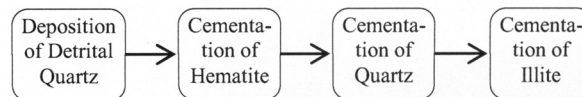


Figure 3: Events sequence produced by the inference algorithm

Such inferences depend on how clear the visual-spatial arrangements of constituents are in a particular sample.

However, in some cases, not even the expert is able to produce a complete sequence of events, because some paragenetic relations may be visible (and then described) in one sample, but not in another one. Thus, for some rock samples the algorithm produces sequences of events that are not totally connected leading to an incomplete interpretation of the sequence.

Although the resulting sequence may sometimes be incomplete, the produced outcome is certainly relevant to the domain. Any sequence of events that can be inferred from a rock description is essential in understanding how the porosity and the permeability of the rock were affected, and how this influences the quality of the oil reservoir.

6 Conclusion

We have presented a new cognitive model for reasoning about event ordering relations in imagistic domains. The proposed model is composed of an ontology of events and an associated Problem-Solving Method. From extensive sessions of knowledge acquisition from an expert we were able to: (i) identify the need for new temporal aspects in a domain ontology, which led us to the formalisation of such aspects as new ontological constructs to represent relative notions of time - so far lacking in the literature; (ii) to abstract the PSM that models a process of event ordering interpretation, and (iii) to identify the need of extending the notion of inference rules in an ontology so as to allow the inference of temporal relations from visual-spatial relations.

The proposed model aims to mimic the reasoning process of an expert who infers events that generated the current domain characteristics from complex visual-spatial relations between domain elements. We have illustrated the effectiveness of the model in a real-world application and initially validated it through the implementation of the model in an expert system. Further experiments leading towards a thorough validation of the system are currently being carried out. We believe that it is possible to use the proposed models in other domains, e.g. to infer the sequence of events that may have caused climate changes by analysing ice testimonies in glaciology. Further, in archaeology it is of fundamental importance to define a sequence of events in the history of the object that led to its current shape and structure [Thagard and Shelley, 1997].

Finally, this work can also be seen as a starting point towards investigating *abductive procedures* that correspond to the reasoning of the expert. This would corroborate Peirce's original claims regarding abduction as a form of visual reasoning [Thagard and Shelley, 1997]. In particular, the models presented here model the way an expert explains and interprets visual-spatial relations in search for the best explanation about the sequence of events that caused them.

References

- [Abel *et al.*, 2003] M. Abel, J. Campbell and L. F. D. Ros. Knowledge acquisition techniques for visual expertise: A study of oil-reservoir evaluation. In *Proc. of the 2nd ICPI - IJCAI03*, pages 74-79, Acapulco, 2003.
- [Abel *et al.*, 2004] M. Abel, L. A. L. Silva, L. F. D. Ros, L. Mastella, J. Campbell and T. Novello. Petrographer: Managing petrographic data and knowledge using an intelligent database application. *Expert Systems with Applications*, 26: 9-18, 2004.
- [Allen, 1983] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26: 832-843, 1983.
- [Anjos *et al.*, 2000] S. M. C. Anjos, L. F. De Ros, R. S. Souza, C. M. A. Silva and C. L. Sombra. Depositional and diagenetic controls on the reservoir quality of lower cretaceous sandstones. *Am. Ass. Petrol. Geol. Bull.*, 84: 1719-1742, 2000.
- [Bennett and Galton, 2004] B. Bennett and A. Galton. A unifying semantics for time and events. *Artif. Intell.*, 153: 13-48, 2004.
- [Cooke, 1994] N. Cooke. Varieties of knowledge elicitation techniques. *Int. Jnl. Hum.-Comp. Studies*, 41: 801-849, 1994.
- [Ericsson and Smith, 1991] K. Ericsson and J. Smith. *Toward a general theory of expertise: Prospects and limits*. Cambridge U. Press, 1991.
- [Gabbay *et al.*, 1994] D. M. Gabbay, I. Hodkinson and M. Reynolds. *Temporal logic*. Oxford U. Press, 1994.
- [Gómez-Pérez and Benjamins, 1999] A. Gómez-Pérez and V. R. Benjamins. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *Proc. of KRR-5 - IJCAI99*, pages 1.1-1.15, 1999.
- [Gómez-Pérez *et al.*, 2004] A. Gómez-Pérez, M. Fernández-López and O. Corcho. *Ontological engineering*. Springer, 2004.
- [Gruber, 1993] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquisition*, 5: 199-220, 1993.
- [Hobbs and Pustejovsky, 2003] J. Hobbs and J. Pustejovsky. Annotating and reasoning about time and events. In *Proc. AAAI Spr. Symp. Logical Form. Commonsense Reas.*, 2003.
- [Kitamura *et al.*, 1997] Y. Kitamura, M. Ikeda and R. Mizoguchi. A causal time ontology for qualitative reasoning. In *Proc. of the 15th IJCAI*, pages 501-507, 1997.
- [Kosslyn, 1994] S. M. Kosslyn. Resolving the imagery debates. *Image and brain: The resolution of the imagery debate*. S. M. Kosslyn (ed.), MIT Press: 2-23, 1994.
- [Martínez-Béjar and Fernández-Breis, 2000] R. Martínez-Béjar and J. T. Fernández-Breis. A cooperative tool for facilitating knowledge management. *Expert Systems with Applications*, 18: 315-330, 2000.
- [Polanyi, 1974] M. Polanyi. *Personal knowledge*. The Univ. of Chicago Press, 1974.
- [Studer *et al.*, 1998] R. Studer, V. R. Benjamins and D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25: 161-197, 1998.
- [Thagard and Shelley, 1997] P. Thagard and C. Shelley. Abductive reasoning: Logic, visual thinking, and coherence. *Logic and scientific methods*. M. D. Chiara (ed.), Kluwer: 413-427, 1997.
- [Yip and Zhao, 1996] K. Yip and F. Zhao. Spatial aggregation: Theory and applications. *Jnl. of Art. Intel. Research*, 5: 01-26, 1996.
- [Zhou and Fikes, 2000] Q. Zhou and R. Fikes. *A reusable time ontology*, Knowledge Systems Lab, Stanford U., 2000.

Propositional Abduction is Almost Always Hard

Gustav Nordh*

Dept. of Computer & Information Science
Linköpings Universitet
S-581 83 Linköping, Sweden
gusno@ida.liu.se

Bruno Zanuttini

GREYC, UMR CNRS 6072
Université de Caen, Bd. du Maréchal Juin
F-14 032 Caen Cedex, France
zanutti@info.unicaen.fr

Abstract

Abduction is a fundamental form of nonmonotonic reasoning that aims at finding explanations for observed manifestations. Applications of this process range from car configuration to medical diagnosis. We study here its computational complexity in the case where the application domain is described by a propositional theory built upon a fixed constraint language and the hypotheses and manifestations are described by sets of literals. We show that depending on the language the problem is either polynomial-time solvable, NP-complete, or Σ_2^P -complete. In particular, we show that under the assumption $P \neq NP$, only languages that are affine of width 2 have a polynomial algorithm, and we exhibit very weak conditions for NP-hardness.

1 Introduction

In this paper we investigate the computational complexity of abduction, a method of reasoning extensively studied by Peirce [1955]. Abductive reasoning is used to search for explanations of observed manifestations. The importance of this problem to Artificial Intelligence was first emphasized by Morgan [1971] and Pople [1973].

The abductive process has demonstrated its practical importance in many domains. In particular, it has been used to formalize processes in medical diagnosis [Bylander *et al.*, 1991], text interpretation [Hobbs *et al.*, 1993], system diagnosis [Stumptner and Wotawa, 2001] and configuration problems [Amilhastre *et al.*, 2002].

In this paper we are interested in propositional logic-based abduction, i.e., the background knowledge is represented by a propositional theory. Even in this framework several formalizations of the problem have been studied in the literature, depending on the syntactic restrictions imposed to the manifestations and on the hypotheses over which explanations must be formed. We use the following formalization: Given a propositional theory T formalizing a particular application domain, a set M of literals describing a set of manifestations, and a set H of literals containing possible hypotheses, find an

explanation for M , that is, a set $E \subseteq H$ such that $T \cup E$ is consistent and logically entails M . This framework is more general than most frameworks studied in the literature; an exception is the definition by Marquis [Marquis, 2000], which allows the manifestation to be encoded by any propositional formula.

Example 1 Consider the following example from the domain of motor vehicles, inspired by [Console *et al.*, 1991].

$$\begin{aligned} T &= (\neg \text{rich_mixture} \vee \neg \text{lean_mixture}) \wedge \\ &\quad (\text{rich_mixture} \rightarrow \neg \text{low_fuel_consumption}) \wedge \\ &\quad (\text{lean_mixture} \rightarrow \text{overheating}) \wedge \\ &\quad (\text{low_water} \rightarrow \text{overheating}), \end{aligned}$$
$$H = \{\text{rich_mixture}, \text{lean_mixture}, \text{low_water}\},$$
$$M = \{\neg \text{low_fuel_consumption}, \text{overheating}\}.$$

Then $E = \{\text{rich_mixture}, \text{low_water}\}$ is an explanation of M , while, e.g., $\{\text{lean_mixture}, \text{low_water}\}$ is not because it does not explain $\neg \text{low_fuel_consumption}$, and $\{\text{rich_mixture}, \text{lean_mixture}\}$ is not because it is inconsistent with T .

Formalizations of abduction also differ in the literature in the notions of preferred explanations: In this setting, the aim is to compute an explanation that is minimal among all explanations according to some criteria (e.g., inclusion or cardinality). A good overview is given by Eiter and Gottlob [1995]. However we are not concerned here with these different notions; we indeed focus on the decision problem asking whether there exists an explanation at all for a given instance. This problem is from now on denoted ABDUCTION.

This problem is well studied from the computational complexity perspective [Eshghi, 1993; Selman and Levesque, 1990; Eiter and Gottlob, 1995; del Val, 2000b; 2000a; Zanuttini, 2003], although with different formalizations of the problem, as mentioned above. It has been proved that it is Σ_2^P -complete in the general case [Eiter and Gottlob, 1995], while propositional deduction is known to be “only” coNP-complete [Cook, 1971]. This negative result raises the problem of identifying restricted cases in which ABDUCTION has computational complexity lower than the general case.

The most natural way to study such restrictions is to study restrictions on the theories representing knowledge bases. This is also the approach followed in most of the previous research in the area. For example, it is known that when the knowledge base T is a conjunction of Horn clauses, then ABDUCTION is NP-complete [Selman and Levesque, 1990].

*Supported by the National Graduate School in Computer Science (CUGS), Sweden.

The ultimate goal of this line of research is to determine the complexity of every restricted special case of the problem. The first result of this type was proved by Schaefer [1978], who proved that the satisfiability problem of conjunctions of Boolean constraints over a fixed language is either in P or NP-complete, depending on the language. Recall the result due to Ladner [1975] stating that if $P \neq NP$, then there exist decision problems in NP that are neither in P nor NP-complete. Hence the existence of dichotomy theorems like Schaefer's cannot be taken for granted. Creignou *et al.*'s book [2001] surveys such results.

In this paper we completely classify the complexity of ABDUCTION in Schaefer's framework and exhibit a trichotomy. More precisely, we prove that ABDUCTION is:

- In P if the language is affine of width 2,
- Otherwise, NP-complete if the language is Horn, dual Horn, bijunctive or affine,
- Otherwise, Σ_2^P -complete.

As far as we know, the (only) polynomial case and the minimal NP-hard languages that we exhibit are all new results. Note that the only property that guarantees a polynomial algorithm is that of being affine of width 2, i.e., the case where the theory representing the domain is only a conjunction of constants and (in)equalities between two variables. Thus the problem is very hard as soon as sets of literals are allowed as hypotheses and manifestations (instead of atoms or sets of atoms as in, e.g., [Zanuttini, 2003]).

We do not consider directly the complexity of the search problem consisting of computing an explanation or asserting there is none. It is however easily seen that this problem is hard as soon as ABDUCTION is; as for languages that are affine of width 2, our proof that ABDUCTION is in P exhibits an efficient algorithm.

2 Preliminaries

The set of all n -tuples of elements from $\{0, 1\}$ is denoted by $\{0, 1\}^n$. Any subset of $\{0, 1\}^n$ is called an n -ary relation on $\{0, 1\}$. The set of all finitary relations over $\{0, 1\}$ is denoted by BR . A constraint language over $\{0, 1\}$ is an arbitrary finite set $\Gamma \subseteq BR$. Constraint languages are the way in which we impose restrictions on the knowledge bases for the ABDUCTION problem.

A constraint over the constraint language Γ is an application of a relation R in Γ to a tuple of variables, written $R(x_1, \dots, x_n)$ (possibly with repeated variables). An assignment m to the variables satisfies the constraint $R(x_1, \dots, x_n)$ if $(m(x_1), \dots, m(x_n))$ is a tuple in R . Such a satisfying assignment m is called a *model* of $R(x_1, \dots, x_n)$. A theory T over Γ is a conjunction of constraints over Γ ; a model m of T is an assignment that satisfies all its constraints simultaneously, denoted $m \models T$. If there is such a model, T is said to be *satisfiable*. Finally, a theory T entails a theory T' , written $T \models T'$, if every model of T is a model of T' .

The unary relations $F = \{(0)\}$ and $T = \{(1)\}$, which force the value of a variable to 0 and 1 respectively, have a special role for the ABDUCTION problem. We will often use the shorthand notation $\neg x$ and x to denote the constraints

$F(x)$ and $T(x)$ respectively. Constraint languages containing F and T will be of particular importance to us. Given a constraint language Γ , the *idempotent constraint language* corresponding to Γ is $\Gamma \cup \{F, T\}$ which is denoted by Γ^{id} .

Given a theory T , $Vars(T)$ denotes the set of all variables that occurs in T . Given a set of variables V , $Lits(V)$ denotes the set of constraints $\bigcup_{x \in V} \{F(x) \cup T(x)\}$ or, equivalently, the set of all literals formed upon the variables in V . The opposite of a literal ℓ is written $\bar{\ell}$. Given a set of literals L , their conjunction is denoted by $\bigwedge L$.

The ABDUCTION problem restricted to the finite constraint language Γ is denoted by $ABDUCTION(\Gamma)$ and is defined as follows; the problem is said to be *parameterized* by Γ .

Problem 2 (ABDUCTION(Γ)) An instance $\mathcal{P}$ of ABDUCTION(Γ) consists of a tuple (V, H, M, T) , where:

- V is a set of variables,
- $H \subseteq Lits(V)$ is the set of hypotheses,
- $M \subseteq Lits(V)$ is the set of manifestations, and
- T is a theory over Γ with $Vars(T) = V$.

The question is whether there exists an explanation for $\mathcal{P}$, i.e., a set $E \subseteq H$ such that $T \wedge \bigwedge E$ is satisfiable and $T \wedge \bigwedge E \models \bigwedge M$.

The size of $\mathcal{P} = (V, H, M, T)$ is the total number of occurrences of variables in it.

Recall the following standard restrictions on the constraint languages (see, e.g. [Creignou *et al.*, 2001]).

Definition 3 (restrictions on Γ)

- Γ is Horn if every relation R in Γ is the set of models of a CNF formula having at most one unnegated variable in each clause,
- Γ is dual Horn if every relation R in Γ is the set of models of a CNF formula having at most one negated variable in each clause,
- Γ is bijunctive if every relation R in Γ is the set of models of a CNF formula having at most two literals in each clause,
- Γ is affine if every relation R in Γ is the set of models of a system of linear equations over $GF(2)$, the field with two elements,
- Γ is affine of width 2 if every relation R in Γ is the set of models of a system of linear equations over $GF(2)$ in which each equation has at most 2 variables.

We emphasize that Γ is assumed to be finite in the definition of ABDUCTION(Γ). Moreover, it parameterizes the problem, and is not part of its input. Thus we can assume any convenient presentation of the relations in Γ is stored in a catalog; thus we will assume, e.g., that if Γ is Horn then a theory T over Γ is given as a CNF formula with at most one unnegated variable per clause.

We now recall Schaefer's result, which will be of great importance throughout the paper. Call $SAT(\Gamma)$ the problem of deciding whether a given theory over Γ is satisfiable. Schaefer completely classified the complexity of $SAT(\Gamma)$; we only report the result for idempotent constraint languages.

Theorem 4 ([Schaefer, 1978]) $\text{SAT}(\Gamma^{id})$ is in P if Γ is Horn, dual Horn, bijnunctive or affine. Otherwise it is NP-complete.

Finally, we assume that the reader is familiar with the basic notions of complexity theory, but we briefly recall the following. P is the class of decision problems solvable in deterministic polynomial time. NP is the class of decision problems solvable in nondeterministic polynomial time. $\Sigma_2^P = NP^{NP}$ is the class solvable in nondeterministic polynomial time with access to an NP-oracle. A problem is NP-complete (Σ_2^P -complete) if every problem in NP (Σ_2^P) is polynomial-time reducible to it. Throughout the paper we assume $P \neq NP \neq \Sigma_2^P$.

3 Polynomial case

The following proposition gives the only (maximal) polynomial case of $\text{ABDUCTION}(\Gamma)$. Note that, as mentioned in the introduction, its proof gives a polynomial algorithm that computes an explanation if there exists one.

Throughout the proof we write $(\ell = a)$ ($a \in \{0, 1\}$) for the linear equation $(x = a)$ if $\ell = x$, and for the linear equation $x = a \oplus 1$ if $\ell = \neg x$. The shorthand $(\ell = \ell')$ is used in the same manner and is equivalent to $(\ell \oplus \ell' = 0)$.

Proposition 5 If Γ is affine of width 2, then $\text{ABDUCTION}(\Gamma)$ is polynomial.

Proof Let $\mathcal{P} = (V, H, M, T)$ be an instance of $\text{ABDUCTION}(\Gamma)$, where Γ is affine of width 2. If M' is the set of all manifestations m such that $T \not\models m$, then obviously the explanations of $\mathcal{P}$ are exactly those of (V, H, M', T) . Since $T \models m$ can be decided efficiently with Gaussian elimination on $T \wedge (m = 0)$, we assume $M' = M$.

For every manifestation $m \in M$ write E_m for the set of literals $\{h \in H \mid T \models (h = m)\}$; once again every E_m can be computed efficiently with Gaussian elimination on $T \wedge (h \oplus m = 1)$ for every $h \in H$. We show that $\mathcal{P}$ has an explanation if and only if $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable and no E_m is empty. Since the satisfiability of $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ can be decided efficiently with again Gaussian elimination (on $T \wedge \bigwedge_{m \in M} \bigwedge_{h \in E_m} (h = 1)$), this will conclude the proof.

Assume first $\mathcal{P}$ has an explanation E . Then $T \wedge \bigwedge M$ is consistent; since for every $m \in M$ and $h \in E_m$ we have $T \models (h = m)$, we also have $T \wedge \bigwedge M \models T \wedge \bigwedge_{m \in M} \bigwedge E_m$, and thus $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable. Now we also have $\forall m \in M, T \models (\bigwedge E \rightarrow m)$. Since T is affine of width 2 it is bijnunctive, thus every clause entailed by T can be minimized into a bijnunctive one; since $T \wedge \bigwedge E$ is satisfiable and $T \not\models m$, the only possibility is a minimal clause of the form $h \rightarrow m$ with $h \in E$. But since T is affine this implies that $m \rightarrow h$ also is an implicate of it, and finally we have $T \models (h \oplus m = 0)$, which shows that E_m is nonempty. For more details we refer the reader to [Zanuttini and Hébrard, 2002]. Conversely, assume $T \wedge \bigwedge_{m \in M} \bigwedge E_m$ is satisfiable and no E_m is empty. Then since $(h \oplus m = 0) \models (h \rightarrow m)$ it is easily seen that $\bigwedge_{m \in M} \bigwedge E_m$ is an explanation for $\mathcal{P}$. $\square$

4 NP-complete cases

We now exhibit the NP-complete cases of $\text{ABDUCTION}(\Gamma)$. Since $T \wedge \bigwedge E \models \bigwedge M$ holds if and only if $T \wedge \bigwedge E \wedge \bar{\ell}$ is

unsatisfiable for every $\ell \in M$, the following result is obvious.

Lemma 6 If $\text{SAT}(\Gamma^{id})$ is in P , then $\text{ABDUCTION}(\Gamma)$ is in NP .

We first establish NP-completeness for particular languages. In Section 5 we will establish more general results.

Let $R_{\neg x \vee y} = \{(1, 1), (0, 1), (0, 0)\}$, i.e., the set of models of $\neg x \vee y$. Observe that $R_{\neg x \vee y}$ is both Horn and dual Horn.

Proposition 7 $\text{ABDUCTION}(\{R_{\neg x \vee y}\})$ is NP-complete.

Proof Membership in NP follows from Theorem 4 and Lemma 6. As regards hardness, we give a reduction from the NP-complete problem MONOTONE-SAT [Garey and Johnson, 1979], i.e., the satisfiability problem for CNF formulas where each clause contains either only positive literals or only negative literals. Let $\psi = \bigwedge_{i=1}^k N_i \wedge \bigwedge_{i=1}^\ell P_i$ be such a formula, where each N_i is a negative clause, written $N_i = \bigvee_{j=1}^{\nu_i} \neg x_i^j$, and every P_i is a positive clause written $P_i = \bigvee_{j=1}^{\pi_i} y_i^j$. We build the instance $\mathcal{P} = (V, H, M, T)$ of $\text{ABDUCTION}(R_{\neg x \vee y})$ where:

- $V = \{\gamma_i \mid i = 1, \dots, k\} \cup \{\delta_i \mid i = 1, \dots, \ell\} \cup \text{Vars}(\psi)$; $\neg \gamma_i$ will intuitively represent satisfaction of clause N_i and δ_i , that of clause P_i
- $T = \bigwedge_{i=1}^k \bigwedge_{j=1}^{\nu_i} (x_i^j \vee \neg \gamma_i) \wedge \bigwedge_{i=1}^\ell \bigwedge_{j=1}^{\pi_i} (\neg y_i^j \vee \delta_i)$; this encodes the implications $\neg x_i^j \rightarrow \neg \gamma_i$ and $y_i^j \rightarrow \delta_i$, i.e., the fact that N_i (P_i) is satisfied if at least one of the $\neg x_i^j$ (y_i^j), $j \in \{1, \dots, \nu_i\}$ ($j \in \{1, \dots, \pi_i\}$) is
- $H = \text{Lits}(\text{Vars}(\varphi))$
- $M = \bigwedge_{i=1}^k \neg \gamma_i \wedge \bigwedge_{i=1}^\ell \delta_i$.

Obviously enough, the theory T is over the language $\{R_{\neg x \vee y}\}$. Now it is easily seen that if ψ has at least one model, say m , then $E = \{\ell \mid m \models \ell\}$ is an explanation for $\mathcal{P}$, and that if $\mathcal{P}$ has an explanation E , then any assignment m to $\text{Vars}(\psi)$ with $\forall \ell \in E, m \models \ell$ is a model of ψ . $\square$

Similarly, we now prove that $\text{ABDUCTION}(\Gamma)$ is NP-complete if Γ is the singleton language containing only $R_{x \vee y} = \{(1, 1), (1, 0), (0, 1)\}$.

Proposition 8 $\text{ABDUCTION}(\{R_{x \vee y}\})$ is NP-complete.

Proof Since $R_{x \vee y}$ is dual Horn, membership in NP follows from Theorem 4 and Lemma 6. As for hardness, we give a reduction from MONOTONE-SAT (see the proof of Proposition 7), where positive clauses in an instance of this problem are restricted to contain at most two literals. Thus an instance of this problem is a formula of the form $\psi = \bigwedge_{i=1}^k N_i \wedge \bigwedge_{i=1}^\ell (y_i^1 \vee y_i^2)$, where the y_i^j 's are variables and every N_i is a negative clause written $N_i = \bigvee_{j=1}^{\nu_i} \neg x_i^j$. The NP-completeness of this restricted problem follows directly from Schaefer's result [1978].

Given an instance ψ of MONOTONE-SAT as above we build the instance $\mathcal{P} = (V, H, M, T)$ of $\text{ABDUCTION}(\{R_{x \vee y}\})$ where:

- $V = \{\gamma_i \mid i = 1, \dots, k\} \cup \text{Vars}(\psi)$; γ_i will intuitively represent satisfaction of clause N_i

- $T = \bigwedge_{i=1}^k \bigwedge_{j=1}^{\nu_i} (x_i^j \vee \gamma_i) \wedge \bigwedge_{i=1}^{\ell} (y_i^1 \vee y_i^2)$; clauses $(x_i^j \vee \gamma_i)$ encode the implications $\neg x_i^j \rightarrow \gamma_i$
- $H = \text{Lits}(\text{Vars}(\varphi))$
- $M = \bigwedge_{i=1}^k \gamma_i$.

We show that ψ has a model if and only if $\mathcal{P}$ has an explanation. Assume first that ψ has a model m ; then it is easily seen that $E = \{\ell \mid m \models \ell\}$ is an explanation for $\mathcal{P}$. Now assume $\mathcal{P}$ has an explanation E . Then from $\forall i = 1, \dots, k, T \wedge \bigwedge E \models \gamma_i$ it follows that for every $i = 1, \dots, k$ E contains at least one $\neg x_i^j$, and thus any assignment satisfying E satisfies every negative clause of ψ ; on the other hand, since $T \wedge \bigwedge E$ is satisfiable there is a model m of $T \wedge \bigwedge E$ that satisfies every positive clause of ψ , and this m thus satisfies ψ . $\square$

The following proposition can be shown with the same proof as Proposition 8 with all variables renamed.

Proposition 9 Let $R_{\neg x \vee \neg y} = \{(1, 0), (0, 1), (0, 0)\}$. $\text{ABDUCTION}(\{R_{\neg x \vee \neg y}\})$ is NP-complete.

We finally prove that $\text{ABDUCTION}(\Gamma)$ is NP-complete for a particular affine language. This will be achieved by reducing to it another important problem in nonmonotonic reasoning, namely the inference problem for propositional circumscription. A model $m = (m_1, \dots, m_n)$ of a formula φ is said to be a *minimal model* of φ if there is no model $m' = (m'_1, \dots, m'_n)$ of φ such that $m \neq m'$ and $\forall i = 1, \dots, n, m'_i \leq m_i$.

Durand and Hermann proved that the inference problem for propositional circumscription of affine formulas is coNP-complete. In the process, they proved the following theorem.

Theorem 10 ([Durand and Hermann, 2003]) *The problem of deciding whether there is a minimal model of a given affine formula φ that does not satisfy a given negative clause $(\neg q_1 \vee \dots \vee \neg q_n)$ ($\forall i = 1, \dots, n, q_i \in \text{Vars}(\varphi)$) is NP-complete.*

A careful reading of their proof shows that the theorem remains true even if the linear equations in the input affine formulas are all restricted to contain at most 6 variables. We thus define the language Γ_{6aff} to be the set of all k -ary affine relations with $k \leq 6$. Obviously, Γ_{6aff} is finite, which is necessary for problem $\text{ABDUCTION}(\Gamma_{6aff})$ to be well-defined.

Proposition 11 $\text{ABDUCTION}(\Gamma_{6aff})$ is NP-complete.

Proof Membership in NP follows from Theorem 4 and Lemma 6. As for hardness, let φ be a formula and $q_1, \dots, q_n \in \text{Vars}(\varphi)$. We show that the clause $(\neg q_1 \vee \dots \vee \neg q_n)$ is false in some minimal model of φ if and only if the abduction problem with $T = \varphi$, $M = \{q_1, \dots, q_n\}$, and $H = \{\neg x \mid x \in \text{Vars}(\varphi) \setminus \{q_1, \dots, q_n\}\}$ has an explanation, which will conclude by Theorem 10 and the above remark.

Assume first that $(\neg q_1 \vee \dots \vee \neg q_n)$ is false in a minimal model m of φ . Define E to be $\{\neg x_i \mid m \models \neg x_i\}$. Since $m \models \varphi$ by assumption and $m \models \bigwedge E$ by construction, $\varphi \wedge \bigwedge E$ is satisfiable. Now assume for sake of contradiction that there is m' satisfying $\varphi \wedge \bigwedge E \wedge (\neg q_1 \vee \dots \vee \neg q_n)$. Then since $m' \models \bigwedge E$ and E is negative we get $\forall x \in E, m'(x) \leq m(x)$; now for $x \in \text{Vars}(\varphi) \setminus \text{Vars}(E)$ we have by assumption

$m(x) = 1$ and thus $m'(x) \leq m(x)$ again. Finally, we have $\exists q_i, m'(q_i) = 0 < 1 = m(q_i)$, which contradicts the minimality of m . Thus $\varphi \wedge \bigwedge E \models (q_1 \wedge \dots \wedge q_n)$ and E is an explanation.

Conversely, assume that E is an explanation. Then $\varphi \wedge \bigwedge E$ is satisfiable; write m for one of its minimal models. By assumption the formula $\varphi \wedge \bigwedge E \wedge (\neg q_1 \vee \dots \vee \neg q_n)$ is unsatisfiable, thus $m \not\models (\neg q_1 \vee \dots \vee \neg q_n)$. We also have $m \models \varphi$ by assumption. Finally, assume for sake of contradiction that m is not a minimal model of φ , and let m' be such that $m' \models \varphi$, $m' \leq m$ and $m' \neq m$. Then since E is negative (because H is) and $m \models \bigwedge E$ we have $m' \models \bigwedge E$, thus $m' \models \varphi \wedge \bigwedge E$, which contradicts the minimality of m among the models of $\varphi \wedge \bigwedge E$. $\square$

5 Classification

We finally put together the results in the previous sections for obtaining our complete classification. The concept of a *relational clone* is central to our approach.

Definition 12 (relational clone) Let $\Gamma \subseteq BR$. The *relational clone* of Γ is written $\langle \Gamma \rangle$ and is the set of all relations that can be expressed using relations from $\Gamma \cup \{=\}$ ($=$ is the equality relation on $\{0, 1\}$), conjunction, and existential quantification.

Intuitively, the constraints over $\langle \Gamma \rangle$ are those which can be simulated by constraints over Γ .

The following result states that when studying the complexity of $\text{ABDUCTION}(\Gamma)$ it is enough to consider constraint languages that are relational clones.

Lemma 13 Let Γ be a finite constraint language and $\Gamma' \subseteq \langle \Gamma \rangle$ finite. Then $\text{ABDUCTION}(\Gamma')$ is polynomial-time reducible to $\text{ABDUCTION}(\Gamma)$.

Proof Let (V', H', M', T') be an instance of $\text{ABDUCTION}(\Gamma')$. By the definition of a relational clone there is a set of variables W disjoint from V' and a theory T_- over $\Gamma \cup \{=\}$ with $\text{Vars}(T_-) = V_- = V' \cup W$ and such that T' is logically equivalent to the formula $\exists W, T_-$. Since W is disjoint from V' there is no variable occurring in H' or M' and in W at the same time, and it is then easily seen that the abduction problem (V_-, H', M', T_-) has an explanation if and only if (V', H', M', T') has one. Now for every constraint $(x_i = x_j)$ ($i < j$) it is enough to replace x_j with x_i everywhere in V_-, T_-, M' and H' and to remove the constraint from T_- for obtaining a still equivalent instance (V, H, M, T) of $\text{ABDUCTION}(\Gamma)$, which concludes. $\square$

We can reduce even further the set of constraints languages to be considered, namely to idempotent ones.

Lemma 14 Let Γ be a finite constraint language. $\text{ABDUCTION}(\Gamma^{id})$ is polynomial time reducible to $\text{ABDUCTION}(\Gamma)$.

Proof Let $\mathcal{P} = (V, H, M, T)$ be an instance of $\text{ABDUCTION}(\Gamma^{id})$. We build an instance $\mathcal{P}' = (V, H', M', T')$ of $\text{ABDUCTION}(\Gamma)$ by removing every constraint $F(x)$ or $T(x)$ from T and adding it to H and M . It is then easy to see that (V, H', M', T') has an explanation if and only if (V, H, M, T) has one. $\square$

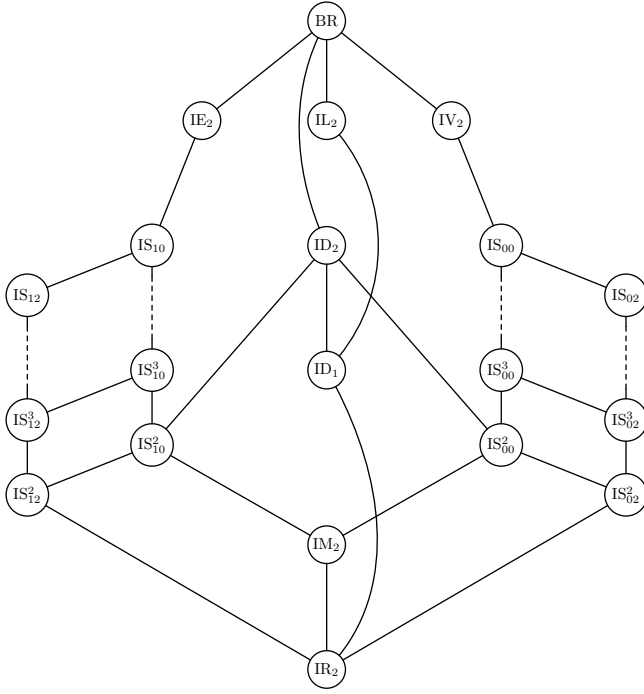


Figure 1: Lattice of all idempotent Boolean relational clones.

Given these two lemmas, our classification of the complexity of $\text{ABDUCTION}(\Gamma)$ heavily relies on Post's remarkable classification of all Boolean relational clones [Post, 1941]. Post proved in particular that the relational clones form a lattice under set inclusion. An excellent introduction to Post's lattice can be found in the recent survey articles [Böhler *et al.*, 2003; 2004].

Lemmas 13 and 14 say that for any finite $\Gamma' \subseteq \langle \Gamma^{id} \rangle$, $\text{ABDUCTION}(\Gamma')$ is polynomial-time reducible to $\text{ABDUCTION}(\Gamma)$. In other words, when studying the complexity of $\text{ABDUCTION}(\Gamma)$ it is enough to consider constraint languages that are idempotent relational clones. The lattice of all idempotent Boolean relational clones is given on Figure 1. Those that are most relevant to our classification are the following:

- BR , the set of all Boolean relations,
- IE_2 , the set of all Horn relations,
- IV_2 , the set of all dual Horn relations,
- ID_2 , the set of all bijunctive relations,
- IL_2 , the set of all affine relations,
- ID_1 , the set of all affine relations of width 2,
- IM_2 , the set of all relations that are Horn and dual Horn.

Thus, according to Post's lattice there is only one idempotent relational clone that is not Horn, not dual Horn, not affine, and not bijunctive, namely the relational clone consisting of all Boolean relations BR . Hence the following result follows intuitively from the result due to Eiter and Gottlob [1995] stating that ABDUCTION is Σ_2^P -complete for the general case of theories given by CNF formulas.

Proposition 15 *If Γ is not Horn, not dual Horn, not affine and not bijunctive then $\text{ABDUCTION}(\Gamma)$ is Σ_2^P -complete.*

Proof It is well-known that for any CNF formula ψ there is a set of variables W disjoint from $\text{Vars}(\psi)$ and a CNF formula ψ' over $\text{Vars}(\psi) \cup W$ with at most 3 variables per clause such that the formulas ψ and $\exists W \psi'$ are logically equivalent. That fact together with a proof similar to that of Lemma 13 show that the abduction problem for general CNF theories reduces to $\text{ABDUCTION}(\Gamma_3)$, where Γ_3 is the (finite) set of all ternary relations. Since Γ_3 is not Horn, not dual Horn, not affine and not bijunctive, we have $\langle \Gamma_3^{id} \rangle = BR$, and Lemma 13 and Lemma 14 concludes. $\square$

We are finally able to completely classify the complexity of $\text{ABDUCTION}(\Gamma)$.

Theorem 16 (classification) *Let Γ be a constraint language. $\text{ABDUCTION}(\Gamma)$ is:*

- In P if Γ is affine of width 2,
- Otherwise, NP-complete if Γ is Horn, dual Horn, bijunctive or affine,
- Otherwise, Σ_2^P -complete.

Proof Proposition 5 shows the result for languages that are affine of width 2. Now it can be seen that the relations $R_{\neg x \vee y}$, $R_{x \vee y}$ and $R_{\neg x \vee \neg y}$ of Propositions 7, 8 and 9 are in the relational clones IM_2 , IS_{02}^2 and IS_{12}^2 , respectively; this can be verified by checking that they are invariant under the operations defining the corresponding clones (for more details see [Böhler *et al.*, 2003; 2004]). Moreover, the language Γ_{aff} of Proposition 11 is affine, thus it is in IL_2 . Consequently, Figure 1 shows that the minimal idempotent relational clones that are not affine of width 2, namely IM_2 , IS_{02}^2 , IS_{12}^2 and IL_2 are NP-complete. On the other hand, we know from Theorem 4 and Lemma 6 that the relational clones IL_2 (affine), ID_2 (bijunctive), IE_2 (Horn) and IV_2 (dual Horn) are in NP. Thus $\text{ABDUCTION}(\Gamma)$ is NP-complete when $\langle \Gamma^{id} \rangle$ contains IM_2 , IS_{02}^2 , IS_{12}^2 or IL_2 and is contained in IL_2 , ID_2 , IE_2 , or IV_2 . This covers exactly the languages that are Horn, dual Horn, bijunctive, or affine and that are not affine of width 2.

Finally, Proposition 15 concludes the proof. $\square$

6 Discussion and future work

We have completely classified the complexity of propositional abduction in Schaefer's framework when manifestations and hypotheses are described by sets of literals. This result can prove useful in helping the designers of knowledge based systems to deal with the expressivity/tractability trade-off when choosing a language for their system. Our result indeed completes the picture of the complexity of reasoning for propositional constraint languages. In particular, we have shown that this problem is very hard, in the sense that only languages that are affine of width 2 allow for polynomial abduction. Also note that in many cases NP-hardness remains even when restricting further the problem; e.g., to $H = \text{Lits}(V \setminus \text{Vars}(M))$ (see the proofs of Propositions 7–9).

It is important to note that the complexity of abduction for a constraint language given in extension (i.e., by the set of all tuples in every relation) can be determined efficiently; the case of Theorem 16 in which a language falls can indeed be determined efficiently by using the closure properties of the concerned co-clones (see, e.g., [Böhler *et al.*, 2004]).

It would be interesting to try to extend this work into at least three directions. First of all, besides the problem of deciding the existence of an explanation for a given instance, of great importance are the problems of relevance and necessity, which ask whether a given hypothesis is part of at least one (resp. of all) preferred explanation(s). These problems involve a preference criterion which can have a great impact on their complexity; for more details we refer the reader to [Eiter and Gottlob, 1995]. Hence, it would be interesting to investigate the complexity of these problems. In the same vein, Eiter and Makino recently studied the problem of enumerating all the explanations of a Horn abduction problem [Eiter and Makino, 2002]; it would be interesting to try to extend their work to other classes of formulas.

Secondly, although Schaefer's framework is quite general, there are restrictions on propositional formulas that it cannot express. For instance, Eshghi [1993] and del Val [2000a] study such restrictions that yield polynomial cases of the abduction problem. It would thus be of great interest to try to identify still more such tractable classes.

Finally, it would be interesting to study the case where the domains of variables are more general, e.g., for conjunctions of constraints over finite domains.

References

- [Amilhastre *et al.*, 2002] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSPs - application to configuration. *Artificial Intelligence*, 135(1-2):199–234, 2002.
- [Böhler *et al.*, 2003] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter*, 34(4):38–52, 2003.
- [Böhler *et al.*, 2004] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part II: Constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35(1):22–35, 2004.
- [Buchler, 1955] J. Buchler. *Philosophical Writings of Peirce*. Dover, New York, 1955.
- [Bylander *et al.*, 1991] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49:25–60, 1991.
- [Console *et al.*, 1991] L. Console, D. Theseider Dupre, and P. Torasso. On the relationship between abduction and deduction. *J. Logic and Computation*, 1(5):661–690, 1991.
- [Cook, 1971] S. Cook. The complexity of theorem proving procedures. In *Proc. STOC'71*, pages 151–158, 1971.
- [Creignou *et al.*, 2001] N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of Boolean constraint satisfaction problems*. SIAM Monographs on Discrete Mathematics and Applications, 2001.
- [del Val, 2000a] A. del Val. The complexity of restricted consequence finding and abduction. In *Proc. AAAI'00*, pages 337–342, 2000.
- [del Val, 2000b] A. del Val. On some tractable classes in deduction and abduction. *Artificial Intelligence*, 116(1-2):297–313, 2000.
- [Durand and Hermann, 2003] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is coNP-complete. In *Proc. STACS'03*, pages 451–462, 2003.
- [Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. of the ACM*, 42(1):3–42, 1995.
- [Eiter and Makino, 2002] T. Eiter and K. Makino. On computing all abductive explanations. In *Proc. AAAI'02*, pages 62–67, 2002.
- [Eshghi, 1993] K. Eshghi. A tractable class of abduction problems. In *Proc. IJCAI'93*, pages 3–8, 1993.
- [Garey and Johnson, 1979] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
- [Hobbs *et al.*, 1993] J. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.
- [Ladner, 1975] R. Ladner. On the structure of polynomial time reducibility. *J. of the ACM*, 22:155–171, 1975.
- [Marquis, 2000] P. Marquis. *Handbook of Defeasible Reasoning and Uncertainty Management Systems (DRUMS)*, volume 5, chapter Consequence finding algorithms, pages 41–145. Kluwer Academic, 2000.
- [Morgan, 1971] C. Morgan. Hypothesis generation by machine. *Artificial Intelligence*, 2:179–187, 1971.
- [Pople, 1973] H. Pople. On the mechanization of abductive logic. In *Proc. IJCAI'73*, pages 147–152, 1973.
- [Post, 1941] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.
- [Schaefer, 1978] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC'78*, pages 216–226, 1978.
- [Selman and Levesque, 1990] B. Selman and H. Levesque. Abductive and default reasoning: A computational core. In *Proc. AAAI'90*, pages 343–348, 1990.
- [Stumptner and Wotawa, 2001] M. Stumptner and F. Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127:1–29, 2001.
- [Zanuttini and Hébrard, 2002] B. Zanuttini and J.-J. Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6):335–339, 2002.
- [Zanuttini, 2003] B. Zanuttini. New polynomial classes for logic-based abduction. *J. Artificial Intelligence Research*, 19:1–10, 2003.

Inverse Resolution as Belief Change

Maurice Pagnucco

ARC Centre of Excel. for Autonomous Sys.
School of Comp. Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia.
Email: morri@cse.unsw.edu.au

David Rajaratnam

School of Comp. Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia.
Email: daver@cse.unsw.edu.au

Abstract

Belief change is concerned with modelling the way in which a rational reasoner maintains its beliefs as it acquires new information. Of particular interest is the way in which new beliefs are acquired and determined and old beliefs are retained or discarded. A parallel can be drawn to symbolic machine learning approaches where examples to be categorised are presented to the learning system and a theory is subsequently derived, usually over a number of iterations. It is therefore not surprising that the term ‘*theory revision*’ is used to describe this process [Ourston and Mooney, 1994]. Viewing a machine learning system as a rational reasoner allows us to begin seeing these seemingly disparate mechanisms in a similar light.

In this paper we are concerned with characterising the well known *inverse resolution* operations [Muggleton, 1987; 1992] (and more recently, *inverse entailment* [Muggleton, 1995]) as AGM-style belief change operations. In particular, our account is based on the abductive expansion operation [Pagnucco *et al.*, 1994; Pagnucco, 1996] and characterised by using the notion of epistemic entrenchment [Gärdenfors and Makinson, 1988] extended for this operation. This work provides a basis for reconciling work in symbolic machine learning and belief revision. Moreover, it allows machine learning techniques to be understood as forms of non-monotonic reasoning.

The study of belief change is concerned with how a rational reasoner maintains its beliefs in the face of new information. As this new information is acquired it must be assimilated into the reasoner’s stock of beliefs. This requires that decisions be made as to which beliefs are retained, which are abandoned and which are incorporated by the reasoner. In this paper we shall consider belief change that adheres to the popular AGM [Gärdenfors, 1988] paradigm.

Symbolic machine learning deals with the generalisation of data for the purpose of classification and categorisation. Here we find the closely related notion of *theory revision* [Ourston and Mooney, 1994]. It too can be viewed as a form of belief change; one in which the aim is to assimilate the new information by generalising the reasoner’s belief corpus.

In this paper we investigate inductive logic programming as a form of belief change. In particular, we consider how the inverse resolution operators introduced by Muggleton [Muggleton, 1987; Muggleton and Buntine, 1988; Muggleton, 1992] can be re-constructed as belief change operators. Our strategy is to view the machine learning process as one in which the agent’s belief corpus represents its current theory. Examples are considered consecutively as new information that causes the agent to revise its belief corpus according to one of the operations. This is essentially the approach adopted in theory revision.

Since the inductive process that inverse resolution is trying to capture is an unsound form of logical inference (i.e., the conclusions do not *necessarily* follow from the premises) we require a form of belief change that allows for this type of inference. Unfortunately, the AGM operators for belief change do not cater for this inferential process. The problem stems from the fact that the inductive process requires the reasoner to formulate and accept hypotheses; hypotheses that go beyond the content carried by the new information (examples) and current belief corpus. As a result we use the notion of *abductive expansion* introduced by Pagnucco [Pagnucco *et al.*, 1994; Pagnucco, 1996] and formulate the required conditions to be satisfied by the epistemic entrenchment relation in order for the inverse resolution operators to be carried out.

The aim of this paper is to take a step towards reconciling the areas of symbolic machine learning and belief change. To that end, we show how the various inverse resolution operations—absorption, identification, inter-construction, intra-construction and truncation—can be viewed as AGM-style belief change operations. In particular, we show how the inverse resolution operations can be captured as forms of abductive expansion by providing the required restrictions on the epistemic entrenchment relation.

In the next section we briefly survey AGM belief change, abductive expansion and the inverse resolution operators. In Section 2 we provide the necessary conditions for each inverse resolution operation. Our approach is discussed in Section 3 and suggestions for future work provided in Section 4.

1 Background

We consider an underlying classical propositional logic with a propositional language $\mathcal{L}$, over a set of atoms, or propositional letters, $\mathbf{P} = \{a, b, c, \dots\}$, and truth-functional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. $\mathcal{L}$ also includes the truth-functional constants $\top$ and $\perp$. The closure operator C_n denotes classical

propositional consequence. We use the term *belief set* to refer to a set of formulas closed under Cn (i.e., $K = Cn(K)$). $\mathcal{K}$ represents the set of all belief sets. The distinguished element $K_{\perp} \in \mathcal{K}$ denotes the inconsistent belief set.

We shall also adopt the following linguistic conventions to simplify the presentation. Lower case Greek characters $\phi, \psi, \chi, \dots$ will represent arbitrary logical formulas. Lower case Roman characters $k, l, \dots$ will denote propositional literals. Upper case Roman characters $A, B, C, \dots$ denote propositional Horn clauses (disjunctions of literals containing at most one positive literal).

1.1 Belief Change

The mostly widely accepted approach to belief change is the one proposed by Alchourrón, Gärdenfors and Makinson [Alchourrón *et al.*, 1985; Gärdenfors, 1988; Hansson, 1999]. This approach is motivated by the concern to characterise *rational* belief change operators. That is, operations of belief change that are guided by principles of rationality. The AGM approach introduces three belief change operations: *expansion* (+), *contraction* (−), and *revision* (*). Expansion deals with the addition of new information without retraction of existing beliefs; contraction takes care of the removal of belief; and, revision deals with the addition of beliefs with the possibility of retracting current beliefs in order to maintain consistency. Each of these operations takes a belief set representing the reasoner's original belief corpus and a sentence representing the new information, returning the modified belief corpus: $(+, -, * : \mathcal{K} \times \mathcal{L} \rightarrow \mathcal{K})$. The belief change operations are characterised by rationality postulates and various constructions. The interested reader is referred to [Gärdenfors, 1988; Hansson, 1999] for further details.

In this paper we are concerned with the process of belief expansion as this is the typical setting under which inverse resolution is applied. This should not be seen as a significant restriction because belief revision proper can be obtained by combining contraction and expansion. However, AGM expansion is very limited in scope. It can be shown that AGM expansion corresponds to closure under logical deduction: $K + \phi = Cn(K \cup \{\phi\})$. What we require is a form of belief change that elaborates upon the newly acquired information. Such an operation has been introduced by Pagnucco [Pagnucco, 1996] in the form of an operation called *abductive expansion* ($\oplus : \mathcal{K} \times \mathcal{L} \rightarrow \mathcal{K}$). This operation looks to amplify the newly acquired information by trying to find an explanation for it (via abduction) and is reminiscent of the way that inverse entailment is defined [Muggleton, 1992]. Formally, abductive expansion may be defined as follows:

Definition 1.1 $K \oplus \phi$ is an *abductive expansion* of K with respect to ϕ iff

$$K \oplus \phi = \begin{cases} Cn(K \cup \{\psi\}) & \text{for some } \psi \in \mathcal{L} \text{ such that:} \\ & \text{(i) } K \cup \{\psi\} \vdash \phi \\ & \text{(ii) } K \cup \{\psi\} \not\vdash \perp \\ K & \text{if no such } \psi \text{ exists} \end{cases}$$

That is, when confronted with new information ϕ the reasoner seeks an hypothesis ψ that explains ϕ with respect to its current beliefs and incorporates this explanation into its corpus.

It can be shown that the operation of abductive expansion satisfies the following rationality postulates:

- (K⊕1) For any sentence ϕ and any belief set K ,
 $K \oplus \phi$ is a belief set
- (K⊕2) If $\neg\phi \notin K$, then $\phi \in K \oplus \phi$
- (K⊕3) $K \subseteq K \oplus \phi$
- (K⊕4) If $\neg\phi \in K$, then $K \oplus \phi = K$
- (K⊕5) If $\neg\phi \notin K$, then $\neg\phi \notin K \oplus \phi$
- (K⊕6) If $K \vdash \phi \leftrightarrow \psi$, then $K \oplus \phi = K \oplus \psi$
- (K⊕7) $K \oplus \phi \subseteq Cn(K \oplus (\phi \vee \psi) \cup \{\phi\})$
- (K⊕8) If $\neg\phi \notin K \oplus (\phi \vee \psi)$, then $K \oplus (\phi \vee \psi) \subseteq K \oplus \phi$

Postulate (K⊕1) maintains the expanded belief corpus as a belief set. (K⊕2) states that the new information should be included in the expanded belief state provided that it is consistent with the original corpus while (K⊕3) requires that the expanded corpus be at least as large as the original. When the new information is inconsistent with the corpus, (K⊕4) prevents change. Postulate (K⊕5) ensures that the corpus does not expand into inconsistency. (K⊕6) states that the expansion process is syntax insensitive. These postulates suffice to characterise abductive expansion as defined above. Often postulates (K⊕7) and (K⊕8) are added and restrict the mechanism that selects hypotheses to one that satisfies transitivity.

An elegant way of constructing an abductive expansion operation is by providing an ordering of *epistemic entrenchment* over the sentences in the language that indicates their plausibility or importance. For AGM operations, entrenchment orders the beliefs while the non-beliefs are relegated together as the least important sentences. For abductive expansion however, we need to be able to discriminate between potential hypotheses and so the traditional epistemic entrenchment ordering is modified so that beliefs are lumped together as the most important while the non-beliefs are ordered by importance in what we term an *abductive entrenchment ordering*.

Definition 1.2 An ordering $\leq$ over $\mathcal{L}$ is an *abductive entrenchment ordering* iff it satisfies (SEE1)–(SEE3) and condition (AE4):

- (SEE1) For any $\phi, \psi, \chi \in \mathcal{L}$, if $\phi \leq \psi$ and $\psi \leq \chi$ then $\phi \leq \chi$
- (SEE2) For any $\phi, \psi \in \mathcal{L}$, if $\{\phi\} \vdash \psi$ then $\phi \leq \psi$
- (SEE3) For any $\phi, \psi \in \mathcal{L}$, $\phi \leq \phi \wedge \psi$ or $\psi \leq \phi \wedge \psi$
- (AE4) When $K \neq K_{\perp}$, $\phi \in K$ iff $\psi \leq \phi$ for all $\psi \in \mathcal{L}$

This ordering is a total preorder (SEE1)–(SEE3) in which the beliefs are maximally entrenched (AE4); it effectively *rank*s the reasoner's non-beliefs.

The following properties of abductive epistemic entrenchment will be useful in proving the results in this paper.

Lemma 1.1 Suppose $\leq$ satisfies (SEE1)–(SEE3), then¹

1. For all $\phi \in \mathcal{L}$ either $\phi \leq \psi$ or $\neg\phi \leq \psi$ for all $\psi \in \mathcal{L}$.
2. For all $\phi \in \mathcal{L}$, $\{\psi : \phi \leq \psi\} = Cn(\{\psi : \phi \leq \psi\})$
3. $\phi \wedge \psi = \min(\phi, \psi)$
4. $\phi \vee \psi \geq \max(\phi, \psi)$

¹Note that $\phi < \psi$ iff $\phi \leq \psi$ and $\psi \not\leq \phi$ and $\phi = \psi$ iff $\phi \leq \psi$ and $\psi \leq \phi$.

The first property states that for every sentence ϕ , either ϕ or $\neg\phi$ (possibly both) is minimally entrenched. The second property notes that taking all sentences more entrenched than a certain rank gives a deductively closed set of sentences. The third property indicates that a conjunction is entrenched at the same level as the minimum of the conjuncts while the fourth property states that a disjunction is at least as entrenched as the maximally entrenched disjunct. These are well known properties of orderings satisfying conditions (SEE1)–(SEE3) which are termed *expectation orderings* by Gärdenfors and Makinson [Gärdenfors and Makinson, 1994].

Pagnucco [Pagnucco, 1996] gives a condition that allows us to determine an abductive expansion function $\oplus_{\leq}$ for a particular epistemic state K given an abductive entrenchment ordering $\leq$.

$$(C \oplus) \quad \psi \in K \oplus_{\leq} \phi \text{ iff either } \psi \in K \text{ or both } \neg\phi \notin K \text{ and } \phi \rightarrow \neg\psi < \phi \rightarrow \psi$$

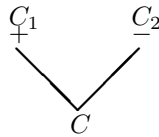
We omit the subscript $\leq$ unless necessary. This condition states that we can accept sentence ψ in the abductively expanded belief state whenever it is believed in the original corpus or when ψ is more plausible given ϕ than $\neg\psi$ given ϕ .

1.2 Inverse Resolution

Resolution is a valid inference procedure which deduces a clause C from two clauses C_1 and C_2 . Given a clause C_1 containing a literal l and a clause C_2 containing the literal $\neg l$, the resolvent of C_1 and C_2 , denoted $C = C_1.C_2$, is

$$C = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{\neg l\}) \quad (1)$$

This process may be visualised in the following diagram [Muggleton and Buntine, 1988]:²



Inverse resolution (and more recently, inverse entailment), on the other hand, is a machine learning technique based upon the following characterisation of inductive inference [Muggleton, 1987; 1989]. Given a partial domain theory Γ and a positive example E that is not a consequence of the domain theory ($\Gamma \not\models E$) we attempt to determine a new domain theory Γ' , using Γ and E , that will account for the example ($\Gamma' \models E$) and the original domain theory ($\Gamma' \vdash \Gamma$). If we think of Γ' as $\Gamma \cup I$ where I represents the result of inverse resolution, then the relationship with abduction should become much clearer. In practice, the domain theory and example are usually represented as Horn clauses. This technique is based on inverting the resolution process and consists of five operators: two V-operators, two W-operators and the truncation operator.

So as not to countenance invalid inference, the notion of an *oracle* is adopted. An oracle is an entity that accepts a clause, constructed using one of the inverse resolution operators, if it is valid in the intended model. In our framework abductive entrenchment is used to discriminate between potential hypotheses and takes the place of the oracle.

²The plus (+) (respectively minus (−)) sign in the diagram denotes that the literal resolved upon appears positive (respectively negative) in that clause.

V-Operators

Previously, a single resolution step was presented in terms of a “V”-shaped diagram. The two V-operators can derive one of the clauses at the top of the V given the other clause at the top of the V and the clause at the bottom of the V. The *absorption* operator derives C_2 given C_1 and C while the *identification* operator derives C_1 given C_2 and C .

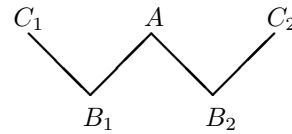
Since the new clause is constructed by finding the inverse of a resolved product, the notion of a resolved quotient³ of C and C_1 is defined [Muggleton and Buntine, 1988] as $C_2 = C/C_1$. Rearranging equation (1) for resolution can obtain $C_2 = (C \setminus (C_1 \setminus \{l\})) \cup \{\neg l\}$ under the following assumption [Muggleton and Buntine, 1988]:

- *Separability Assumption* — Clauses $C_1 \setminus \{l\}$ and $C_2 \setminus \{\neg l\}$ contain no common literals.

This assumption also simplifies the calculation of resolved quotients (i.e., absorptions or identifications).

W-Operators

Joining two resolution “V”s we obtain a form analogous to that for the V-operators [Muggleton and Buntine, 1988]:



In this situation a common literal l , contained in A , resolves with clauses C_1 and C_2 to produce B_1 and B_2 . Clauses B_1 and B_2 represent the new information and clauses A , C_1 and C_2 the constructed clauses. Interestingly, since l is resolved away, the constructed clauses A , C_1 and C_2 will contain a literal whose propositional symbol does not appear in either B_1 or B_2 . If l occurs negative in A then the operator is referred to as *intra-construction* and if it occurs positive in A the operator is called *inter-construction*.

Truncation

The *truncation* operator results from the special case where the empty clause occurs at the base of a V or W schemata. In a propositional system, this corresponds to dropping negative literals from a clause. In the first-order case Muggleton and Buntine [Muggleton and Buntine, 1988] show that two literals may be truncated by taking their least-general-generalisation. Rouveriol and Puget [Rouveriol and Puget, 1990] generalise this to a truncation operator which replaces terms by variables and drops literals from clauses.

In the case of a propositional language, a number of schema may be used to compute the required inverse resolution. These are displayed in Table 1 (see [Muggleton, 1987; 1989]).⁴ In each instance, the top line represents the initial clauses and the bottom line represents the constructed clause(s). We shall use these directly in our approach to characterise the inverse resolution operators as belief change operators (abductive belief expansion operators in fact).

³Absorption is considered here. Identification is similar.

⁴We adopt a slight renaming of the terms to those presented in [Muggleton, 1989], having found them more amenable to study. In the case of absorption and identification, the first clause on the top

Name	Rule
Absorption	$\frac{A \rightarrow k, A \wedge B \rightarrow j}{B \wedge k \rightarrow j}$
Identification	$\frac{B \wedge k \rightarrow j, A \wedge B \rightarrow j}{A \rightarrow k}$
Inter-Construction	$\frac{A \wedge B \rightarrow j, A \wedge C \rightarrow k}{B \wedge l \rightarrow j, C \wedge l \rightarrow k, A \rightarrow l}$
Intra-Construction	$\frac{A \wedge B \rightarrow j, A \wedge C \rightarrow j}{A \wedge l \rightarrow j, B \rightarrow l, C \rightarrow l}$
Truncation	$\frac{A \wedge B \rightarrow j}{A \rightarrow j}$

Table 1: Propositional inverse resolution operators.⁵

2 Approach

Our aim here is to render the individual inverse resolution operations as belief change operators. In particular, we shall give conditions on the abductive epistemic entrenchment that will guarantee that the results of an inverse resolution operation are included in the revised (or, more precisely, abductively expanded) belief corpus.

To make this more precise, we take the reasoner's belief corpus K and an example ϕ as the newly acquired information. We want to construct abductive expansion operations $\oplus$ for each of the five inverse resolution operations such that $K \oplus \phi$ represents the result of applying that particular inverse resolution operation on the given example and elements of the current belief corpus. This construction is achieved by specifying a condition(s) on abductive entrenchment that, when added to conditions (SEE1)–(SEE3) and (AE4), guarantees the result of the inverse resolution operation.

As noted above, abductive entrenchment encodes the 'choices' that are made by the oracle. In other words, entrenchment will be used to determine which potential hypotheses are accepted and which are rejected. Furthermore, it may be the case that in expanding the reasoner's belief corpus in this way, more than one potential hypothesis is accepted for a given example ϕ . This could be easily restricted to one hypothesis by imposing additional restrictions on the entrenchment ordering however, intuitively, this seems procrustean. Multiple generalisations allow for a more flexible characterisation of the inverse resolution process. Finally note that we will only be concerned with examples (new information) that are presented as Horn clauses in keeping with the way in which inverse resolution is traditionally realised.

2.1 Absorption

The typical case for Absorption occurs when $A \rightarrow k$ is believed and $A \wedge B \rightarrow j$ is presented as an example (newly acquired information). The Absorption operator looks for a sentence $B \wedge k \rightarrow j$ to add to the current theory in this case. By $(C\oplus)$, the condition we need to guarantee is therefore:

$$(A \wedge B \rightarrow j) \rightarrow \neg(B \wedge k \rightarrow j) < (A \wedge B \rightarrow j) \rightarrow (B \wedge k \rightarrow j)$$

line of the schemata is taken from the domain theory while the second represents the new data.

⁵Here A, B, C represent conjunctions of atoms while j, k, l represent atoms.

What we require here is a restriction on abductive entrenchment that, given $A \rightarrow k \in K$, guarantees this condition. Consider the following restriction:

$$(Abs) \quad \perp < B \wedge k \rightarrow j$$

It states that the sentence $B \wedge k \rightarrow j$ is not minimally entrenched (by Lemma 1.1(1), $\perp$ is minimally entrenched). In other words, when an example $A \wedge B \rightarrow j$ is presented, any hypothesis of the form $B \wedge k \rightarrow j$ that is not minimally entrenched and where $A \rightarrow k$ is believed, will be accepted. Now, by Lemma 1.1, condition (Abs) is equivalent to $\neg(B \wedge k \rightarrow j) < B \wedge k \rightarrow j$ which says that this hypothesis is preferred to its negation. The following theorem indicates the appropriateness of this condition.

Theorem 2.1 *Given a belief set $K \neq K_\perp$, a sentence $A \rightarrow k \in K$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Abs), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition $(C\oplus)$ gives the same result as Absorption (i.e., $B \wedge k \rightarrow j \in K \oplus (A \wedge B \rightarrow j)$).*

Proof: Now $(A \rightarrow k) \wedge (B \wedge k \rightarrow j) \vdash A \wedge B \rightarrow j$ so $(A \rightarrow k) \wedge (B \wedge k \rightarrow j) \leq A \wedge B \rightarrow j$ by (SEE2). But $B \wedge k \rightarrow j \leq A \rightarrow k$ by (AE4). It follows by Lemma 1.1(3) that $B \wedge k \rightarrow j = A \wedge B \rightarrow j$. Furthermore $(B \wedge k \rightarrow j) \wedge (A \wedge B \rightarrow j) = B \wedge k \rightarrow j$. But by (Abs) $\perp < (B \wedge k \rightarrow j) \wedge (A \wedge B \rightarrow j)$. Therefore $\neg((B \wedge k \rightarrow j) \wedge (A \wedge B \rightarrow j)) = \perp$ by Lemma 1.1(1) and so $\neg((B \wedge k \rightarrow j) \wedge (A \wedge B \rightarrow j)) < B \wedge k \rightarrow j$. It follows by Lemma 1.1(4) $\neg((B \wedge k \rightarrow j) \wedge (A \wedge B \rightarrow j)) < \neg(A \wedge B \rightarrow j) \vee (B \wedge k \rightarrow j)$. Hence by logical equivalence $(A \wedge B \rightarrow j) \rightarrow \neg(B \wedge k \rightarrow j) < (A \wedge B \rightarrow j) \rightarrow (B \wedge k \rightarrow j)$ as required. $\square$

Now it may seem strange that condition (Abs) does not mention the new example $A \wedge B \rightarrow j$. By Lemma 1.1(2) it can be shown that whenever $A \rightarrow k \in K$, it is always the case that $B \wedge k \rightarrow j \leq A \wedge B \rightarrow j$. That is, the new example cannot be less entrenched than any potential hypothesis.

However, we might re-consider the typical Absorption case and look at what happens when we assume that $A \wedge B \rightarrow j \in K$ and $A \rightarrow k$ is the new example. In this case we require condition (Abs) together with the following condition:

$$(Abs') \quad B \wedge k \rightarrow j \leq A \rightarrow k$$

The following result indicates this formally:

Theorem 2.2 *Given a belief set $K \neq K_\perp$, a sentence $A \wedge B \rightarrow j \in K$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Abs) and (Abs'), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition $(C\oplus)$ gives the same result as Absorption (i.e., $B \wedge k \rightarrow j \in K \oplus (A \rightarrow k)$).*

2.2 Identification

The typical case for Identification occurs when $B \wedge k \rightarrow j$ is believed and $A \wedge B \rightarrow j$ is presented as the new example. Identification returns $A \rightarrow k$ to be added to the belief corpus. The condition that we need to guarantee is, by $(C\oplus)$:

$$(A \wedge B \rightarrow j) \rightarrow \neg(A \rightarrow k) < (A \wedge B \rightarrow j) \rightarrow (A \rightarrow k)$$

Analogously to Absorption we require the following restriction on entrenchment to guarantee this condition.

$$(Ident) \quad \perp < A \rightarrow k$$

It states that any potential hypothesis $A \rightarrow k$ is not minimally

entrenched and by Lemma 1.1 is equivalent to saying that any potential hypothesis be preferred to its negation. This result is stated formally in the following theorem:

Theorem 2.3 *Given a belief set $K \neq K_\perp$, a sentence $B \wedge k \rightarrow j \in K$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Ident), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition (C $\oplus$) gives the same result as Identification (i.e., $A \rightarrow k \in K \oplus (A \wedge B \rightarrow j)$).*

The proof follows a similar idea to that for Absorption and so, due to lack of space, we omit it here.

Again, if we re-consider Identification and suppose that $A \wedge B \rightarrow j \in K$ while $B \wedge k \rightarrow j$ is the new example, then we also require the following condition on entrenchment:

$$(Ident') \quad A \rightarrow k \leq B \wedge k \rightarrow j$$

2.3 Inter-Construction

In the case of Inter-Construction there is only ever one case to consider because, without loss of generality, we may assume that sentence $A \wedge B \rightarrow j$ is believed and $A \wedge C \rightarrow k$ is presented as the new example. In this case three new clauses are derived ($B \wedge l \rightarrow j$, $C \wedge l \rightarrow k$, and $A \rightarrow l$) and we need to guarantee three conditions which are, by (C $\oplus$):

$$\begin{aligned} (A \wedge C \rightarrow k) \rightarrow \neg(B \wedge l \rightarrow j) &< (A \wedge C \rightarrow k) \rightarrow (B \wedge l \rightarrow j) \\ (A \wedge C \rightarrow k) \rightarrow \neg(C \wedge l \rightarrow k) &< (A \wedge C \rightarrow k) \rightarrow (C \wedge l \rightarrow k) \\ (A \wedge C \rightarrow k) \rightarrow \neg(A \rightarrow l) &< (A \wedge C \rightarrow k) \rightarrow (A \rightarrow l) \end{aligned}$$

Given that $A \wedge B \rightarrow j \in K$, the condition we require is:

$$(Inter) \quad \perp < (B \wedge l \rightarrow j) \wedge (C \wedge l \rightarrow k) \wedge (A \rightarrow l)$$

This condition can be read in a number of ways. Literally it says, in analogy to (Abs) and (Ident), that the conjunction of the potential hypotheses is not minimally entrenched. However, by Lemma 1.1(3), this also means that each of the individual hypotheses themselves is not minimally entrenched and so by Lemma 1.1(1) that they are each preferred to their negations. We now obtain the following result.

Theorem 2.4 *Given a belief set $K \neq K_\perp$, a sentence $A \wedge B \rightarrow j \in K$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Inter), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition (C $\oplus$) gives the same result as Inter-construction (i.e., $B \wedge l \rightarrow j$, $C \wedge l \rightarrow k$, $A \rightarrow l \in K \oplus (A \wedge C \rightarrow k)$).*

In inverse resolution, the literal l in the sentences above is newly introduced. It represents a concept that was not present in the original theory. The terms ‘predicate invention’ or ‘constructive induction’ are often used to describe what the W-operators are achieving by introducing a new element into the object language. Current theories of belief change do not allow for the expansion of the object language so we assume that the language is suitably equipped with a supply of propositions that can be used to create literals like l here. Of course, l will be present in any belief set; tautologies like $l \rightarrow l$ are present in every belief set. However, we can require additional conditions like $l \notin K$ if we want to ensure that l is newly generated. This is not an ideal solution as one is difficult to derive using belief sets but should go some way to achieving the desired results.

2.4 Intra-Construction

In Intra-construction, again without loss of generality, we may assume that $A \wedge B \rightarrow j \in K$ and $A \wedge C \rightarrow j$ is the new example. In this case, by (C $\oplus$), we need to guarantee the following three conditions:

$$\begin{aligned} (A \wedge C \rightarrow j) \rightarrow \neg(A \wedge l \rightarrow j) &< (A \wedge C \rightarrow j) \rightarrow (A \wedge l \rightarrow j) \\ (A \wedge C \rightarrow j) \rightarrow \neg(B \rightarrow l) &< (A \wedge C \rightarrow j) \rightarrow (B \rightarrow l) \\ (A \wedge C \rightarrow j) \rightarrow \neg(C \rightarrow l) &< (A \wedge C \rightarrow j) \rightarrow (C \rightarrow l) \end{aligned}$$

Again, the desired result is achieved by ensuring that none of the potential hypotheses is minimally entrenched.

$$(Intra) \quad \perp < (A \wedge l \rightarrow j) \wedge (B \rightarrow l) \wedge (C \rightarrow l)$$

The following theorem shows that this condition is correct.

Theorem 2.5 *Given a belief set $K \neq K_\perp$, a sentence $A \wedge B \rightarrow j \in K$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Intra), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition (C $\oplus$) gives the same result as Intra-construction (i.e., $A \wedge l \rightarrow j$, $B \rightarrow l$, $C \rightarrow l \in K \oplus (A \wedge C \rightarrow j)$).*

2.5 Truncation

For truncation, there are no requirements on the original belief corpus and all we assume is that $A \wedge B \rightarrow j$ is the new information. Truncation generalises this by adding $A \rightarrow j$ to the belief corpus. The condition we need guarantee is:

$$(A \wedge B \rightarrow j) \rightarrow \neg(A \rightarrow j) < (A \wedge B \rightarrow j) \rightarrow (A \rightarrow j)$$

This can be achieved by the following condition stating that $A \rightarrow j$ is not minimally entrenched.

$$(Trunc) \quad \perp < A \rightarrow j$$

The following theorem indicates that this is correct.

Theorem 2.6 *Given a belief set $K \neq K_\perp$ and an abductive epistemic entrenchment relation $\leq$ satisfying (Trunc), the abductive expansion operator $\oplus_{\leq}$ obtained from $\leq$ via condition (C $\oplus$) gives the same result as Truncation (i.e., $A \rightarrow j \in K \oplus (A \wedge B \rightarrow j)$).*

3 Discussion

The idea of using a form of belief expansion capable of introducing new hypotheses has also been suggested by Levi [Levi, 1991] who introduces an operation known as *deliberate expansion*. While his method of determining and selecting hypotheses differs from ours, the notion that belief expansion should allow for the addition of sentences that do not necessarily follow from the new information and the belief corpus is the same. The level to which the reasoner wishes to accept such hypotheses will be determined by their aversion to courting erroneous beliefs and this is reflected in the abductive entrenchment ordering in our framework. Such errors, when realised later, may of course be fixed by subsequent revision. We follow Levi’s *commensurability thesis* in maintaining that any form of belief revision can be obtained via a sequence of (abductive) expansions and contractions.

Forms of symbolic machine learning such as inverse resolution are often equated with *inductive* inference (or what

might be termed ‘*enumerative induction*’ [Harman, 1968]. However, here we have preferred the notion of abductive inference. There has been a substantial amount of debate in the artificial intelligence community as to what constitutes induction and abduction and whether they are essentially the same form of inference. We do not address this at any great length here. Levi [Levi, 1991] adopts Peirce’s later ideas on this subject where abduction is concerned with formulating potential hypotheses and induction selects the best of these. Abductive entrenchment to a large extent serves both purposes here.

One issue that is not strictly addressed by the inverse resolution operators but is important in the wider literature on symbolic machine learning is that of positive and negative examples. Clearly our framework and the inverse resolution operators deal quite well with positive examples. When it comes to negative examples this issue is not so clear. On the one hand new information like $\neg\phi$ can be dealt with quite easily. However, negative information of the form $\phi \notin K$, cannot be dealt with adequately here. It is possible to place restrictions on the entrenchment relation that will not permit ϕ to be believed but it is not clear how this information should be presented. On first sight it would appear that this is more an issue for iterated revision.

4 Conclusions and Future Work

In this paper we have developed an account of how the machine learning technique of inverse resolution, can be viewed as a form of belief change. In particular, we have given restrictions on epistemic entrenchment that will guarantee that the resulting belief change operations behave exactly in the manner of the inverse resolution operations. This is the first step in unifying the study of machine learning and belief change and has important repercussions for the links between machine learning and nonmonotonic reasoning.

This paper opens many avenues for future research in a number of interesting directions. Firstly, the development of belief change postulates that characterise the various inverse resolution operators. Such rationality conditions would allow another way of comparing these operations with belief revision. Perhaps more importantly, at this point it would be relatively straightforward to characterise each of the inverse resolution operators as a nonmonotonic consequence relation in the way of Kraus, Lehman and Magidor [Kraus *et al.*, 1990] allowing comparison with another large body of research.

In this work we have been able to capture each of the inverse resolution operators as a belief change operation. Given an example, we need to state which operation to apply in order to generalise the reasoner’s beliefs. A more general form of this theory would not require that the operation be stated but that a single belief change operation represent all inverse resolution operations and that it determine which are applied when new information is given. It is not clear however, how feasible such a *super-selecting* belief change operation is.

One important principle that guides belief change is that of *minimal change*; the reasoner’s belief corpus should be modified minimally to accept the new information. It would be interesting to see how this notion in belief change is manifested in work on inverse resolution and symbolic machine

learning and the heuristics and metrics applied there to select hypotheses for generalisation. Extending this work to the first-order case and investigating forms of iterated (abductive) belief change are also interesting and important possibilities.

References

- [Alchourrón *et al.*, 1985] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.
- [Gärdenfors and Makinson, 1988] P. Gärdenfors and D. Makinson. Revisions of knowledge systems using epistemic entrenchment. In *Proc. 2nd Conf. on Theor. Aspects of Reasoning About Know.*, pages 83–96, 1988.
- [Gärdenfors and Makinson, 1994] P. Gärdenfors and D. Makinson. Nonmonotonic inference based on expectations. *Artificial Intelligence*, 65:197–245, 1994.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.
- [Hansson, 1999] S. O. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer, 1999.
- [Harman, 1968] G. H. Harman. Enumerative induction as inference to the best explanation. *The Journal of Philosophy*, 65(18):529–533, 1968.
- [Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence*, 44:167–207, 1990.
- [Levi, 1991] I. Levi. *The Fixation of Belief and its Undoing*. Cambridge University Press, 1991.
- [Muggleton and Buntine, 1988] S. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proc. of the 5th Int. Machine Learning Workshop*, pages 339–352, 1988.
- [Muggleton, 1987] S. Muggleton. Duce, an oracle based approach to constructive induction. In *Proc. of the 10th Int. Joint Conf. on AI*, pages 287–292, 1987.
- [Muggleton, 1989] S. Muggleton. Inverting the resolution principle. In *Machine Intelligence 12*. Oxford, 1989.
- [Muggleton, 1992] S. Muggleton. *Inductive Logic Programming*. Academic Press, 1992.
- [Muggleton, 1995] S. Muggleton. Inverse entailment and Prolog. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.
- [Ourston and Mooney, 1994] D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:311–344, 1994.
- [Pagnucco *et al.*, 1994] M. Pagnucco, A. C. Nayak, and N. Y. Foo. Abductive expansion: The application of abductive inference to the process of belief change. In *Proc. 7th Aust. Joint Conf. on AI*, pages 267–274, 1994.
- [Pagnucco, 1996] M. Pagnucco. *The Role of Abductive Reasoning within the Process of Belief Revision*. PhD thesis, Basser Department of Computer Science, 1996.
- [Rouveirol and Puget, 1990] C. Rouveirol and J. F. Puget. Beyond inversion of resolution. In *Proc. of the 7th Int. Machine Learning Workshop*, pages 122–130, 1990.

Building the Semantic Web Tower from RDF Straw

Peter F. Patel-Schneider

Bell Labs Research

600 Mountain Ave, Murray Hill, NJ 07974, U. S. A.

pfps@research.bell-labs.com

Abstract

A same-syntax extension of RDF to first-order logic results in a collapse of the model theory due to logical paradoxes resulting from diagonalization. RDF is thus the wrong material for building the Semantic Web tower.

1 The Semantic Web

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation. [Berners-Lee *et al.*, 2001]

Thus the goals of the Semantic Web are very similar to the goals of Knowledge Representation—to provide means for representing information in a way that can be processed by machines.

The base language of the Semantic Web is RDF [Manola and Miller, 2004], a simple representation language based on a graph structure.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. [...] It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming. [<http://www.w3.org/2001/sw/>]

RDF is based on labelled graphs—an RDF knowledge base is (roughly) a graph where some of the nodes have labels (in the form of URI references and literal data) and the edges have labels (which are also URI references). Node labels are unique, i.e., a URI reference can be the label of at most one node.

The normative surface syntax for RDF is XML. This syntax is called RDF/XML [Dave Beckett, 2004]. There are other surface syntaxes for RDF, including N-triples [Jan Grant and Dave Beckett, 2004]. What counts in RDF is the graph that is conveyed, and not any other aspect of the surface syntax.

RDF, however, is more than just a graph language. It is actually a logic, with a model theoretic semantics and an entailment relationship. The details of all this have been specified in an W3C Recommendation [Hayes, 2004]. In essence, an

RDF graph can be read as a collection of facts corresponding to the edges in the graph (which are generally referred to as triples) where an edge with label *R* (the property) from node n_1 (the subject) to node n_2 (the object) corresponds to the fact $R(n_1, n_2)$. If n_1 (n_2) has a label then it is treated as a constant and its label is the name of the constant, if n_1 (n_2) doesn't have a label then it is treated as an existentially quantified variable with some arbitrary name. The model theory for RDF is slightly non-standard, to provide an integrated meaning for edge labels that are also node labels.

So in RDF the only way of providing information is via triples (graph edges). These triples are often written informally in a form very similar to N-triples. An edge with label *R* from a node labelled with the URI reference *A* to node labelled with the URI reference *B* is written as

A R B .

As URI references can be very long, names of the form *N:L* are very often used where the URI reference is constructed by concatenating the expansion of *N* to *L*. (The expansion is generally specified informally or left unspecified.) Nodes without labels are given “local tags” of the form $_l$. These tags are not URI references or constant names; they instead correspond to existentially quantified variables.

Because RDF has only triples, there is no way of providing syntactic constructs beyond what is specified above (i.e., binary facts, implicit conjunction, and existential quantification). Any other syntactic constructs have to be encoded as triples and their special meaning conveyed by stating semantic conditions in the model theory.

RDF has only a few of these semantic conditions. There is a particular property, *rdf:type*, which is considered to be the typing construct, i.e., the triple

A rdf:type B .

is interpreted as *A* has type *B* (or *A* belongs to the class *B*). There is a semantic condition that any name used as an edge label, i.e., as a property, must have type *rdf:Property*.

RDF is thus a very simple representation language. However, triples are a very powerful construct in that any ground or existentially quantified atomic fact can be encoded in RDF triples by using a new unnamed node with triples that link it to each of its arguments. This simple trick has been used in many other knowledge representation formalisms.

2 Extending the Semantic Web

The power of triples has given rise to a vision that RDF should be the universal language for the Semantic Web. This means that the syntax of all Semantic Web languages should be (RDF) triples and their meaning should be specified as semantic conditions that add to the RDF model theory. (This is often referred to as a same-syntax extension of RDF.)

This process was successfully carried out with RDF Schema [Dan Brinkley and R. V. Guha, 2004], an extension to RDF that provides simple vocabulary structuring facilities. RDF Schema interprets nodes labelled with particular URI references as providing these vocabulary structuring facilities. In particular, `rdfs:Class` is interpreted as the type of all types, so that if something is used as a type its types must include the interpretation of `rdfs:Class` (and objects of type `rdfs:Class` are called classes); and `rdfs:subClassOf` is interpreted as requiring a subtyping relationship between its arguments. These interpretations are provided by appropriate semantic conditions that extend the RDF model theory to the RDF Schema model theory [Hayes, 2004].

Problems first arose in this vision when the Semantic Web was extended to include an ontology language. The charter for the group charged with doing this extension included the requirement that the language “will be designed for maximum compatibility with XML and RDF language conventions” [<http://www.w3.org/2002/11/swv2/charters/WebOntologyCharter>]. This was interpreted for the working group as requiring that the resultant language be a same-syntax extension of RDF.

After considerable work, and some compromises, the resultant language, OWL [Dean *et al.*, 2004], was finally specified as a same-syntax extension of RDF [Patel-Schneider *et al.*, 2004]. Because the only syntax allowed is triples, complex OWL constructs (such as Description Logic [Baader *et al.*, 2003] restrictions) have to be encoded as a number of triples. For example, the Description Logic restriction that represents everything whose friends are lawyers ($\forall \text{ex:friend. ex:lawyer}$) is encoded as the triples

```
_:x rdf:type owl:Restriction .
_:x owl:onProperty ex:friend .
_:x owl:allValuesFrom ex:lawyer .
```

Because all triples are *facts*, entailment would not work correctly in OWL unless OWL interpretations required that there be facts corresponding to every possible piece of complex OWL syntax. (These requirements are often called *comprehension principles*.) Otherwise, it would not be the case that `ex:John` belonging to $\forall \text{ex:friend. ex:lawyer}$ entailed `ex:John` belonging to $\forall \text{ex:friend. (ex:lawyer} \sqcup \text{ex:doctor)}$, the restriction that represents everything that whose friends are lawyers or doctors. (Why? Because there might not be enough facts to encode the union of lawyers and doctors.)

The success of RDF Schema and OWL as same-syntax extensions of RDF has led to several calls to make another extension of the Semantic Web in a similar style to a first-order logic language.

What do I mean here by a “similar style”? Well, “style” is in the eye of the beholder, but some ground rules can be laid down. First, the syntactic constructs of the extension have to

be encoded as RDF triples. This means that, for example, encoding first order sentences as strings or Goedel numbers is not permitted. Second, the extension has to give meaning to every RDF graph. This means that the extension can’t identify a subset of RDF graphs (such as tree-like RDF graphs) as well-behaved, and only provide meaning for these graphs.

What then would an “RDF Logic” look like? It would provide enough constructs to nicely provide facilities for all of first-order logic (including predicates, constants, variables, conjunction, disjunction, negation, quantification, equality, functions, etc.). Its entailment relationship would, when restricted to the constructs that are used to provide the facilities of first-order logic, mirror entailment in first-order logic.

A language suitable for this purpose is binary predicate calculus with equality. (Recall that predicate calculus is function-free first-order logic without function symbols and equality [Barr and Feigenbaum, 1981, Volume 1, p. 163].) As indicated above, binary predicates can be used to encode predicates of arbitrary arity, also equality can be used to encode functions thus obtaining all the facilities of first-order logic.

3 RDF Logic

Our task is thus to design a same-syntax extension of RDF to binary predicate calculus with equality. We are defining “logical” interpretations, special RDF interpretations that provide a mechanism for logical constructs. We both add some new vocabulary and some new semantic conditions, in the same style as used in the RDF model theory [Hayes, 2004].

3.1 Equality

Equality is the easiest part of our task. All we need to do is to “anoint” a particular URI reference as the equality predicate by providing its semantic conditions. As OWL already has a notion of equality, we will use the URI reference that it uses for this purpose, which we will write, using the standard abbreviation, as `owl:sameAs`.

The semantic condition for equality is

Equality Condition:

$\langle x, y \rangle \in IEXT(I(\text{owl:sameAs}))$ iff $x = y$

This condition says that in any logical interpretation two domain elements are related by the relationship that is the property interpretation (I followed by $IEXT$)¹ of `owl:sameAs` precisely if they are the same.

3.2 Encoding Formulae

Because of the need to use only RDF syntax we have to have an encoding facility for predicate calculus formulae. There are many possible schemes to do so, but let’s pick one that

¹Here we see the slightly unusual nature of the RDF semantics. In a standard semantics for first-order logic we would go directly from the syntactic token `owl:sameAs` to its property interpretation. Here, instead, there is a preliminary mapping to an object in the domain of discourse (I) followed by a mapping to the property interpretation ($IEXT$). This means that properties are objects of the domain of discourse.

uses RDF as much as possible.² As in OWL, we are also going to need comprehension principles so that the formulae exist in all interpretations, so we might as well introduce them along with the syntax encoding.

We will use Ix:Formula as the class of formulae and Ix:Atomic as the class of atomic formulae. We can't just use RDF triples directly as atomic formulae because these formulae might not be true and they might have variables in them.

We require that atomic formulae be formulae, i.e.,

$$\text{Atomic Formulae Condition:} \\ ICEXT(I(\text{Ix:Atomic})) \subseteq ICEXT(I(\text{Ix:Formula}))$$

We are here using some terminology from the RDF Schema semantic conditions. $ICEXT(x)$, the class extension of x , is defined as those domain elements that have x as one of their types.

We will use Ix:predicate , Ix:subject , Ix:object , and Ix:subjectv and Ix:objectv to construct atomic formulae. In our syntax anything can be a variable, so we need to distinguish when something is being used as a variable or not and thus we need two variants of the subject and object links. We do not allow predicates to be variables, i.e., our logic is syntactically first-order, not higher-order.

Our first comprehension principles are concerned with atomic formulae, requiring that all atomic formulae exist in all logical interpretations. There are four different comprehension principles corresponding to the different ways variables can be used in atomic formulae.

$$\text{Atomic Formulae Comprehension Principles:} \\ \forall r \in IP \quad \forall x, y \in IR^3$$

$$\begin{aligned} \exists f_1 \in IR \quad & f_1 \in ICEXT(I(\text{Ix:Atomic})) \& \\ & \langle f_1, x \rangle \in IEXT(I(\text{Ix:subject})) \& \\ & \langle f_1, r \rangle \in IEXT(I(\text{Ix:predicate})) \& \\ & \langle f_1, y \rangle \in IEXT(I(\text{Ix:object})); \\ \exists f_2 \in IR \quad & f_2 \in ICEXT(I(\text{Ix:Atomic})) \& \\ & \langle f_2, x \rangle \in IEXT(I(\text{Ix:subjectv})) \& \\ & \langle f_2, r \rangle \in IEXT(I(\text{Ix:predicate})) \& \\ & \langle f_2, y \rangle \in IEXT(I(\text{Ix:object})); \\ \exists f_3 \in IR \quad & f_3 \in ICEXT(I(\text{Ix:Atomic})) \& \\ & \langle f_3, x \rangle \in IEXT(I(\text{Ix:subject})) \& \\ & \langle f_3, r \rangle \in IEXT(I(\text{Ix:predicate})) \& \\ & \langle f_3, y \rangle \in IEXT(I(\text{Ix:objectv})); \\ \exists f_4 \in IR \quad & f_4 \in ICEXT(I(\text{Ix:Atomic})) \& \\ & \langle f_4, x \rangle \in IEXT(I(\text{Ix:subjectv})) \& \\ & \langle f_4, r \rangle \in IEXT(I(\text{Ix:predicate})) \& \\ & \langle f_4, y \rangle \in IEXT(I(\text{Ix:objectv})). \end{aligned}$$

We will use a limited but complete set of connectives to construct composite formulae, so Ix:Negation , Ix:Conjunction , and Ix:Exists are our only classes of composite formulae. We will use Ix:left and Ix:right as our structural connectives for all composite formulae.

We require that composite formulae be formulae, i.e.,

Composite Formulae Conditions:

$$\begin{aligned} ICEXT(I(\text{Ix:Negation})) &\subseteq ICEXT(I(\text{Ix:Formula})) \\ ICEXT(I(\text{Ix:Conjunction})) &\subseteq ICEXT(I(\text{Ix:Formula})) \\ ICEXT(I(\text{Ix:Exists})) &\subseteq ICEXT(I(\text{Ix:Formula})) \end{aligned}$$

We also need the composite formulae to exist.

Negation Comprehension Principle:

$$\begin{aligned} \forall f \in ICEXT(I(\text{Ix:Formula})) \\ \exists n \in ICEXT(I(\text{Ix:Negation})) \\ \langle n, f \rangle \in IEXT(I(\text{Ix:left})) \end{aligned}$$

Conjunction Comprehension Principle:

$$\begin{aligned} \forall f, g \in ICEXT(I(\text{Ix:Formula})) \\ \exists j \in ICEXT(I(\text{Ix:Conjunction})) \\ \langle j, f \rangle \in IEXT(I(\text{Ix:left})) \& \langle j, g \rangle \in IEXT(I(\text{Ix:right})) \end{aligned}$$

Quantification Comprehension Principle:

$$\begin{aligned} \forall f \in ICEXT(I(\text{Ix:Formula})), \forall v \in IR \\ \exists q \in ICEXT(I(\text{Ix:Exists})) \\ \langle q, v \rangle \in IEXT(I(\text{Ix:left})) \& \langle q, f \rangle \in IEXT(I(\text{Ix:right})) \end{aligned}$$

We have now guaranteed that all finite non-circular formulae exist in all interpretations. Lots of other formulae or quasi-formulae might exist as well, of course, but (in keeping with the permissive nature of RDF) we won't rule them out.

3.3 Truth

So far all we have is the existence of formulae, not whether they are true or not. We now need to provide a mechanism for determining the truth of formulae in an interpretation. We start out doing this meta-theoretically and only later provide syntax for determining true formulae.

Because this is a first-order logic, i.e., it has variables, we need some mapping from variables to their values. Our mechanism needs to be slightly different from that in standard first-order logic because our variables are just elements of the domain of discourse.

Definition 1 (Variable mapping) Let $\mathcal{V}$ be the set of total mappings from IR to IR .

Now we can provide a meta-theoretic mapping that, given a formula and a variable mapping, assigns a truth value to the formula.

Definition 2 (Formula truth) Let $Bool$ be a total mapping from $IR \times \mathcal{V}$ into boolean values (true and false) such that $\forall V \in \mathcal{V}$

1. $\forall f \in ICEXT(I(\text{Ix:Atomic})), \forall x \in IR, \forall r \in IP, \forall y \in IR$
 $((\langle f, x \rangle \in IEXT(I(\text{Ix:subject})) \&$
 $\langle f, r \rangle \in IEXT(I(\text{Ix:predicate})) \&$
 $\langle f, y \rangle \in IEXT(I(\text{Ix:object}))) \rightarrow$
 $Bool(f, V) = \text{true} \text{ iff } \langle x, y \rangle \in IEXT(r)$
2. $\forall f \in ICEXT(I(\text{Ix:Atomic})), \forall x \in IR, \forall r \in IP, \forall y \in IR$
 $((\langle f, x \rangle \in IEXT(I(\text{Ix:subjectv})) \&$
 $\langle f, r \rangle \in IEXT(I(\text{Ix:predicate})) \&$
 $\langle f, y \rangle \in IEXT(I(\text{Ix:object}))) \rightarrow$
 $Bool(f, V) = \text{true} \text{ iff } \langle V(x), y \rangle \in IEXT(r)$
3. $\forall f \in ICEXT(I(\text{Ix:Atomic})), \forall x \in IR, \forall r \in IP, \forall y \in IR$
 $((\langle f, x \rangle \in IEXT(I(\text{Ix:subject})) \&$
 $\langle f, r \rangle \in IEXT(I(\text{Ix:predicate})) \&$
 $\langle f, y \rangle \in IEXT(I(\text{Ix:objectv}))) \rightarrow$
 $Bool(f, V) = \text{true} \text{ iff } \langle x, V(y) \rangle \in IEXT(r)$

²It really doesn't matter which one as long as formulae can be "manipulated" within the formalism itself.

³IP is the set of all RDF properties. IR is the set of all RDF domain elements, including properties.

4. $\forall f \in ICEXT(I(lx:Atomic)), \forall x \in IR, \forall r \in IP, \forall y \in IR$
 $(\langle f, x \rangle \in IEXT(I(lx:subjectv)) \&$
 $\langle f, r \rangle \in IEXT(I(lx:predicate)) \&$
 $\langle f, y \rangle \in IEXT(I(lx:objectv))) \rightarrow$
 $Bool(f, V) = true \text{ iff } \langle V(x), V(y) \rangle \in IEXT(r)$
5. $\forall f \in ICEXT(I(lx:Negation)), \forall x \in IR$
 $\langle f, x \rangle \in IEXT(I(lx:left)) \rightarrow$
 $Bool(f, V) = true \text{ iff } Bool(x, V) = false$
6. $\forall f \in ICEXT(I(lx:Conjunction)), \forall x \in IR, \forall y \in IR$
 $(\langle f, x \rangle \in IEXT(I(lx:left)) \&$
 $\langle f, y \rangle \in IEXT(I(lx:right))) \rightarrow$
 $Bool(f, V) = true \text{ iff }$
 $(Bool(x, V) = true \& Bool(y, V) = true)$
7. $\forall f \in ICEXT(I(lx:Exists)), \forall x \in IR, \forall y \in IR$
 $(\langle f, x \rangle \in IEXT(I(lx:left)) \&$
 $\langle f, y \rangle \in IEXT(I(lx:right))) \rightarrow$
 $Bool(f, V) = true \text{ iff } \exists w \in IR Bool(y, V') = true$
where $V'(x) = w, V'(z) = V(z)$ otherwise.

Note that this gives truth values to all elements of the domain of discourse, including non-formulae, but does not give any specification of what these truth values are for non-formulae. Note also that this does provide conditions on the truth values for over-specified formulae like a formula that is both a conjunction and a disjunction. Finally, note that this is not a recursive definition of truth values. Instead we just require that *Bool* satisfies the above conditions and rule out interpretations for which no such *Bool* exists; which amounts to almost the same thing but with fewer opportunities for paradoxes. For example, this method will work for infinite and circular formulae.

Now we need some way of getting at the truth of formulae. We will use *lx:True* to do so, providing the appropriate semantic condition so that true formulae belong to its class extension:

Truth Condition:
 $f \in ICEXT(I(lx:True)) \text{ iff}$
 $\exists V \in \mathcal{V} \quad Bool(f, V) = true$

This looks as if $ICEXT(I(lx:True))$ is defined somehow after *Bool*, but this is not really the case, as *Bool* depends on $ICEXT(I(lx:True))$ for formulae that use *lx:True*.

A *logical* interpretation is then simply an RDF interpretation that satisfies the above semantic conditions, including having a *Bool* mapping, and *logical* entailment is defined in the obvious way.

3.4 The Result

I claim that this is about as good a version of RDF Logic as one can get. I would have liked to make the following claim:

Claim 1 *For every logical interpretation, L , there is a (standard) predicate calculus with equality interpretation, P , such that for every predicate calculus sentence, S , i.e., a predicate calculus formula with no free variables, for every domain element, d , in L that corresponds to S (in the obvious way) S is true in P iff $d \in ICEXT(I(lx:True))$.*

This could be then used to show a strong correspondence between RDF Logic and predicate calculus with equality.

Unfortunately, this claim is *not* true. Ian Horrocks [Horrocks and Patel-Schneider, 2003] has pointed out that there are sentences whose truth is affected by the unusual nature of the RDF semantics. (Actually Ian's observation was done in the context of SCL (<http://www.ihmc.us/users/phayes/CL/>), but SCL has similar semantic features to those of RDF.) For example, in an RDF interpretation that has only one domain element, there is only *one* predicate, no matter what name.

This doesn't matter much in RDF and RDF Schema because neither can place conditions on the size of the domain. However, such conditions can be easily stated in predicate calculus with equality. For example, the formula

$$(\forall x \forall y x = y) \rightarrow (ex:P(ex:a, ex:a) \leftrightarrow ex:Q(ex:a, ex:a))$$

is not a theorem of predicate calculus with equality but every encoding of it is true in every logical interpretation.

So I can only claim that this version of RDF Logic is as good as any other. Further, I claim that this version of RDF Logic is as paradox-free as any other.

4 Self-Reference and Diagonalization

Does the above encoding actually work? Well, in a word, *no*. It falls prey to one of the (many) paradoxes related to the truth predicate, first noticed by Tarski [Tarski, 1933].

As in OWL, we have only provided comprehension principles for tree-like formulae. This means we don't require the existence of self-referential sentences like the Liar's sentence ("I am lying" or "This sentence is false"), which would here be encoded as

$_lx \text{ rdf:type } lx:Negation .$
 $_lx \text{ lx:left } _lx .$

If this construction had to exist in every logical interpretation then there would be no logical interpretations. (Try to determine whether $I(_lx) \in ICEXT(I(lx:True))$.) We have thus lost something of the free-form flavour of RDF, as some self-referential sentences are useful and we can't entail them, but otherwise our model theory have have easily been shown to break down.

We have also been careful not to break down on self-referential sentences if they do occur. For example, if we had *defined* *Bool* in a recursive manner starting with atomic formulae, then we would have had problems with circular and infinite formulae.

We have thus been fairly careful not to get hung up on the obvious problems. However we still have enough power in our formalism for it to break down. This is actually not a surprising result—just about any encoding of first-order logic falls prey to the problem, but our minimalist RDF Logic is weaker in some ways than the standard first-order logics in which this result is usually shown.

What we are going to do next is to show how our encoding breaks down by building a self-referential sentence using a process of diagonalization. We are proceeding in a somewhat similar fashion as done by Goedel in his diagonalization lemma.

4.1 Building Up Formulae

Throughout this process we will need to get ahold of composite syntactic structures. The formulae for this turn out to be

fairly long, so we will introduce some abbreviations. These are *only* abbreviations and thus don't add any power.

First, an abbreviation for a formula that checks for a single triple.

Definition 3 Let s , p , and o be URI references, and let sv and ov be boolean values. Then **require**(s,sv,p,o,ov) is a formula that looks like

```
_:r rdf:type lx:Atomic .
_:r lx:predicate p .
_:r lx:subjectv s .    if sv = true
_:r lx:subject s .    if sv = false
_:r lx:objectv o .    if ov = true
_:r lx:object o .    if ov = false
```

Next, an abbreviation for a formula that checks for a collection of triples all with the same subject.

Definition 4 Let s be a URI reference (used for the subject); let t be a URI reference (used for the type of the subject); let $p_1, a_1, p_2, a_2, \dots, p_n, a_n$ be URI references (used for predicates and values); let $q_1, v_1, q_2, v_2, \dots, q_m, v_m$ be URI references (used for predicates and variables). Then **construct**($s;t;p_1,a_1,\dots,p_n,a_n;q_1,v_1,\dots,q_m,v_m$) is a formula that is a conjunction of

```
require(s,true,rdf:type,t,false)
require(s,true,p1,a1,false) ...
require(s,true,pn,an,false)
require(s,true,q1,v1,true) ...
require(s,true,qm,vm,true)
```

This doesn't actually construct anything, of course. Neither does it require the existence of anything. However, for any domain element v such that

```
v ∈ ICEXT(I(t))
⟨v, I(a1)⟩ ∈ IEXT(I(p1)) ...
⟨v, I(an)⟩ ∈ IEXT(I(pn))
⟨v, v1⟩ ∈ IEXT(I(q1)) ...
⟨v, vm⟩ ∈ IEXT(I(qm))
```

then **construct**($s;t;p_1,a_1,\dots,p_n,a_n;q_1,v_1,\dots,q_m,v_m$) is true under any variable mapping that maps s to v and v_i to v_i , $1 \leq i \leq m$. Note especially that because all the non-circular formulae exist, from the formula comprehension principles, therefore any non-circular formula "built" in this way will exist.

4.2 Diagonalization

Diagonalization of a formula with one free variable is the application of that formula to itself, i.e., the diagonalization of $A(x)$ is $A('A(x)')$, where the quotes are not string quotes, but instead represent some encoding process. Diagonalization is part of the process of creating a self-referential structure.

Definition 5 Let s and g be URI references (used as variable names) Then **diag**(s,g) is a formula of the form

```
_:e1 rdf:type lx:Exists .
_:e1 lx:left s1 .
_:e1 lx:right _:e2 .
_:e2 rdf:type lx:Exists .
_:e2 lx:left s2 .
```

```
_:e2 lx:right _:c1 .
_:c1 rdf:type lx:Conjunction .
_:c1 lx:left construct(s;lx:Exists;
                    lx:left,x;lx:right,s1) .

_:c1 lx:right c2 .
_:c2 rdf:type lx:Conjunction .
_:c2 lx:left construct(s1;lx:Conjunction;
                    lx:left,s2;lx:right,g) .
_:c2 lx:right construct(s2;lx:Atomic;
                    lx:subjectv,x,
                    lx:predicate,owl:sameAs;
                    lx:object,g) .
```

diag(s,g) is true under a variable mapping V precisely when there are some domain elements s_1 and s_2 such that the variable mapping of s , $V(s)$, is a domain element with the following characteristics

```
V(s) ∈ ICEXT(I(lx:Exists))
⟨V(s), I(x)⟩ ∈ IEXT(I(lx:left))
⟨V(s), s1⟩ ∈ IEXT(I(lx:right))
s1 ∈ ICEXT(I(lx:Conjunction))
⟨s1, s2⟩ ∈ IEXT(I(lx:left))
⟨s1, V(g)⟩ ∈ IEXT(I(lx:right))
s2 ∈ ICEXT(I(lx:Atomic))
⟨s2, I(x)⟩ ∈ IEXT(I(lx:subjectv))
⟨s2, I(owl:sameAs)⟩ ∈ IEXT(I(lx:predicate))
⟨s2, V(g)⟩ ∈ IEXT(I(lx:object))
```

Informally, if g is a formula with only x free, which we could write $g(x)$, then if **diag**(s,g) then s is the formula $g('g(x)')$.

4.3 A Paradoxical Formula

Now consider any logical interpretation. All non-circular finite formula exist in this interpretation, so, in particular there is a formula with the following characteristics. (Here I am being a bit sloppy, using triples instead of a metatheoretical formulation, but this does not introduce any problems.)

```
_:p rdf:type lx:Exists .
_:p lx:left y .
_:p lx:right _:p1 .
_:p1 rdf:type lx:Conjunction .
_:p1 lx:left _:p2 .
_:p1 lx:right _:d .
_:p2 rdf:type lx:Atomic .
_:p2 lx:subjectv y .
_:p2 lx:predicate owl:sameAs .
_:p2 lx:object _:d .
```

```
_:d rdf:type lx:Exists .
_:d lx:left x .
_:d lx:right _:d1 .
_:d1 rdf:type lx:Conjunction .
_:d1 lx:left diag(x,y) .4
_:d1 lx:right _:d2 .
_:d2 rdf:type lx:Negation .
_:d2 lx:left _:d3 .
_:d3 rdf:type lx:Atomic .
```

⁴That is, $_:d1$'s left conjunct is a formula that requires that x be a diagonalization of y .

```

_:d3 lx:subjectv x .
_:d3 lx:predicate rdf:type .
_:d3 lx:object lx:True .

```

Informally this is the formula $\exists x (x = D) \ \& \ D$ where D is $\exists y \text{diag}(y, x) \ \& \ \neg \text{True}(y)$.

Now what is the truth value of $_p$, i.e., is $_p$ in $\text{ICEXT}(I(lx:\text{True}))$? Well the truth value of $_p$ is precisely the truth value of $_d$ under a variable mapping that maps x to $_d$, i.e., $\text{Bool}(_d, x = _d)$.

But this is true exactly when there is some diagonalization of $_d$ whose truth value is *false* and false exactly when every diagonalization of $_d$ has truth value *true*. As the diagonalization of $_d$ is a vanilla formula (finite, non-circular) thus at least one exists in any interpretation.

But the truth value of any diagonalization of $_d$ is precisely the truth value of $_d$ under a variable mapping that maps x to $_d$, i.e., $\text{Bool}(_d, x = _d)$. So if $\text{Bool}(_d, x = _d) = \text{true}$ then it must be false, and if $\text{Bool}(_d, x = _d) = \text{false}$ then it must be true. Therefore no truth valuation for this formula can exist.

As a formula of this form exists in every interpretation, therefore there are no logical interpretations and our extension breaks down. Nothing in this argument depends on the details of the encoding and constructions so any extension falls prey to the same problem as long as it is possible to represent the diagonalization relationship.

5 Discussion

What happened?

Because the syntax of RDF is so impoverished we *had* to write formulae using facts—facts that have to be true everywhere. Then to state that a formula is true we *had* to use a truth predicate—no other solution is possible.⁵ As the resultant logic has sufficient power to specify a form of diagonalization a paradox exists in the logic, resulting in its collapse.

Is it possible to wiggle out of this dilemma?

Not really.

If strings were used for formulae and no string manipulation primitives were available, then there would be no problem, but I have already argued that such string encodings are not permissible. In any case, adding string manipulation (i.e., concatenation) to the language would bring back the paradox. Similar issues rule out using other encodings like Goedel numbers.

If our truth “predicate” was weakened, perhaps so that it treated itself specially, as in KIF (<http://logic.stanford.edu/kif/kif.html>), then it might be possible to wiggle out. However, our truth predicate forms the basis of our logic, and thus weakening it weakens our logic, which I claim is also not permissible.

This also rules out, for example, potential solutions related to overall incomplete reasoners like the CycL reasoner ([⁵OWL has a form of truth predicate as well—various OWL properties, including `owl:equivalentClass` and `rdf:type`, play the role of specialized truth predicates. OWL, however, does not have the same power as first-order logic, and thus might not fall prey to diagonalization paradoxes.](http://www.cyc.com/doc/handbook/oe/oe-handbook-</p>
</div>
<div data-bbox=)

`toc-opencyc.html`). It is not that the reasoning process of the solution is broken. Instead the fundamental model theory of

I see no way out of this dilemma, and so I claim:

Claim 2 *A paradox-free same-syntax extension of RDF to first-order logic is **not** possible.*

Note that this is a much stronger claim than

Claim 3 *A same-syntax extension of RDF to first-order logic is **not** desirable.*

In conclusion, a puff of philosophical wind is sufficient to knock over a Semantic Web tower built solely from RDF. A stronger tower needs better building materials.

References

- [Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation, and applications*. Cambridge University Press, 2003.
- [Barr and Feigenbaum, 1981] Avron Barr and Edward A. Feigenbaum, editors. *The Handbook of Artificial Intelligence*. William Kauffman, Los Altos, California, 1981.
- [Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.
- [Corcoran, 1983] John Corcoran, editor. *Logic, Semantics, Metamathematics, papers from 1923 to 1938*. Hackett Publishing Company, Indianapolis, 1983.
- [Dan Brinkley and R. V. Guha, 2004] RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, <http://www.w3.org/TR/rdf-schema>, 2004.
- [Dave Beckett, 2004] RDF/XML syntax specification (revised). W3C Recommendation, <http://www.w3.org/TR/rdf-syntax-grammar/>, 2004.
- [Dean *et al.*, 2004] Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language: Reference. W3C Recommendation, <http://www.w3.org/TR/owl-ref/>, 2004.
- [Hayes, 2004] Patrick Hayes. RDF semantics. W3C Recommendation, <http://www.w3.org/TR/rdf-mt/>, 2004.
- [Horrocks and Patel-Schneider, 2003] Ian Horrocks and Peter F. Patel-Schneider. Three theses of knowledge representation in the semantic web. In *The Twelfth International World Wide Web Conference*, pages 39–47. ACM Press, May 2003.
- [Jan Grant and Dave Beckett, 2004] RDF test cases. W3C Recommendation, <http://www.w3.org/TR/rdf-testcases/>, 2004.
- [Manola and Miller, 2004] Frank Manola and Eric Miller. RDF primer. W3C Recommendation, <http://www.w3.org/TR/rdf-primer/>, 2004.
- [Patel-Schneider *et al.*, 2004] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language: Semantics and abstract syntax. W3C Recommendation, <http://www.w3.org/TR/owl-semantics/>, 2004.
- [Tarski, 1933] A. Tarski. The concept of truth in the languages of the deductive sciences [in Polish]. *Prace Towarzystwa Naukowego Warszawskiego, Wydział III Nauk Matematyczno-Fizycznych*, 34, 1933. Reprinted in Zygmunt 1995, pages 13–172; expanded English translation in [Corcoran, 1983], pages 152–278.

Measuring conflict and agreement between two prioritized belief bases

Guilin Qi, Weiru Liu, David A. Bell

School of Computer Science

Queen's University Belfast

Belfast, BT7 1NN, UK

{G.Qi,W.Liu,DA.Bell}@qub.ac.uk

Abstract

In this paper, we investigate the relationship between two prioritized knowledge bases by measuring both the conflict and the agreement between them. First of all, a quantity of conflict and two quantities of agreement are defined. The former is shown to be a generalization of the Dalal distance. The latter are, respectively, a quantity of strong agreement which measures the amount of information on which two belief bases “totally” agree, and a quantity of weak agreement which measures the amount of information that is believed by one source but is unknown to the other. All three quantity measures are based on the *weighted prime implicant*, which represents beliefs in a prioritized belief base. We then define a degree of conflict and two degrees of agreement based on our quantity of conflict and the quantities of agreement. We also consider the impact of these measures on belief merging and information source ordering.

1 Introduction

In the belief revision and belief merging literature, the well-known Dalal distance known as the Hamming distance between interpretations [Dalal, 1988], plays a key role in the notion of minimal change. The Dalal distance between two interpretations models how many atoms are in conflict, so it measures only the quantity of conflict between them. Hunter has defined a degree of conflict between two knowledge bases based on the Dalal distance [Hunter, 2004].

In recent years, relationships between two knowledge bases have been defined by measures of information and contradiction. In [Konieczny *et al.*, 2003], a degree of contradiction and a degree of ignorance were defined and they can be used to order the sources of information. If a knowledge base has a high degree of contradiction and a low degree of ignorance, then it has a low order. In [Hunter, 2002], some compromise relations between two knowledge bases were defined according to the quantities of conflict information and total information in them.

In all the relationships described above, the quantity of conflict information between two knowledge bases was focused upon. However, in reality, when establishing the rela-

tionships between two agents, not only information in conflict, but also information in agreement should be considered. The quantities of conflict and agreement can affect each other. Considering two agents with low quantity of conflict between them, our perception of the degree of conflict between them will be further weakened if they have a lot in common. Furthermore, when two agents have no information in conflict, it is useful to consider the agreement between them.

We use two quantities of agreement; one is called the quantity of strong agreement which measures the information that both agents “totally” agree with, and the other is called the quantity of weak agreement which measures the information that is believed by one source but is *unknown* to the other. Both quantities will influence the degree of conflict, but their influences are different. Intuitively, the quantity of strong agreement will have more influence on the degree of conflict than the quantity of weak agreement. To illustrate, let us consider the following three knowledge bases: $B_1 = \{\phi, \psi\}$, $B_2 = \{\neg\phi, \psi\}$, and $B_3 = \{\neg\phi\}$. B_1 is in conflict with both B_2 and B_3 . B_1 and B_2 strongly agree on ψ , whilst B_1 only weakly agrees with B_3 on ψ . Clearly the degree of conflict between B_1 and B_2 should be smaller than that between B_1 and B_3 because there is a topic that both B_1 and B_2 agree upon. However, when defining the degree of conflict [Hunter, 2004], Hunter did not distinguish the influences of strong agreement and weak agreement. To accompany the degree of conflict, we can define a degree of strong agreement and a degree of weak agreement.

In this paper, we will measure the conflict and agreement between two *prioritized* knowledge bases, where the priority of a formula, based on necessity degrees, is handled in possibilistic logic. It is well-known that priority plays an important role in inconsistency handling, belief revision and belief merging [Gärdenfors, 1988; Konieczny & Ramon, 1998; Benferhat *et al.*, 2002]. Possibilistic logic provides a good framework for expressing priority. We first define the *weighted prime implicant* (WPI), which is a generalization of the regular *prime implicant* to possibilistic logic. Then, the measures of conflict and agreement will be defined by WPIS.

This paper is organized as follows. Section 2 gives some preliminaries. We then define the weighted prime implicant and measures of conflict and agreement in Section 3. Section 4 discusses the impact of the measures of conflict and agreement. Finally we conclude the paper in Section 5.

2 Preliminaries

Classical logic: In this paper, we consider a propositional language $\mathcal{L}_{PS}$ from a finite set PS of propositional symbols. The classical consequence relation is denoted as $\vdash$. An interpretation is a total function from PS to $\{true, false\}$. Ω is the set of all possible interpretations. An interpretation w is a model of a formula ϕ iff $w(\phi) = true$. $p, q, r, \dots$ represent atoms in PS . A literal is an atom p or its negation $\neg p$. We will denote literals by $l, l_1, \dots$ and formulae in $\mathcal{L}_{PS}$ by $\phi, \psi, \gamma, \dots$. For each formula ϕ , we use $M(\phi)$ to denote its set of models. A classical knowledge base B is a finite set of propositional formulae. B is consistent iff there exists an interpretation w such that $w(\phi) = true$ for all $\phi \in B$. A clause C is a disjunction of literals: $C = l_1 \vee \dots \vee l_n$ and its dual clause, or term D , is a conjunction of literals: $D = l_1 \wedge \dots \wedge l_n$.

Possibilistic Logic [Dubois et al., 1994]: Possibilistic logic is a weighted logic where each classical logic formula is associated with a number in $[0, 1]$, a necessity degree, which represents the extent to which the formula is true. A possibilistic knowledge base (PKB) is the set of possibilistic formulae of the form $B = \{(\phi_i, \alpha_i) : i = 1, \dots, n\}$. Possibilistic formula (ϕ_i, α_i) means that the necessity degree of ϕ_i is at least equal to α_i . A classical knowledge base $B = \{\phi_i : i = 1, \dots, n\}$ corresponds to a PKB $B' = \{(\phi_i, 1) : i = 1, \dots, n\}$. In this paper, we consider only PKBs where every formula ϕ is a classical propositional formula. The classical base associated with B is denoted as B^* , namely $B^* = \{\phi_i | (\phi_i, \alpha_i) \in B\}$. A PKB is consistent iff its classical base is consistent.

Semantically, the most basic and important notion is *possibility distribution* $\pi: \Omega \rightarrow [0, 1]$. $\pi(\omega)$ represents the possibility degree of interpretation ω with available beliefs. From *possibility distribution* π , two measures can be determined, one is the possibility degree of formula ϕ , defined as $\Pi(\phi) = \max\{\pi(\omega) : \omega \models \phi\}$, the other is the necessity degree of formula ϕ , defined as $N(\phi) = 1 - \Pi(\neg\phi)$. The possibility measure is max-decomposable, i.e. $\Pi(\phi \vee \psi) = \max(\Pi(\phi), \Pi(\psi))$. Whilst the necessity measure is min-decomposable, i.e. $N(\phi \wedge \psi) = \min(N(\phi), N(\psi))$.

Definition 1 [Dubois et al., 1994] Let B be a PKB, and $\alpha \in [0, 1]$. The α -cut of B is $B_{\geq \alpha} = \{\phi \in B^* | (\phi, a) \in B \text{ and } a \geq \alpha\}$.

Definition 2 [Dubois et al., 1994] Let B be a PKB. A formula ϕ is said to be a possibilistic consequence of B to degree a , denoted by $B \vdash_{\pi}(\phi, a)$, iff the following conditions hold: (1) $B_{\geq a}$ is consistent; (2) $B_{\geq a} \vdash \phi$; (3) $\forall b > a, B_{\geq b} \not\vdash \phi$.

3 Measures of Conflicts and Agreements

3.1 Weighted prime implicant

In this section, we will define and discuss weighted prime implicants of PKBs.

A term D is an implicant of formula ϕ iff $D \vdash \phi$ and D does not contain two complementary literals.

Definition 3 [Cadoli & Donini, 1997] A prime implicant of knowledge base B is an implicant D of B such that for every other implicant D' of B , $D \not\vdash D'$.

Prime implicants are often used in knowledge compilation [Cadoli & Donini, 1997] to make the deduction tractable. Suppose $D_1, \dots, D_k$ are all the prime implicants of B , we have $B \vdash \phi$, for any ϕ iff for every prime implicant D_i , $D_i \vdash \phi$.

Now we define weighted prime implicants of a PKB. Let us first define weighted prime implicants for PKB $B = \{(\phi_1, a_1), \dots, (\phi_n, a_n)\}$ where ϕ_i are clauses. For a more general PKB, we can decompose it to an equivalent PKB whose formulae are clauses by the min-decomposability of necessity measures, i.e., $N(\bigwedge_{i=1,k} \phi_i) \geq m \Leftrightarrow \forall i, N(\phi_i) \geq m$.

Let $B = \{(\phi_1, a_1), \dots, (\phi_n, a_n)\}$ be a PKB where ϕ_i are clauses. A weighted implicant of B is $D = \{(\psi_1, b_1), \dots, (\psi_k, b_k)\}$, a PKB, such that $D \vdash_{\pi} B$, where ψ_i are literals. Let D and D' be two weighted implicants of B , D is said to be more specific than D' iff $D \neq D'$, $D^* \subseteq D'^*$ and $\forall (\psi_i, a_i) \in D, \exists (\psi_i, b_i) \in D'$ with $b_i \leq a_i$.

Definition 4 Let $B = \{(\phi_1, a_1), \dots, (\phi_n, a_n)\}$ be a PKB where ϕ_i are clauses. A weighted prime implicant (WPI) of B is D such that

1. D is a weighted implicant of B
2. $\nexists D'$ of B such that D is more specific than D' .

Let us look at an example to illustrate how to construct WPIs.

Example 1 Let $B = \{(p, 0.8), (q \vee r, 0.5), (q \vee \neg s, 0.6)\}$ be a PKB. The WPIs of B are $D_1 = \{(p, 0.8), (q, 0.6)\}$, $D_2 = \{(p, 0.8), (r, 0.5), (\neg s, 0.6)\}$, and $D_3 = \{(p, 0.8), (q, 0.5), (\neg s, 0.6)\}$.

The WPI generalizes the prime implicant.

Proposition 1 Let $B = \{(\phi_1, 1), \dots, (\phi_n, 1)\}$ be a PKB which contains formulae with weight 1, i.e., B is a classical knowledge base. Then D is a WPI of B iff D is a prime implicant of B .

However, given PKB B , if D is a WPI of B , then D^* is not necessarily a prime implicant of B^* . A counterexample can be found in Example 1, where D_3 is a WPI, but $D_3^* = \{p, q, \neg s\}$ is not a prime implicant of B^* .

The following proposition says that WPIs can be used to compile a PKB.

Proposition 2 Let B be a PKB. If $D_1, \dots, D_n$ are all the WPIs of B , then for any formula ϕ , we have,

$$B \vdash_{\pi}(\phi, a) \text{ iff } D_i \vdash_{\pi}(\phi, a), \text{ for all } D_i.$$

Next we give some justification for the WPI.

First of all, to measure information in a single classical knowledge base (this knowledge base may be inconsistent), most of the current methods are based on the *models* of the knowledge base [Hunter, 2002; Lozinskii, 1994]. In [Hunter, 2002], the degree of inconsistency is measured based on the *model* of an inconsistent knowledge base in the framework of quasi-classical logic. In [Lozinskii, 1994], a *quasi-model* of an inconsistent knowledge base, which is a maximal consistent subbase of the knowledge base, is defined to measure information for inconsistent sets. By Definition 4, each WPI can be interpreted as a partial truth assignment. Suppose p is an atom and D is a WPI, then $(p, a) \in D$ means

that there is an argument for p in D with certainty degree a , and $(\neg p, b) \in D$ means that there is an argument against p in D with certainty degree b , while $\phi \notin D^*$ means the truth value of ϕ is undetermined in D . By Proposition 2, a WPI can be viewed as a *partial model* of a possibilistic knowledge base. This is consistent with the methods in [Hunter, 2002; Lozinskii, 1994].

Second, when all the formulae in a PKB have the same weight 1, a WPI is a prime implicant. In classical logic, a classical model is often used to define the distance between two knowledge bases [Dalal, 1988]. However, classical models are not suitable for us to define the quantities of agreement between knowledge bases because a classical model must assign a truth value to every atom in the knowledge bases. Let us look at the example in the introduction again. The only model for B_1 is $w = \{\phi, \psi\}$ and there are two models for B_3 , i.e., $w_1 = \{\neg\phi, \psi\}$ and $w_2 = \{\neg\phi, \neg\psi\}$. B_1 and B_3 weakly agree on ψ because only B_1 supports ψ . However, by comparing w with w_1 or comparing w with w_2 we cannot get such a conclusion. In contrast, a prime implicant can be viewed as a *partial* truth assignment. That is, only some of the atoms are assigned truth values. Given a prime implicant D of B , a three-value semantics can be associated with it as follows:

$$v_D(p) = \begin{cases} \text{true} & \text{if } D \vdash p, \\ \text{false} & \text{if } D \vdash \neg p, \\ \text{undetermined} & \text{otherwise.} \end{cases} \quad (1)$$

In the example, B_1 has one prime implicant $D_1 = \{\phi, \psi\}$ and B_3 has one prime implicant $D_2 = \{\neg\phi\}$, where D_2 does not contain any information on ψ ; so the quantity of weak agreement between D_1 and D_2 is 1. As a consequence, the weak agreement between B_1 and B_3 is 1, which is consistent with our analysis above.

3.2 Quantity of conflict and quantities of agreement

In this section, we will measure the quantities of conflict and agreement between two PKBs based on the WPI.

First we define the quantity of conflict between two WPIs.

Definition 5 Let B_1 and B_2 be two PKBs. Suppose C and D are WPIs of B_1 and B_2 respectively, then the quantity of conflict between C and D is defined as

$$q_{Con}(C, D) = \sum_{(p,a) \in C \text{ and } (\neg p,b) \in D} \min(a, b). \quad (2)$$

When all the weights are 1, $q_{Con}(C, D)$ is the cardinality of the set of atoms which are in conflict in $C \cup D$.

Definition 6 Let B_1 and B_2 be two PKBs. Suppose C and D are the sets of WPIs of B_1 and B_2 respectively, then the quantity of conflict between B_1 and B_2 is defined as

$$Q_{Con}(B_1, B_2) = \min\{q_{Con}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (3)$$

The quantity of conflict between B_1 and B_2 measures the minimum amount of information in conflict between them.

Example 2 Let $B_1 = \{(\neg p, 0.8), (\neg q \vee r, 0.6)\}$ and $B_2 = \{(p \vee \neg r, 0.7), (q, 0.5)\}$ be two PKBs. The

WPIs of B_1 are $C_1 = \{(\neg p, 0.8), (\neg q, 0.6)\}$ and $C_2 = \{(\neg p, 0.8), (r, 0.6)\}$, and the WPIs of B_2 are $D_1 = \{(p, 0.7), (q, 0.5)\}$ and $D_2 = \{(\neg r, 0.7), (q, 0.5)\}$. It is easy to calculate that $q_{Con}(C_1, D_1) = 1.2$, $q_{Con}(C_1, D_2) = 0.5$, $q_{Con}(C_2, D_1) = 0.7$, $q_{Con}(C_2, D_2) = 0.6$. Therefore, the quantity of conflict between B_1 and B_2 is 0.5.

Proposition 3 Let B , B_1 and B_2 be three PKBs. If $B_1 \subseteq B_2$, then $Q_{Con}(B, B_1) \leq Q_{Con}(B, B_2)$.

Proposition 3 tells us that the quantity of conflict between two PKBs will increase (not strictly) when one of them has some new information added.

Let X be a set of classical propositional formulae. Let $I(X)$ be the set of interpretations of X delineated by the atoms used in X (i.e. $I(X) = 2^{Atoms(X)}$, where $Atoms(X)$ denotes the set of atoms appearing in X). Let $M(X, Y)$ be the set of **models** of X that are in $I(Y)$. That is, $M(X, Y) = \{w | w \models X \wedge w \in I(Y)\}$. The Dalal distance [Dalal, 1988] between two models w_i, w_j of a classical formula is the Hamming distance between them, i.e., $Dalal(w_i, w_j) = |w_i - w_j| + |w_j - w_i|$.

Proposition 4 Let B_1 and B_2 be two consistent classical knowledge bases. Let $Dalal(B_1, B_2) = \min\{Dalal(w_i, w_j) | w_i \in M(B_1, B_1 \cup B_2), w_j \in M(B_2, B_1 \cup B_2)\}$. Then we have

$$Q_{Con}(B_1, B_2) = Dalal(B_1, B_2).$$

Proposition 4 is very important, because it reveals that our quantity of conflict coincides with the Dalal distance in classical logic. Therefore, the quantity of conflict $Q_{Con}(B_1, B_2)$ can be taken as a generalization of the Dalal distance.

Definition 7 Let B_1 and B_2 be two PKBs. Suppose C and D are WPIs of B_1 and B_2 respectively, then the quantity of strong agreement between C and D is defined as

$$q_{SA}(C, D) = \sum_{(p,a) \in C, (p,b) \in D} \min(a, b), \quad (4)$$

When all the weights are 1, $q_{Con}(C, D)$ is the cardinality of the set of literals that are in both C and D .

Definition 8 Let B_1 and B_2 be two PKBs. Suppose C and D are the sets of WPIs of B_1 and B_2 respectively, then the quantity of strong agreement between B_1 and B_2 is defined as

$$Q_{SA}(B_1, B_2) = \max\{q_{SA}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (5)$$

The quantity of strong agreement between B_1 and B_2 measures how much information is supported by both B_1 and B_2 .

Example 3 (Continue Example 2) By Equation 4, we have $q_{SA}(C_1, D_1) = q_{SA}(C_1, D_2) = 0$ and $q_{SA}(C_2, D_1) = q_{SA}(C_2, D_2) = 0$. Therefore, the quantity of strong agreement between B_1 and B_2 is $Q_{SA}(B_1, B_2) = 0$.

Definition 9 Let B_1 and B_2 be two PKBs. Suppose C and D are WPIs of B_1 and B_2 respectively, then the quantity of weak agreement between C and D is defined as

$$q_{WA}(C, D) = \sum_{(p_i, a_i) \in C \cup D, p_i \notin C^* \cap D^* \text{ and } \neg p_i \notin C^* \cup D^*} a_i. \quad (6)$$

When all the weights are 1, $q_{Con}(C, D)$ is the cardinality of the set of literals which are in only one of C or D .

Definition 10 Let B_1 and B_2 be two PBKs. Suppose C and D are the sets of WPIs of B_1 and B_2 respectively, then the quantity of weak agreement between B_1 and B_2 is defined as

$$Q_{WA}(B_1, B_2) = \max\{q_{WA}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (7)$$

The quantity of weak agreement between B_1 and B_2 measures the information supported by only one knowledge base and unknown to the other.

Example 4 (Continue Example 2) By Equation 6, we have $q_{WA}(C_1, D_1) = 0$, $q_{WA}(C_1, D_2) = 1.5$, and $q_{WA}(C_2, D_1) = 1.1$, $q_{WA}(C_2, D_2) = 1.3$. Therefore, the quantity of weak agreement between B_1 and B_2 is $Q_{WA}(B_1, B_2) = 1.5$.

The function Q_{WA} is not monotonic with regard to the subset relation, as shown below

Example 5 Let $B_1 = \{(p, 1)\}$ and $B_2 = \{(q, 1)\}$, then $Q_{WA}(B_1, B_2) = 2$. However, the quantity of weak agreement between $B_3 = \{(p, 1), (p \rightarrow \neg q, 1)\}$ and B_2 is $Q_{WA}(B_2, B_3) = 1$, where $B_1 \subseteq B_3$.

Based on the quantity of conflict and quantities of agreement, we can define the following relationships between two knowledge bases B_1 and B_2 as

- B_1 and B_2 are said to be *totally in conflict* iff $Q_c(B_1, B_2) > 0$ and $Q_{SA}(B_1, B_2) = Q_{WA}(B_1, B_2) = 0$.
- B_1 and B_2 are *totally in agreement* iff $Q_{Con}(B_1, B_2) = Q_{WA}(B_1, B_2) = 0$ and $Q_{SA}(B_1, B_2) > 0$.
- B_1 and B_2 are *partially in conflict* iff $Q_{Con}(B_1, B_2) > 0$ and $Q_{SA}(B_1, B_2) + Q_{WA}(B_1, B_2) > 0$.

3.3 Degree of conflict and degrees of agreement

In this subsection, we will define a degree of conflict and two degrees of agreement between two PKBs. The degree of conflict measures the extent to which two knowledge bases are in conflict. It was first introduced in [Hunter, 2004] to measure the believability of arguments.

Definition 11 Let B_1 and B_2 be two self-consistent knowledge bases, and $Dalal(B_1, B_2)$ be the Dalal distance between B_1 and B_2 . The degree of conflict between B_1 and B_2 , denoted as $C(B_1, B_2)$, is defined as follows:

$$C(B_1, B_2) = \frac{Dalal(B_1, B_2)}{\log_2(|I(B_1 \cup B_2)|)} \quad (8)$$

Although this definition gives a method to measure the degree of conflict, it can sometimes overestimate the degree of conflict between two knowledge bases, because it doesn't distinguish the influences of strong agreement and weak agreement. For example, let us consider two pairs of knowledge bases (B_1, B_2) and (B'_1, B'_2) , where $B_1 = \{p, q, r\}$, $B_2 = \{\neg p, q, r\}$ and $B'_1 = \{p\}$. Although the quantity of conflict between B_1 and B_2 is 1, the quantity of strongly agreement between them is 2. This means B_1 and B_2 have more in agreement than in conflict. In contrast, although the

quantity of conflict between B'_1 and B_2 is also 1, but with $D_{SA}(B'_1, B_2) = 0$ and $D_{WA}(B'_1, B_2) = 2$, the degree of conflict between B'_1 and B_2 should be higher than that between B_1 and B_2 . However, by Equation 8, $C(B_1, B_2) = C(B'_1, B_2) = 1/3$. This is not reasonable.

We propose the following revised definition of the degree of conflict.

Definition 12 Let B_1 and B_2 be two PKBs. Let C and D be WPIs of B_1 and B_2 respectively. $Atom_C(C, D)$ denotes the cardinality of the set of atoms which are in conflict in $C \cup D$. Then the degree of conflict between C and D is defined as

$$d_{Con}(C, D) = \frac{q_{Con}(C, D)}{Atom_C(C, D) + q_{SA}(C, D) + \lambda q_{WA}(C, D)}, \quad (9)$$

where $\lambda \in (0, 1]$ is used to weaken the influence of the quantity of weak agreement on the degree of conflict. In the following, we always assume that $\lambda = 0.5$, that is, the quantity of weak agreement only has "half" as much the influence on the degree of conflict as the quantity of strong agreement.

Definition 13 Let B_1 and B_2 be two PKBs. Suppose C and D are the sets of WPIs of B_1 and B_2 respectively, then the degree of conflict between B_1 and B_2 is defined as

$$D_{Con}(B_1, B_2) = \min\{d_{Con}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (10)$$

The advantage of our degree of conflict can be seen from the following example.

Example 6 Let us consider a dialogue between three people John, Mary, and Gary. They are discussing "whether Italy is the best football team in the world" (p) and "whether the best forwards are in Brazil" (q). John says "I think Italy is the best football team in the world and the best forwards are in Brazil", Mary says "No, I think France is the best team, but I agree with you that the best forwards are in Brazil", and Gary says "No, I think France is the best team". So the knowledge bases are $John = \{p, q\}$, $Mary = \{\neg p, q\}$ and $Gary = \{\neg p\}$. By Equation 8, we have $C(John, Mary) = C(John, Gary) = 1/2$. This is not reasonable, because John and Mary agree on q , the degree of conflict between them should be less than the degree of conflict between John and Gary. In contrast, we have $D_c(John, Mary) = 1/2$ and $D_{Con}(John, Gary) = 2/3$, so $D_c(John, Mary) < D_{Con}(John, Gary)$.

Proposition 5 Let B_1, B_2 be two classical knowledge bases. Suppose $C(B_1, B_2)$ and $D_{Con}(B_1, B_2)$ are the degrees of conflict defined by Definition 11 and Definition 13 respectively. Then $C(B_1, B_2) \geq D_{Con}(B_1, B_2)$.

Similarly, we can define the degree of strong agreement. We hold that the influence of the quantity of conflict on the degree of strong agreement is more than that of the quantity of weak agreement.

Definition 14 Let B_1 and B_2 be two PKBs. Let C and D be WPIs of B_1 and B_2 respectively. $Atom_{SA}(C, D)$ denotes the cardinality of the set of atoms which are included in both C and D . Then the degree of strong agreement between C and

D is defined as

$$d_{SA}(C, D) = \frac{q_{SA}(C, D)}{Atom_{SA}(C, D) + q_{Con}(C, D) + \lambda q_{WA}(C, D)}, \quad (11)$$

where $\lambda \in (0, 1]$ is used to weaken the influence of the quantity of weak agreement on the degree of conflict. As in Definition 12, we usually take $\lambda = 0.5$.

Definition 15 Let B_1 and B_2 be two PKBs. Suppose C and D are the sets of weighted prime implicants of B_1 and B_2 respectively, then the degree of strong agreement between B_1 and B_2 is defined as

$$D_{SA}(B_1, B_2) = \max\{d_{SA}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (12)$$

Example 7 Let $B_1 = \{(p, 0.8), (q \vee r, 0.4), (p \rightarrow s, 0.5)\}$ and $B_2 = \{(p \vee \neg r, 0.8), (q, 0.6), (\neg s, 0.7)\}$. The WPIs of B_1 are $C_1 = \{(p, 0.8), (q, 0.4), (s, 0.5)\}$ and $C_2 = \{(p, 0.8), (r, 0.4), (s, 0.5)\}$, and the WPIs of B_2 are $D_1 = \{(p, 0.8), (q, 0.6), (\neg s, 0.7)\}$ and $D_2 = \{(\neg r, 0.8), (q, 0.6), (\neg s, 0.7)\}$. So $d_{SA}(C_1, D_1) = 0.48$, $d_{SA}(C_1, D_2) = 0.17$, $d_{SA}(C_2, D_1) = 0.4$, $d_{SA}(C_2, D_2) = 0$. Therefore, $D_{SA}(B_1, B_2) = 0.48$.

The degrees of conflict and strong agreement are related to each other.

Proposition 6 Let B_1 and B_2 be two PKBs. Then their degree of conflict and degree of strong agreement cannot be greater than 0.5 at the same time, i.e., if $D_{Con}(B_1, B_2) > 0.5$, then $D_{SA}(B_1, B_2) \leq 0.5$.

We can also define the degree of weak agreement.

Definition 16 Let B_1 and B_2 be two PKBs. Let C and D be WPIs of B_1 and B_2 respectively. $Atom_{WA}(C, D)$ denotes the cardinality of the set of atoms which are included in only one of C and D but not both. Then the degree of weak agreement between C and D is defined as

$$d_{WA}(C, D) = \frac{q_{WA}(C, D)}{Atom_{WA}(C, D) + q_{Con}(C, D) + q_{SA}(C, D)}, \quad (13)$$

In Definition 16, the quantity of conflict and quantity of strong agreement have the same influence on the degree of weak agreement. When both B_1 and B_2 are classical knowledge bases, we have $d_{WA}(C, D) = \frac{Atom_{WA}(C, D)}{|Atom(C \cup D)|}$.

Definition 17 Let B_1 and B_2 be two PKBs. Suppose C and D are the sets of WPIs of B_1 and B_2 respectively, then the degree of weak agreement between B_1 and B_2 is defined as

$$D_{WA}(B_1, B_2) = \max\{d_{WA}(C, D) | C \in \mathcal{C}, D \in \mathcal{D}\}. \quad (14)$$

Example 8 (Continue Example 7) By Definition 16, we have $d_{WA}(C_1, D_1) = 0$, $d_{WA}(C_1, D_2) = 0.55$, $d_{WA}(C_2, D_1) = 0.3$, $d_{WA}(C_2, D_2) = 0.48$. So the degree of weak agreement between B_1 and B_2 is 0.55.

Proposition 7 Let B_1 and B_2 be two possibilistic knowledge bases. If $d_{WA}(B_1, B_2) > 0.5$, then $d_{Con}(B_1, B_2) < 0.5$ and $d_{SA}(B_1, B_2) < 0.5$.

Proposition 7 shows that if the degree of weak agreement between two knowledge bases is large, i.e., greater than 0.5, then both the degree of conflict and degree of strong agreement between them should be small, i.e., less than 0.5.

4 Impact of Measures of Conflict and Agreement

4.1 Choice of combination operators

Many operators have been proposed for merging PKBs. Given two PKBs B_1 and B_2 with possibility distributions π_{B_1} and π_{B_2} respectively, the semantic results of their combination by a T-norm tn and a T-conorm ct are

$$\forall \omega, \pi_{tn}(\omega) = tn(\pi_{B_1}(\omega), \pi_{B_2}(\omega)), \quad (15)$$

$$\forall \omega, \pi_{ct}(\omega) = ct(\pi_{B_1}(\omega), \pi_{B_2}(\omega)). \quad (16)$$

The syntactic results associated with π_{tn} and π_{ct} are respectively the following PKBs [Benferhat *et al.*, 2002]:

$$B_{tn} = B_1 \cup B_2 \cup \{(\phi_i \vee \psi_j, ct(\alpha_i, \beta_j)) | (\phi_i, \alpha_i) \in B_1 \text{ and } (\psi_j, \beta_j) \in B_2\}, \quad (17)$$

$$B_{ct} = \{(\phi_i \vee \psi_j, tn(\alpha_i, \beta_j)) | (\phi_i, \alpha_i) \in B_1 \text{ and } (\psi_j, \beta_j) \in B_2\}. \quad (18)$$

If we require that the result of the combination be a consistent knowledge base, then the T-norm based operator cannot be used when there is a conflict between B_1 and B_2 . In this case, we can only use a T-conorm based operator.

Typical T-norm operators are the minimum operator, the product operator and the Łukasiewicz T-norm (tn_L for short) ($\max(0, a + b - 1)$). The duality relation respectively yields the following T-conorm: the maximum operator, the probabilistic sum (ct_{ps} for short) ($a + b - ab$), and the bounded sum (ct_{bs} for short) ($\min(1, a + b)$).

Although some criteria to choose between merging operators have been given in [Benferhat *et al.*, 1997; 2002], these criteria are not enough.

Suppose two PKBs B_1 and B_2 are consistent, then the degree of conflict between them must be 0 and at least one of the degrees of agreement is greater than 0. If the degree of strong agreement between B_1 and B_2 is very high, then B_1 and B_2 share beliefs on most of the topics. In this case, it is advisable to combine them using an operator with higher reinforcement effect, for example, the Łukasiewicz t-norm $\max(0, a+b-1)$. However, if the degree of strong agreement between B_1 and B_2 is low and the degree of weak agreement between them is very high, it is advisable to combine them using the minimum operator which does not have any reinforcement effect.

Suppose B_1 and B_2 are in conflict, we usually use a T-conorm to combine them. When the degree of conflict between B_1 and B_2 is very high, then B_1 and B_2 have mostly different beliefs and we can choose the “bounded sum” operator which has a high counteract effect. On the other hand, if the degree of conflict between B_1 and B_2 is very low, we can choose the maximum which does not have any counteract effect.

More formally, we have the following criteria to choose between merging operators.

Merging operators selection criterion: Let $\oplus_1$ and $\oplus_2$ be two operators applied to merge A and B , and C and D respectively, then for all $a, b \in [0, 1]$,

- (1) $\oplus_1(a, b) \leq \oplus_2(a, b)$ if $0 < D_{Con}(A, B) < D_{Con}(C, D)$
- (2) $\oplus_1(a, b) \geq \oplus_2(a, b)$ if $D_{Con}(A, B) = 0$ and $D_{SA}(A, B) < D_{SA}(C, D)$.

Example 9 Let $B_1 = \{(p, 0.6), (q \vee \neg r, 0.7), (s, 0.6)\}$ and $B_2 = \{(p, 0.5), (q, 0.4), (s, 0.4)\}$, where $D_{Con}(B_1, B_2) = 0$ and $D_{SA}(B_1, B_2) = 0.43$. The merging operator here should be the product operator, and the result of merging is $B = \{(p, 0.6), (q \vee \neg r, 0.7), (s, 0.6), (p, 0.5), (q, 0.4), (s, 0.4), (p, 0.8), (p \vee q, 0.76), (p \vee s, 0.76), (p \vee q \vee \neg r, 0.85), (q \vee \neg r, 0.88), (q \vee \neg r \vee s, 0.88), (p \vee s, 0.8), (q \vee s, 0.76), (s, 0.76)\}$. However, if we use a Lukasiewicz t-norm, the result of merging is $B' = \{(p, 0.6), (q \vee \neg r, 0.7), (s, 0.6), (p, 0.5), (q, 0.4), (s, 0.4), (p, 1), (p \vee q, 1), (p \vee s, 1), (p \vee q \vee \neg r, 1), (q \vee \neg r, 1), (q \vee \neg r \vee s, 1), (p \vee s, 1), (q \vee s, 1), (s, 1)\}$. In B' , the weights of formulae p and s are reinforced to 1. However, the certainty degrees of p and s are not high in both B_1 and B_2 . Moreover, B_1 and B_2 are not in strong agreement with each other because $D_{SA}(B_1, B_2) = 0.43$. So it is not reasonable to increase the weights of p and s to the highest certainty degree 1. In contrast, in B , p and s have certainty degrees of 0.8 and 0.76 respectively. Therefore the result of the product operator reflects the reinforcement of B_1 and B_2 more accurately than that of the Lukasiewicz t-norm.

4.2 Ordering sources

In this section, we define an ordering relation to compare different knowledge bases based on the degree of conflict.

Definition 18 Let B_i, B_j, B be three PKBs. A closeness relation $\preceq_B$ with regard to B is defined as.

$$B_i \preceq_B B_j \text{ iff } D_{Con}(B_j, B) \leq D_{Con}(B_i, B)$$

B_j is closer to B than B_i to B ($B_i \preceq_B B_j$) iff B_j has less quantity of conflict and more quantities of agreement with B than B_i . If $B_i \preceq_B B_j$, then we may view B_j is less problematic or more reliable than B_i with regard to B .

Example 10 Let $B_1 = \{(\neg p, 0.8), (\neg q, 0.5), (\neg r \vee s, 0.7)\}$, $B_2 = \{(\neg p, 0.8), (\neg q, 0.5), (\neg r, 1), (s, 0.7)\}$, and $B = \{(p \vee q, 0.8), (\neg s \vee r, 1)\}$. Since $D_{Con}(B_1, B) = 0.22 < 0.41 = D_{Con}(B_2, B)$, so $B_2 \preceq_B B_1$.

5 Conclusion

The main contribution of this work is that we not only measure the conflict between two prioritized knowledge bases but also measure their agreement in two ways. We defined the quantity of conflict and two quantities of agreement. The quantity of conflict is a generalization of the Dalal distance. We then defined the degree of conflict and degrees of agreement based on both the quantity of conflict and the quantities of agreement. We have shown that the definition of degree of conflict is more reasonable than that defined in [Hunter, 2004]. The measures of conflict and agreement can be very useful in many applications, such as belief merging, argumentation and heterogeneous source integration and management. Another potential application of the measures of conflict is to belief revision as we have shown that the quantity of conflict generalizes the Dalal distance.

We didn't touch the computational issue in this paper. It is clear that the computations of measures of conflict and agreement defined in Section 3.2 and 3.3 depend on the computation of WPIs. It has been shown in [Bittencourt et al.,

2004] that the computation of the set of prime implicants of a formula represented by conjunctive normal form is NP-complete using a transformation algorithm in [Bittencourt et al., 2003]. Given a PKB $B = \{(\phi_1, a_1), \dots, (\phi_n, a_n)\}$ where ϕ_i are clauses, it is expected that the computation of all the WPIs of B is also NP-complete by generalizing the transformation algorithm. This problem will be discussed in a future paper.

References

- [Benferhat et al., 1997] S. Benferhat, D. Dubois, and H. Prade. From semantic to syntactic approaches to information combination in possibilistic logic. In Bouchon-Meunier, B. eds., *Aggregation and Fusion of Imperfect Information*, 141-151, 1997. Physica. Verlag.
- [Benferhat et al., 2002] S. Benferhat, D. Dubois, S. Kaci, H. Prade. Possibilistic merging and distance-Based fusion of propositional information. *Annals of Mathematics and Artificial Intelligence*, 34(1-3): 217-252, 2002.
- [Benferhat & Kaci, 2003] S. Benferhat and S. Kaci. Fusion of possibilistic knowledge bases from a postulate point of view. *International Journal of Approximate Reasoning*, 33(3): 255-285, 2003.
- [Bittencourt et al., 2004] G. Bittencourt, L. Perrussel and J. Marchi. A syntactical approach to revision. In *Proc. of ECAI'04*, 788-792, 2004.
- [Bittencourt et al., 2003] G. Bittencourt, J. Marchi and R.S. Padilha. A syntactical approach to satisfaction. In *Proc. of the 4th International Workshop on the Implementation of Logics*, 18-32, 2003.
- [Cadoli & Donini, 1997] M. Cadoli and F.M. Donini. A survey on knowledge compilation. *AI Communication*, 10(3-4): 137-150, 1997.
- [Dalal, 1988] M. Dalal. Investigations into a theory of knowledge base revision: Preliminary report. *Proc. of AAAI'88*, 3-7, 1988. MIT Press.
- [Dubois et al., 1994] D. Dubois, J. Lang, and H. Prade. Possibilistic Logic. In *Handbook of logic in Artificial Intelligence and Logic Programming*, Volume 3, 439-513, 1994.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in flux: modeling the dynamics of epistemic states*. MIT Press, 1988.
- [Hunter, 2002] A. Hunter. Measuring inconsistency in knowledge via quasi-classical models. *Proc. of AAAI'02*, 68-73, 2002.
- [Hunter, 2004] A. Hunter. Making argumentation more believable. *Proc. of AAAI'04*, 269-274, 2004.
- [Konieczny et al., 2003] S. Konieczny, J. Lang, and P. Marquis. Quantifying information and contradiction in propositional logic through test actions. *Proc. of IJCAI'03*, 106-111, 2003.
- [Konieczny & Ramon, 1998] S. Konieczny and R. Pino Prez. On the logic of merging. *Proc. of KR'98*, 488-498, 1998.
- [Lozinskii, 1994] E. Lozinskii. Resolving contradictions: a plausible semantics for inconsistent systems. *Journal of Automated Reasoning*, 12:1-31, 1994.

Minimal and Absent Information in Contexts

Floris Roelofsen¹ and Luciano Serafini²

¹ Institute for Logic, Language, and Computation, Amsterdam

² Istituto per la Ricerca Scientifica e Tecnologica, Trento

Abstract

Multi-context systems (MCS) represent contextual information flow. We show that the semantics of an MCS is completely determined by the information that is obtained when simulating the MCS, in such a way that a *minimal* amount of information is deduced at each step of the simulation.

In MCS, the acquisition of new information is based on the *presence* of other information only. We give a generalized account to model situations in which information can be obtained as a result of the *absence* of other information as well.

1 Introduction

Based on motivational papers by McCarthy [1987] and Giunchiglia [1993] several formalizations of contextual information and inter-contextual information flow have been proposed. Most notable are the propositional logic of context developed by McCarthy, Buvač and Mason [1993; 1998], and the multi-context systems devised by Giunchiglia and Serafini [1994], which later have been associated with the local model semantics introduced by Giunchiglia and Ghidini [2001]. Serafini and Bouquet [2004] have argued from a technical point of view that multi-context systems constitute the most general formal framework. This conclusion is supported by a more conceptual argument of Benerecetti et.al. [2000].

A multi-context system describes the information available in a number of contexts (i.e., to a number of people / agents / databases, etc.) and specifies the information flow between those contexts. The local model semantics defines a system to entail a piece of information, if and only if that piece of information is acquired, independently of how the information flow described by the system is accomplished.

Our first contribution is based on the observation that the local model semantics of a multi-context system is completely determined by the information that is obtained when simulating the information flow specified by the system, in such a way that a *minimal* amount of information is deduced at each step of the simulation. We define an operator which suitably implements such a simulation, and thus determines the information entailed by the system. This operator constitutes a first constructive account of the local model semantics.

Our second contribution is based on the observation that in multi-context systems, new information is derived based on the *presence* of other information only. However, in many natural situations (concrete examples will be given below), new information is obtained due to a *lack* of other information. We propose a generalized framework so as to account for such situations. Non-monotonic reasoning techniques are applied to formulate a suitable semantics for this framework.

We proceed, in section 2, with a brief review of multi-context system syntax and local model semantics. Minimality is discussed in section 3, and the generalized framework is presented in section 4. We conclude, in section 5, with a concise recapitulation of our main observations and results.¹

2 Preliminaries

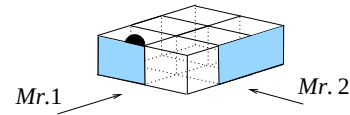


Figure 1: a magic box.

A simple illustration of the main intuitions underlying the multi-context system framework is provided by the situation depicted in figure 1. Two agents, Mr.1 and Mr.2, are looking at a box from different angles. The box is called magic, because neither Mr.1 nor Mr.2 can make out its depth. As some sections of the box are out of sight, both agents have partial information about the box. To express this information, Mr.1 only uses proposition letters l (there is a ball on the left) and r (there is a ball on the right), while Mr.2 also uses a third proposition letter c (there is a ball in the center).

In general, we consider a set of contexts I , and a language L_i for each context $i \in I$. We assume I and $\{L_i\}_{i \in I}$ to be fixed, unless specified otherwise. Moreover, for the purpose of this paper we assume each L_i to be built over a finite set of proposition letters, using standard propositional connectives.

To state that the information expressed by a formula $\varphi \in L_i$ is established in context i we use so-called *labeled formulas* of the form $i : \varphi$ (if no ambiguity arises, we simply refer

¹See <http://home.student.uva.nl/f.roelofsen/> for all proofs that are omitted or merely sketched here.

to labeled formulas as formulas, and we even use capital letters F , G , and H to denote labeled formulas, if the context label is irrelevant). A rule r is an expression of the form:

$$F \leftarrow G_1 \wedge \dots \wedge G_n \quad (1)$$

where F and all G 's are labeled formulas; F is called the consequence of r and is denoted by $\text{cons}(r)$; all G 's are called premises of r and together make up the set $\text{prem}(r)$. Rules without premises are called *facts*. Rules with at least one premiss are called *bridge rules*. A *multi-context system* (system hereafter) is a finite set of rules. A fact describes information that is established in a certain context, independently of which information is obtained in other contexts. A bridge rule specifies which information is obtained in one context, if other pieces of information are acquired in different contexts. Thus a system can be thought of as a specification of contextual information and an inter-contextual information flow.

Example 1 The situation in figure 1 can be modeled by the following system:

$$\begin{array}{ll} 1 : \neg r & \leftarrow \\ 2 : l & \leftarrow \\ 1 : l \vee r & \leftarrow \quad 2 : l \vee c \vee r \\ 2 : l \vee c \vee r & \leftarrow \quad 1 : l \vee r \end{array}$$

Mr.1 knows that there is no ball on the right, Mr.2 knows that there is a ball on the left, and if any agent gets to know that there is a ball in the box, then he will inform the other agent.

A classical interpretation m of language L_i is called a *local model* of context i . A set of local models is called a *local information state*. Intuitively, every local model in a local information state represents a “possible state of affairs”. If a local information state contains exactly one local model, then it represents complete information. If it contains more than one local model, then it represents partial information: more than one state of affairs is considered possible. A *distributed information state* is a set of local information states, one for each context. In conformity with the literature, we will refer to distributed information states as *chains*.

Example 2 The situation in figure 1, in which Mr.1 knows that there is no ball on the right but does not know whether there is a ball on the left, is represented by a chain whose first component $[\{l, \neg r\}, \{ \neg l, \neg r \}]$ contains two local models. As such, the chain reflects Mr.1's uncertainty about the left section of the box.

A chain c satisfies a labeled formula $i : \varphi$ (denoted $c \models i : \varphi$) iff all local models in its i^{th} component classically satisfy φ . A rule r is *applicable* with respect to a chain c iff c satisfies every premiss of r . Notice that facts are applicable with respect to any chain. A chain c *complies* with a rule r , iff, whenever r is applicable with respect to c , then c satisfies r 's consequence. We call c a *solution chain* of a system S iff it complies with every rule in S . A formula F is *true* in S (denoted $S \models F$) iff every solution chain of S satisfies F .

Let $\mathbf{C}$ denote the set of all chains. Notice that, as each L_i is assumed to be built over a finite set of proposition letters, $\mathbf{C}$ is assumed to be finite. Let $c^\perp$ denote the chain containing every local model of every context (notice that $c^\perp$ does not satisfy any non-tautological expression); let $c^\top$ denote the chain

containing no local models at all (notice that $c^\top$ satisfies any expression). If C is a set of chains, then the component-wise union of C is the chain, whose i^{th} component consists of all local models that are in the i^{th} component of some chain in C . If c and c' are chains, then $c \setminus c'$ denotes the chain, whose i^{th} component consists of all local models that are in c_i but not in c'_i . Finally, we sometimes say that a local model m is (not) in c , when we actually mean that m is (not) in some (any) component c_i of c .

3 Minimality

We order chains according to the amount of information they convey. Intuitively, the more local models a chain component contains, the more possibilities it permits, so the less informative it is. Formally, we say that c is *less informative* than c' ($c \preceq c'$), if for every i we have $c_i \supseteq c'_i$. If, moreover, for at least one i we have $c_i \supset c'_i$, then we say that c is *strictly less informative* than c' ($c \prec c'$).

Lemma 1 Let c and c' be two chains, such that $c \preceq c'$. Then any formula that is satisfied by c is also satisfied by c' .

We call c *minimal* among a set of chains C , iff c is in C and no other chain c' in C is strictly less informative than c . In particular, we call c a *minimal solution chain* of a system S , if it is minimal among the set of all solution chains of S .

Theorem 1 Every system S has a unique minimal solution chain c_S .

Proof. Let C_S be the set of all solution chains of S . $C_S \neq \emptyset$ as $c^\top \in C_S$ for any S . Let c_S be the component-wise union of all chains in C_S . Then $c_S \in C_S$ and $c_S \preceq c$ for any $c \in C_S$. So c_S is the unique minimal solution chain of S . $\square$

Theorem 2 The semantics of a system S is completely determined by its unique minimal solution chain c_S :

$$S \models F \quad \Leftrightarrow \quad c_S \models F$$

Proof. F is true in S iff F is satisfied by all solution chains of S iff F is satisfied by the component-wise union c_S of all solution chains of S . $\square$

Theorem (1) and (2) are extremely useful, because they establish that, to answer queries about a system S , it is no longer necessary to compute all solution chains of S ; we only need to consider the system's minimal solution chain c_S .

3.1 Computing the Minimal Solution Chain

The minimal solution chain of S can be characterized as the $\preceq$ -least fixpoint of an operator $\mathbf{T}_S$, which, intuitively, simulates the information flow specified by S . Let $S^*(c)$ denote the set of rules in S , which are applicable w.r.t. c . Then we define:

$$\mathbf{T}_S(c) = c \setminus \{m \mid \exists r \in S^*(c) : m \notin \text{cons}(r)\} \quad (2)$$

For every rule r in S that is applicable w.r.t. c , $\mathbf{T}_S$ removes from c all local models that do not satisfy $\text{cons}(r)$.

Intuitively, this corresponds to augmenting c with the information expressed by $\text{cons}(r)$. In this sense, $\mathbf{T}_S$ simulates the information flow described by S . As $\mathbf{T}_S(c)$ is obtained from c only by removing local models from it, $\mathbf{T}_S(c)$ is always more informative than c .

As $(\mathbf{C}, \preceq)$ forms a complete lattice, and $\mathbf{T}_S$ is monotone and continuous w.r.t. $\preceq$, [Tarski, 1955] yields:

Theorem 3 $\mathbf{T}_S$ has a $\preceq$ -least fixpoint, which is obtained after a finite number of consecutive applications of $\mathbf{T}_S$ to $c^\perp$.

Lemma 2 Let c be a chain and let S be a system. Then c is a fixpoint of $\mathbf{T}_S$ if and only if c is a solution chain of S .

Theorem 4 Let S be a system. Then the minimal solution chain c_S of S coincides with the $\preceq$ -least fixpoint of $\mathbf{T}_S$.

From theorems 3 and 4 we conclude that the minimal solution chain c_S of a system S is obtained after a finite number of applications of $\mathbf{T}_S$ to the least informative chain $c^\perp$. But we can even prove a slightly stronger result:

Theorem 5 Let S be a system and let $|S|$ denote the number of bridge rules in S . Then the minimal solution chain c_S of S is obtained after at most $|S| + 1$ applications of $\mathbf{T}_S$ to $c^\perp$.

In fact, a slightly more involved, but essentially equivalent procedure was introduced for rather different reasons by Roelofsen et.al. [2004]. This procedure was shown to have worst-case time complexity $O(|S|^2 \times 2^M)$, where M is the maximum number of propositional variables in either one of the contexts involved in S .

Example 3 Consider the system from example 1. Applying $\mathbf{T}_S$ to $c^\perp$ establishes the facts given by the first two rules of the system. But then Mr.2 knows that there is a ball in the box, so the next application of $\mathbf{T}_S$ simulates the information flow specified by the third rule of the system: Mr.2 informs Mr.1 of the presence of the ball. The resulting chain is left unaltered by any further application of $\mathbf{T}_S$, and therefore constitutes the minimal solution chain of S . The fact that this chain satisfies the formula $1 : l$ reflects, as desired, that Mr.1 has come to know that there is a ball in the left section of the box.

4 Absent Information

Rules of the form (1) only allow us to model a rather restricted kind of information flow, namely one in which new information is established based on the presence of other information only. There are many natural situations in which information is obtained as a result of the absence of other information. Such situations cannot be modeled by the present formalism.

Example 4 (Coordination) Let d_1, d_2 be two meteorological databases that collect their respective data from sensors located in different parts of the country. At the end of the day each database produces a weather forecast based on its own data but also on the information obtained by the other database. For example, d_1 predicts rain, if that follows from its own data and if, moreover, d_2 does not maintain that it won't rain:

$$1 : r \leftarrow 1 : r \wedge \text{not } 2 : \neg r$$

Example 5 (Integration) Let d_1 and d_2 be as in example 4 and let d_3 be a third database, which integrates the information obtained in d_1 and d_2 , respectively. Any piece of information that is established by d_1 and not refuted by d_2 (or vice versa) is included in d_3 :

$$\begin{aligned} 3 : \varphi &\leftarrow 1 : \varphi \wedge \text{not } 2 : \neg \varphi \\ 3 : \varphi &\leftarrow 2 : \varphi \wedge \text{not } 1 : \neg \varphi \end{aligned}$$

Example 6 (Trust) Let d_1, d_2 , and d_3 be as in example 5. It would be natural for d_3 to regard d_1 as more trustworthy than d_2 (or vice versa). In this case any piece of information that is established in d_1 is automatically included in d_3 , but information obtained in d_2 is only included in d_3 if it is not refuted by d_1 :

$$\begin{aligned} 3 : \varphi &\leftarrow 1 : \varphi \\ 3 : \varphi &\leftarrow 2 : \varphi \wedge \text{not } 1 : \neg \varphi \end{aligned}$$

To model these situations we need rules r of the form:

$$F \leftarrow G_1 \wedge \dots \wedge G_m \wedge \text{not } H_1 \wedge \dots \wedge \text{not } H_n \quad (3)$$

where F , all G 's, and all H 's are labeled formulas. As before, F is called the consequence of r ($\text{cons}(r)$). $G_1, \dots, G_m$ are called *positive premises* of r and together constitute the set $\text{prem}^+(r)$. $H_1, \dots, H_n$ are called *negative premises* of r and make up the set $\text{prem}^-(r)$. A rule does not necessarily have any premises ($m, n \geq 0$). In analogy with commonplace terminology in deductive database and logic programming theory, we call such rules *normal* rules, and finite sets of them *normal multi-context systems* (normal systems for short). If a rule only has positive premises, we call it a *positive* rule. Note that a system, which consists of positive rules only conforms with the original definition of multi-context systems. From now on we call such systems *positive* systems.

Our aim is to generalize the result obtained section 3, i.e. to define the semantics of a normal system S in terms of a single canonical chain c_S of S , such that, whenever S is a positive system, c_S coincides with the minimal solution chain of S .

A first naive attempt would be to say that a chain c complies with a normal rule r iff it satisfies r 's consequence, whenever it satisfies every positive premise of r and does not satisfy any negative premise of r . The (minimal) solution chains of a normal system can then be defined as for positive systems. However, as the following example shows, a normal system does not generally have a unique minimal solution chain, and worse, its minimal solution chains do not generally correspond with its intended meaning.

Example 7 Let a system S be given by the following rule:

$$1 : p \leftarrow \text{not } 2 : q$$

Then S has two minimal solution chains:

$$\begin{aligned} c^p &= \left\{ \left[\begin{array}{c} \{p\} \\ \end{array} \right]_1 \left[\begin{array}{c} \{q\} \\ \{-q\} \end{array} \right]_2 \right\} \\ c^q &= \left\{ \left[\begin{array}{c} \{p\} \\ \{-p\} \end{array} \right]_1 \left[\begin{array}{c} \{q\} \\ \end{array} \right]_2 \right\} \end{aligned}$$

Intuitively, S provides no ground for deriving q in context 2. Thus, p should be derived in context 1, and every “proper” canonical chain of S should satisfy $1 : p$. As c^q fails to do so, it should be rejected as such. But how, then, should the canonical chain of a normal system be characterized?

Extensive research efforts have been involved with an analogous question in the setting of logic programming, when, in the late 80’s / early 90’s, a proper semantics for normal logic programs was sought. In motivating our characterization of canonical chains for normal multi-context systems, we will recall some important intuitions and adapt some crucial definitions that have resulted from these efforts.

A first desired property of canonical chains, introduced in the setting of logic programming by Apt et.al. [1988] and Bidoit and Froidevaux [1991], is termed *supportedness*. Intuitively, a chain c is a supported solution chain of a normal system S iff, whenever c satisfies a formula F , then S provides an explanation for why this is so.

Definition 1 We call a chain c a supported solution chain of a normal system S iff, whenever c satisfies a formula F , then S contains a set R of rules, such that:

- $\forall r \in R : \begin{cases} \forall G \in \text{prem}^+(r) : c \models G \\ \forall H \in \text{prem}^-(r) : c \not\models H \end{cases}$
- $\bigcup_{r \in R} \text{cons}(r) \models F$

Example 8 In example 7, as desired, c^p is a supported solution chain of S , while c^q is not. But both c^p and c^q are supported solution chains of the following extension S' of S :

$$\begin{array}{lcl} 1 : p & \leftarrow & \text{not } 2 : q \\ 2 : q & \leftarrow & 2 : q \end{array}$$

Intuitively, c^p should be accepted as a canonical chain of S' , but c^q should be rejected as such, because the explanation provided by S' for the fact that c^q satisfies $2 : q$ is *circular*, i.e., it relies on the very fact that c^q satisfies $2 : q$. So, in general, the concept of supportedness does not satisfactorily characterize the canonical chain of a normal system.

The notion of *well-supportedness*, first introduced for logic programs by Fages [1991], refines the notion of supportedness to avoid the counter-intuitive result obtained in example 8. Intuitively, a chain c is a well-supported solution chain of a normal system S iff, whenever c satisfies a formula F , then S provides a *non-circular* explanation for why this is so.

Fages also proved this notion to be equivalent to the notion of *stability*, which had been defined somewhat earlier by Gelfond and Lifschitz [1988]. The results obtained in section 3 pave the way for a straightforward adaptation of the notion of stability to our present setting.

Definition 2 Let c be a chain and S a normal system. Define:

$$\begin{aligned} S'(c) &= \{r \in S \mid \forall H \in \text{prem}^-(r) : c \not\models H\} \\ S''(c) &= \text{pos}(S'(c)) \end{aligned}$$

where $\text{pos}(S'(c))$ is obtained from $S'(c)$ by removing all negative premises from its rules. Then, c is a stable solution chain of S , iff it is the unique minimal solution chain of $S''(c)$.

Intuitively, a solution chain c of a system S is stable iff, whenever the information represented by c is assumed, then

the information flow specified by S reproduces exactly c . Namely, if c is assumed to contain valid information, then any rule in S , one of whose negative premises is satisfied by c , is certainly not applicable w.r.t. c . Negative premises which are *not* satisfied by c can be removed from the remaining rules, because they do certainly not inhibit those rules from being applicable w.r.t. c . Thus, c is stable iff it corresponds exactly to the meaning of $S''(c)$, i.e., by Theorem 2, to its minimal solution chain.

Example 9 In example 8, as desired, c^p is a stable solution chain of S' , while c^q is not.

For many systems, stability suitably characterizes a unique canonical chain. There are still some special cases, however, in which it fails to do so. We give some typical examples.

Example 10 Both c^p and c^q from example 7 are stable solution chains of the system given by the following rules:

$$\begin{array}{lcl} 1 : p & \leftarrow & \text{not } 2 : q \\ 2 : q & \leftarrow & \text{not } 1 : p \end{array}$$

Example 11 The following system does not have any stable solution chains.

$$1 : p \leftarrow \text{not } 1 : p$$

In both cases we think it is most reasonable to conclude that no information is derived at all, i.e. to regard $c^\perp$ as the proper canonical chain.

Example 12 The following system does not have any stable solution chains either.

$$\begin{array}{lcl} 1 : p & \leftarrow & \text{not } 1 : p \\ 1 : t & \leftarrow & \text{not } 2 : q \\ 2 : r & \leftarrow & 1 : t \end{array}$$

Example 12 illustrates that, even if the rest of the system is unproblematic, one single rule (in this case the first one) can cause the system not to have any stable solution chain at all. In this case, t and r should be derived in context 1 and 2, resp.

The *well-founded semantics*, first proposed for logic programs by van Gelder et.al. [1991] avoids the problems encountered in the above examples. The well-founded model of a program is defined as the least fixpoint of an operator, which, given an interpretation, determines the atoms that are necessarily true and those that are necessarily not true with respect to the program and the interpretation. It assigns *true* to the former set of atoms, and *false* to the latter. As a result, more atoms may become necessarily true or necessarily not true. Corresponding truth values are assigned until a fixpoint is reached. All atoms that have not been assigned a definite truth value, are interpreted as *unknown*.

Our approach shares an important intuition with the well-founded semantics for logic programs, namely, that while constructing the canonical chain of a system, it is not only important to accumulate the information that *can certainly be derived* from the system, but also to keep track of information that *can certainly not be derived* from the system.

But the two approaches are also fundamentally different. The well-founded semantics constructs a 3-valued interpretation I , which is minimal with respect to a *truth order* $\sqsubseteq$ (i.e. $I \sqsubseteq I'$ iff I makes less atoms true and more atoms false than I'), whereas we seek a chain which is minimal with respect to an *information order* $\preceq$ (i.e. $c \preceq c'$ iff c makes less expressions either true or false than c'). This particularly results in a different treatment of expressions that are found *not* to be true. To regard these expressions as false, as the well-founded semantics does, would be to introduce redundant information. Instead, in our setting, such expressions should simply be recorded as not being derivable.

4.1 Constructing the Canonical Chain

The canonical chain of a normal system S , henceforward denoted by c_S , is constructed by an iterative transformation of a datastructure $\langle c, a \rangle$, where:

- c is the “canonical chain under construction”. Initially, $c = c^\perp$. Every transformation of c removes from it those local models that are found not to be in c_S . So at any phase of the construction of c_S , c contains those local models that are *possibly* in c_S , and as such represents the information that is *necessarily* conveyed by c_S .
- a is the “anti-chain”. Initially, $c = c^\top$. Every transformation of a adds to it those local models that are found to be in c_S . So at any phase of the construction of c_S , a contains those local models that are *necessarily* in c_S , and as such represents the information that is *possibly* conveyed by c_S .

Observation 1 By construction, we have $c \preceq c_S \preceq a$. Therefore, by lemma 1, for any formula F :

$$\begin{aligned} c \models F &\Rightarrow c_S \models F \\ a \not\models F &\Rightarrow c_S \not\models F \end{aligned}$$

□

We call a chain-anti-chain pair $\langle c, a \rangle$ *less evolved* than another such pair $\langle c', a' \rangle$ (denoted as $\langle c, a \rangle \leq \langle c', a' \rangle$) iff c is less informative than c' and a is more informative than a' . If, moreover, c is strictly less informative than c' or a is strictly more informative than a' , then we say that $\langle c, a \rangle$ is strictly less evolved than $\langle c', a' \rangle$. We call $\langle c, a \rangle$ *minimal* among a set $\mathcal{CA}$ of chain-anti-chain pairs, iff $\langle c, a \rangle \in \mathcal{CA}$ and no other chain-anti-chain pair $\langle c', a' \rangle$ in $\mathcal{CA}$ is strictly less evolved than $\langle c, a \rangle$. Notice that, if $\langle c, a \rangle$ is minimal among $\mathcal{CA}$, then c is minimal among $\{c \mid \langle c, a \rangle \in \mathcal{CA}\}$.

Given a certain chain-anti-chain pair $\langle c, a \rangle$, the intended transformation Ψ_S first determines which rules in S will (not) be applicable w.r.t. c_S , and then refines $\langle c, a \rangle$ accordingly. The canonical chain c_S of S will be characterized as the first component of the $\leq$ -least fixpoint of Ψ_S .

We first specify how Ψ_S determines which rules will (not) be applicable w.r.t. c_S . Let $\langle c, a \rangle$ and a rule r in S be given. If r has a positive premise G , which is satisfied by c , then G will also be satisfied by c_S . On the other hand, if r has a negative premiss H , which is *not* satisfied by a , then H will not be satisfied by c_S either. So if all positive premises

of r are satisfied by c and all negative premises of r are not satisfied by a , then r will be applicable with respect to c_S :

$$S^+(c, a) = \left\{ r \in S \mid \begin{array}{l} \forall G \in \text{prem}^+(r) : c \models G \\ \text{and} \\ \forall H \in \text{prem}^-(r) : a \not\models H \end{array} \right\}$$

If r has a positive premise G , which is not satisfied by a , then G will not be satisfied by c_S either. If r has a negative premise H , which is satisfied by c , then H will be satisfied by c_S as well. In both cases r will certainly not be applicable with respect to c_S :

$$S^-(c, a) = \left\{ r \in S \mid \begin{array}{l} \exists G \in \text{prem}^+(r) : a \not\models G \\ \text{or} \\ \exists H \in \text{prem}^-(r) : c \models H \end{array} \right\}$$

For convenience, we write:

$$S^\sim(c, a) = S \setminus S^-(c, a)$$

Think of $S^\sim(c, a)$ as the set of rules that are *possibly* applicable with respect to c_S , and notice that $S^+(c, a) \subseteq S^\sim(c, a)$, whenever $c \preceq a$, and that $S^+(c, a) = S^\sim(c, a)$, if $c = a$.

Lemma 3 If S is a normal system and $\langle c, a \rangle$ and $\langle c', a' \rangle$ are two chain-anti-chain pairs s.t. $\langle c, a \rangle \leq \langle c', a' \rangle$, then we have:

1. $S^+(c, a) \subseteq S^+(c', a')$
2. $S^-(c, a) \subseteq S^-(c', a')$
3. $S^\sim(c, a) \supseteq S^\sim(c', a')$

Proof. Suppose that $\langle c, a \rangle \leq \langle c', a' \rangle$. Then, by definition, $c \preceq c'$ and $a' \preceq a$. Let r be a rule in S . For the first statement, suppose that $r \in S^+(c, a)$. Then c satisfies all of r 's positive premises, and a does not satisfy any of r 's negative premises. By lemma 1, the same goes for c' and a' , respectively, which implies that $r \in S^+(c', a')$. The second statement is proven analogously; the third follows directly from the second. □

Next, we specify how Ψ_S refines $\langle c, a \rangle$, based on $S^+(c, a)$ and $S^\sim(c, a)$. Every local model $m \in c_i$ that does not satisfy the consequence of a rule in $S^+(c, a)$ should certainly not be in c_S and is therefore removed from c . On the other hand, every local model $m \in c_i$ that satisfies the consequences of every rule in $S^\sim(c, a)$ should certainly be in c_S (S provides no ground for removing it) and is therefore added to a .

$$\Psi_S(\langle c, a \rangle) = \langle \Psi_S^c(\langle c, a \rangle), \Psi_S^a(\langle c, a \rangle) \rangle$$

where:

$$\begin{aligned} \Psi_S^c(\langle c, a \rangle) &= c \setminus \{m \mid \exists r \in S^+(c, a) : m \not\models \text{cons}(r)\} \\ \Psi_S^a(\langle c, a \rangle) &= a \cup \{m \mid \forall r \in S^\sim(c, a) : m \models \text{cons}(r)\} \end{aligned}$$

As $(\mathbf{C} \times \mathbf{C}, \leq)$ forms a complete lattice, and Ψ_S is monotone and continuous w.r.t. $\leq$, [Tarski, 1955] yields:

Theorem 6 Ψ_S has a $\leq$ -least fixpoint, which is obtained after finitely many iterations of Ψ_S , starting with $\langle c^\perp, c^\top \rangle$.

Definition 3 Let S be a normal system, and let $\langle c_S, a_S \rangle$ be the $\leq$ -least fixpoint of Ψ_S . We define c_S to be the canonical chain of S , and we define the semantics of S to be completely determined by c_S . That is, for every formula F :

$$S \models F \quad \equiv \quad c_S \models F$$

□

A bound on the number of iterations needed by Ψ_S to reach its $\leq$ -least fixpoint can be formulated in terms of the number of bridge rules in S .

Theorem 7 Let $|S|$ denote the number of bridge rules of a normal system S . Then, starting with $\langle c^\perp, c^\top \rangle$, Ψ_S will reach its $\leq$ -least fixpoint after at most $|S| + 1$ iterations.

The semantics for normal systems defined above properly generalizes the local model semantics for positive systems.

Theorem 8 The canonical chain of a positive system S coincides with its minimal solution chain.

Proof. If S is positive, then for every pair $\langle c, a \rangle$, $S^+(\langle c, a \rangle)$ and $S^*(c)$ coincide, so $\Psi_S^c(\langle c, a \rangle)$ does not depend on a . As a consequence, c_S is the $\leq$ -least fixpoint of $\mathbf{T}_S$ iff, for some anti-chain a_S , $\langle c_S, a_S \rangle$ is the $\leq$ -least fixpoint of Ψ_S . □

The canonical chain of a system S , and other fixpoints of Ψ_S , are intimately related to the stable solution chains of S .

Theorem 9 Let S be a normal system, let $\langle c_S, a_S \rangle$ be the $\leq$ -least fixpoint of Ψ_S , and let c_{stable} be any stable solution chain of S . Then $c_S \preceq c_{stable} \preceq a_S$.

Theorem 10 Let S be a normal system and let $\langle c_S, a_S \rangle$ be the $\leq$ -least fixpoint of Ψ_S . If c_S and a_S coincide, then c_S is the unique stable solution chain of S .

Finally, we remark that, in our view, all the examples presented above are suitably dealt with by the present analysis. We treat one of them explicitly.

Example 13 Let S be the system from example 12. Then:

$$c_S = \left\{ \left[\begin{array}{c} \{p, t\} \\ \{-p, t\} \end{array} \right]_1 \quad \left[\begin{array}{c} \{q, r\} \\ \{-q, r\} \end{array} \right]_2 \right\}$$

As desired, no information is derived about p and q , while t and r are indeed established in context 1 and 2, respectively.

5 Conclusions

We showed that the semantics of a multi-context system is completely determined by its least informative solution chain. We provided a way to compute this chain, and thus gave a first constructive account of the local model semantics.

We presented a generalized framework in which new information can be derived based on the absence of other information. We applied non-monotonic reasoning techniques to establish a suitable semantics for this framework.

References

- [Apt et al., 1988] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. 1988.
- [Benerecetti et al., 2000] M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(3):279–305, 2000.
- [Bidoit and Froidevaux, 1991] N. Bidoit and C. Froidevaux. General logical databases and programs: Default logic semantics and stratification. *Information and Computation*, 91:15–54, 1991.
- [Fages, 1991] F. Fages. A new fixpoint semantics for general logic programs compared with the wellfounded and the stable model semantics. *New Generation Computing*, 9(4), 1991.
- [Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *International Conference on Logic Programming (ICLP 88)*, pages 1070–1080, 1988.
- [Ghidini and Giunchiglia, 2001] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.
- [Giunchiglia and Serafini, 1994] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.
- [Giunchiglia, 1993] F. Giunchiglia. Contextual reasoning. *Epistemologia*, XVI:345–364, 1993.
- [McCarthy and Buvač, 1998] J. McCarthy and S. Buvač. Formalizing context (expanded notes). In *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. 1998.
- [McCarthy, 1987] J. McCarthy. Generality in artificial intelligence. *Communications of ACM*, 30(12):1030–1035, 1987.
- [McCarthy, 1993] J. McCarthy. Notes on formalizing context. In *International Joint Conference on Artificial Intelligence (IJCAI 93)*, pages 555–560, 1993.
- [Roelofsen et al., 2004] F. Roelofsen, L. Serafini, and A. Cimatti. Many hands make light work: Localized satisfiability for multi-context systems. In *European Conference on Artificial Intelligence (ECAI 04)*, pages 58–62, 2004.
- [Serafini and Bouquet, 2004] L. Serafini and P. Bouquet. Comparing formal theories of context in AI. *Artificial Intelligence*, 155:41–67, 2004.
- [Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
- [van Gelder et al., 1991] A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

Supervaluation Semantics for an Inland Water Feature Ontology

Paulo Santos* and **Brandon Bennett**
{psantos, brandon}@comp.leeds.ac.uk
School of Computing
University of Leeds, Leeds, UK

Georgios Sakellariou
georgios.sakellariou@ic.ac.uk
Electrical and Electronics Engineering
Imperial College, London, UK

Abstract

This paper describes an ontology for inland water features built using formal concept analysis and supervaluation semantics. The first is used to generate a complete lattice of the water domain, whereas supervaluation semantics is used to model the variability of the concepts in terms of threshold parameters. We also present an algorithm for a mechanism of individuation and classification of water features, from snapshots of river networks, according to the proposed ontology.

1 Introduction

There are at least three main aspects of an ontology for Geographical Information Systems (GIS): the ontology at the *data level*, whereby features of how raw data (either from sensor images or geographic surveys) are expressed in terms of numeric or symbolic information; the ontology at the *expert level*, that incorporates scientific knowledge about geographical objects; and the ontology at the *user level*, which should in some way specify the meanings of the vague and ambiguous geographic terminology which is used to describe the world in natural language. The present work is an investigation of an ontology of the latter kind for inland hydrographic features.

The ontology of inland water features proposed in this work is based on the *Waters* 'lexical field' presented as an example in [Ganter and Wille, 1999]. However, we shall analyse the domain in greater detail. In particular we shall explicitly model the intrinsic vagueness and ambiguity in water feature terms (such as *pond* and *lake*) by means of a variant of *supervaluation semantics* as proposed in [Fine, 1975; Bennett, 2001a; 2001b]. The methods developed in this work, however, are not restricted to the water feature domain, but can be applied to any ontology for GIS or, indeed, any complex domain containing vague concepts.

We argue in this work that *formal concept analysis* (FCA) and *supervaluation semantics* complement each other for building ontologies of naïve geography. On the one hand, FCA provides constructive tools for extracting complete lattices for any particular domain. Therefore, formal aspects of

the ontology can be assessed using the tools of lattice theory; moreover, there are also automatic methods for building lattices from a set of terms. Therefore, the problem of going from elements of a domain to the formal structure of this domain is solved by formal concept analysis. On the other hand, FCA does not handle changes in the interpretation of terms in the ontology due to intrinsic vagueness in the meaning of the corresponding concept. Supervaluation semantics provides rigorous methods for handling such changes of 'standpoint' while providing the machinery to make *reliable inferences* involving vague concepts. This ability to derive the reliable consequences of vague knowledge is the main advantage of supervaluation semantics over fuzzy logic [Elkan, 1993].

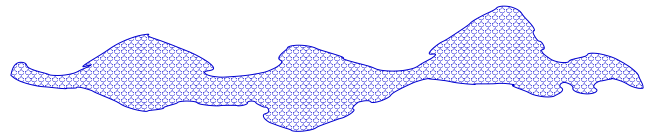


Figure 1: Vagueness in water features: *three lakes or a meandering river?*

Figure 1 shows an example of the kind of vagueness we are going to be dealing with in this work. This figure may depict either three lakes connected by channels or a meandering river stretch. The decision of one interpretation from another depends on the *standpoint* of the observer, i.e., on how the set of parameters distinguishing rivers from lakes are determined by the observer's judgement. Therefore, not only is vagueness pervasive to the feature definitions but also on how the features are individuated. It is worth pointing out also that this work concentrates on *sortes* vagueness, which is an indeterminacy in the boundaries of applicability of a term. This can be distinguished from conceptual *ambiguity* where a term has several qualitatively different interpretations.

The ontology of water features and the supervaluation semantics are encoded into a prototype implementation whose purpose is precisely to individuate qualitatively distinct features from a snapshot of a river network. Each individual feature is, thus, classified according to the definitions in the ontology, taking into account parameters handled by the supervaluation semantics.

This paper is organised as follows: Section 2 overviews some basic definitions of formal concept analysis and intro-

*Currently at Centro Universitário da FEI, São Paulo, Brazil.

duces the conceptual domain of inland water features (Section 2.2). Supervaluation semantics is discussed in Section 3. Section 4 describes our prototype implementation and Section 6 concludes this work.

2 Formal Concept Analysis

Formal concept analysis (FCA) [Ganter and Wille, 1999] is a powerful tool for formalising conceptual domains by means of algebraic structures such that the set of concepts can be represented as complete lattices.

2.1 Overview

The basic ideas underlying formal concept analysis are those of “formal context” and “formal concept” that are introduced below¹. Informally, a *context* refers to the formal structure that characterises a conceptual domain and a *concept* within a context is composed of the ontology terms and their attributes. *Terms* in an ontology are called *objects* in formal concept analysis terminology, in this work we use these words interchangeably.

Formal contexts and concepts

A context K is a triple $K \equiv \langle G, M, I \rangle$ in which G is the set of terms in the context K , M is the set of attributes of the context and I is an *incidence relation* between G and M . Let’s assume $g \in G$ and $m \in M$, then gIm is read as “the term g has attribute m ”.

In order to obtain the set of all attributes m ($m \in M$) common to a particular subset of objects, A , a *derivation operator* is defined as follows:

$$A' = \{m \in M \mid gIm \text{ for all } g \in A\}.$$

This operator is overloaded to refer to the set of all objects $g \in G$ that have all of their attributes in a particular subset of M . Formally:

$$B' = \{g \in G \mid gIm \text{ for all } m \in B\}.$$

Given a context $\langle G, M, I \rangle$, a *concept* is identified with a pair (A, B) , where $A \subseteq G$, $B \subseteq M$, such that $A' = B$ and $B' = A$. The sets A and B are called, respectively, the *extent* and the *intent* of the concept (A, B) .

Given two concepts $c_1 = (A_1, B_1)$ and $c_2 = (A_2, B_2)$, we say that c_1 is a subconcept of c_2 (equivalently, c_2 is a superconcept of c_1), or simply $(A_1, B_1) \leq (A_2, B_2)$, if $A_1 \subseteq A_2$. The set of all concepts of a particular context, ordered in this way, forms a complete lattice in which infimum and supremum are, respectively²:

$$\bigwedge_{t \in T} (A_t, B_t) = (\bigcap_{t \in T} A_t, (\bigcup_{t \in T} B_t)''),$$

and

$$\bigvee_{t \in T} (A_t, B_t) = ((\bigcup_{t \in T} A_t)'', \bigcap_{t \in T} B_t).$$

¹For simplicity we are going to drop the word formal in “formal context” and “formal concept” throughout this paper.

²As proved by the *basic theorem of concept lattice* [Ganter and Wille, 1999].

Many-valued contexts and scaling

Until now we have been describing the basic machinery of formal concept analysis without taking into account values of attributes with respect to objects. In fact, the definition of context described above only deals with cases of one-valued attributes where objects either have a particular attribute or not. However, in order to deal with the many-valued attributes found in the water feature domain (such as “flow” and “depth”), many-valued contexts should be taken into account.

A many-valued context is a quadruple $\langle G, M, W, I \rangle$ where G is a set of objects, M is a set of many-valued attributes, W are the attributes values and I is a ternary relation of $G \times M \times W$.

In order to represent many-valued contexts as complete lattices, they have to be reduced to one-valued contexts. This is done by means of partitioning each many-value attribute into a finite number of one-value attributes that are distinguished from one another by a set of thresholds. This partitioning process is named *scaling* and the set of partitions of an attribute is called the *conceptual scale* of that attribute in FCA terminology. A scaling can be understood as an interpretation of each of the attributes of a many-valued context by means of a *context of attributes*. A scale for a many-valued attribute m is a one-valued context $Sm = \langle Gm, Mm, Im \rangle$ with $m(G) \subseteq Gm$. In practise, every many-valued attribute m of a context is replaced by the one-valued scale attributes relative to $m(g)$, for any object $g \in G$.

2.2 The context of inland water features

In this section we propose a context for inland water recalling the *lexical field waters* introduced in [Ganter and Wille, 1999]. The present context is constituted of a set of objects (or water features): $G = \{\text{river, stream, canal, reservoir, lake, marsh, pond}\}$ whose set of attributes is $M = \{\text{linearity, size, origin, flow, depth}\}$.

The scale assumed for the linearity attribute partitions its values into *linear* and *non linear* according to a threshold on a *linearity coefficient*. The linearity coefficient assumed in this work is defined as the ratio between the diameters of the minimal bounding circle and the maximal internal circle w.r.t. the feature’s border.

In an ideal world, where lakes have the shape of discs and rivers the shape of lines, rivers and lakes could be identified as features whose linearity ratio are, respectively, different from and equal to one. However, this idealistic distinction does not hold for lakes that are elongated or for wide river stretches, for instance. In order to cope with this issue, we assume a threshold on the linearity ratio (called the *linearity threshold*) whereby if the ratio of a water feature is greater or equal than this threshold, this feature is linear. It will be considered non-linear otherwise. This is in fact a simplification of the actual threshold used in the implementation presented in Section 4. The actual definition, however, needs concepts that will be introduced later on this paper.

Size is partitioned into four categories: *large linear*, *small linear*, *large non-linear* and *small non-linear*. The linear sizes are determined according to a threshold on the maximum length of linear features (*maximum linear length thresh-*

old), whereas the non-linear sizes take into account thresholds on both: the *maximum non-linear length* and the *maximum non-linear width*³. The distinct size thresholds for linear and non-linear features is due to the intrinsic dissimilarity between judgement of sizes regarding objects in these two classes (for instance, a *small lake* does not necessarily have the same dimensions as a *small river*).

Flow is scaled as *flowing* or *stagnant* given a threshold on the average flow of the feature. Likewise, depth is partitioned into *deep* and *shallow* according to a threshold on depth. Finally, origin is partitioned into *natural* and *artificial*.

All of the scales used in this work partition the many valued attributes into mutually exclusive partitions, so called *nominal scales* [Ganter and Wille, 1999]. Other possibilities for scales could also be considered in this context depending on the level of granularity required for the attributes in the application domain. In this work, however, we are not going to get into details about the diverse possibilities of scales.

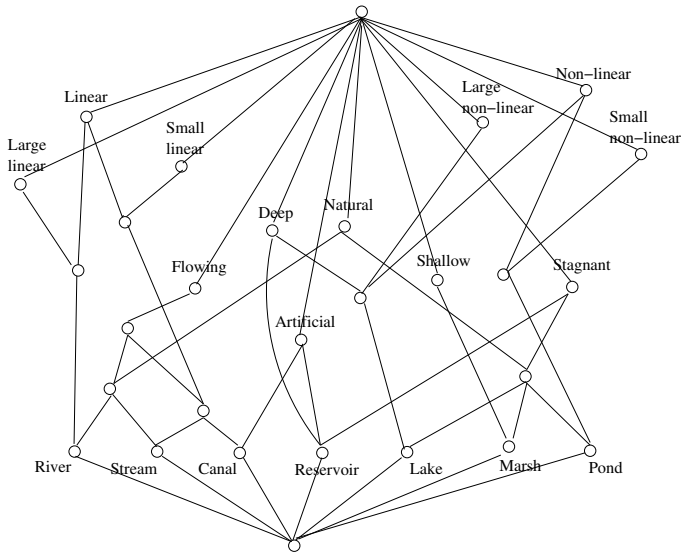


Figure 2: Concept lattice of inland water features.

The context of inland water features can, thus, be represented as the lattice shown in Figure 2.

Extracting definitions from the water feature formal context

The lattice in Figure 2 can be easily translated into definitions that could be handled by the supervaluation semantics, as discussed in Section 3 below.

In order to present the definitions of water features we first introduce the following functions on features. Let X be a water feature, then $linearity_ratio(X)$, $average_flow(X)$, $average_depth(X)$, $max_length(X)$ and $max_width(X)$ are mappings from the feature X to real values representing, respectively, its linearity ratio, average flow, average depth, maximum length and maximum width. With these functions,

³It is worth pointing out that, for clarity, we maintained *linear* and *non-linear* along with *large (small) linear/non-linear*, even though the related attributes are never used separately in this work.

and appropriate thresholds, it is now possible to define the attributes of a inland water feature (F), as presented in Definition 1⁴.

Definition 1 (Relations of the domain water feature)

- $Non_linear(F) \equiv linearity_ratio(F) < linearity_thresh$
- $Linear(F) \equiv linearity_ratio(F) \geq linearity_thresh$
- $Large_linear(F) \equiv linear(F) \wedge max_length(F) \geq linear_length_thresh$
- $Small_linear(F) \equiv linear(F) \wedge max_length(F) < linear_length_thresh$
- $Large_non_linear(F) \equiv linear(F) \wedge max_length(F) \geq nonlinear_length_thresh \wedge max_width(F) \geq nonlinear_width_thresh$
- $Small_non_linear(F) \equiv non_linear(F) \wedge maximum_length(F) < nonlinear_length_thresh \wedge maximum_width(F) < nonlinear_width_thresh$
- $Flowing(F) \equiv average_flow(F) \geq flow_thresh$
- $Stagnant(F) \equiv average_flow(F) < flow_thresh$
- $Deep(F) \equiv average_depth(F) \geq depth_thresh$
- $Shallow(F) \equiv average_depth(F) < depth_thresh$
- $Natural(F) \equiv \neg Artificial(F)$

In this paper we do not stipulate the actual thresholds for Definition 1. Future research may use machine or user surveying to determine appropriate default values corresponding to an intuitive classification.

We now can define the inland water features, as presented in Definition 2.

Definition 2 (Inland Water Features) Let each of the objects of the inland water feature context be a predicate symbol. The terms of the ontology are defined as follows.

- $\{River\}' \equiv \{Linear, Large_linear, Flowing, Natural\}$
- $\{Stream\}' \equiv \{Linear, Small_linear, Flowing, Natural\}$
- $\{Canal\}' \equiv \{Linear, Small_linear, Flowing, Artificial\}$
- $\{Reservoir\}' \equiv \{Stagnant, Artificial, Deep\}$
- $\{Marsh\}' \equiv \{Stagnant, Natural, Shallow\}$
- $\{Lake\}' \equiv \{Non_linear, Large_non_linear, Stagnant, Natural\}$
- $\{Pond\}' \equiv \{Non_linear, Small_non_linear, Stagnant, Natural\}$

The next section introduces the supervaluation semantics and discusses how it can handle variations in the meaning of the definitions above.

⁴In this paper predicate symbols start with an uppercase letter whereas functions are written with lower case letters.

3 Supervaluation Semantics

The formalisation of the inland water feature domain presented above used thresholds to define partitions on the values of attributes. However the implications of varying the values of these thresholds is not accounted for by the formal concept analysis. In fact, distinct sets of values for the thresholds may result in distinct (sometimes incompatible) definitions of the context objects. It is desirable to view such variations as *standpoints* on the vague definitions of features. We assume the supervaluation semantics [Fine, 1975] to provide a formal framework for standpoints on feature definition.

Supervaluation semantics views a vague language as a set of distinct precise versions of itself. Each of these versions is called a *precisification*. Formally, each precisification p is identified with a particular interpretation I_p of the language. A *supervaluation model* is defined as a set of precisifications. Therefore, given a supervaluation model Υ we can talk about propositions that are unequivocally true (i.e. are true in *every* interpretation $I_p \in \Upsilon$) and propositions that are *in some sense* true (i.e. are true in *some* interpretation $I_p \in \Upsilon$).

In this work the scales applied to attributes of the inland water feature domain compose one particular precisification. Therefore, the definitions of the inland water feature domain may vary according to the multiple meanings that the vague concepts might assume, or multiple values that the thresholds can assume. For instance, the two possible interpretations for the scenario in Figure 1 correspond to two distinct precisifications whereby the particular linearity thresholds dictate whether the scenario depicts three lakes or a meandering river. In fact, the apparatus of supervaluation semantics allows for logical relationships between vague concepts to be represented by quantifying over the (possibly infinite) space of precisifications [Bennett, 2001a].

Moreover, attributes in formal concepts can be used as predicates in the bodies of Horn clauses and the context objects as arguments in the clause heads of logic-based axioms. A set of such clauses can be fed in a system allowing logic reasoning about elements of the ontology. In other words, the methods of generating concept lattices provided by the formal concept analysis can be adapted to generate logical axioms that could be used within a logic-based reasoning system encoding supervaluation semantics.

The next section presents a prototype implementation of these ideas.

4 Implementation

In this section we discuss a prototype implementation of the inland water feature ontology with supervaluation semantics. The purpose of this prototype is to automatically individuate and classify different geographical features within two-dimensional images of river networks (e.g. river stretches, lakes and ponds). In a general way, the process of individuation and classification of water features involves the construction of the medial axis (or skeleton) of the river network and further translate it into a data structure. This data structure provides the basis for feature segmentation/classification. The elements involved in this process are described below.

4.1 Individuation and classification of features

By individuation in the context of water features we mean the process of identifying qualitative distinct stretches of river networks. In this work, this distinction is accomplished in three phases. First, the medial axis (or skeleton) of a river network is obtained by an off-the-shelf image processing method. Second, the axis thus obtained is processed into a data structure from which qualitative distinct features from the network can be identified by observing changes in the width along the axis. Finally, the distinct stretches are classified according to the definitions proposed in Sections 2.2 and 3 above.

Skeletonisation and data structure

The medial axis (or *skeleton*) [Blum, 1973] is commonly defined as the locus of the diameters of all maximal discs enclosed in a Jordan curve⁵ that touch at least two points of the curve's boundary. The best known application of skeletonisation is in the field of image processing, where the skeletons of object's images are used as an abstraction of the two dimensional shape of the objects depicted, as surveyed in [Loncaric, 1998]. In fact, a linear algorithm has been proposed that computes the medial axis of simple polygons⁶ in linear time [Chin *et al.*, 1995].

The idea of representing river networks by their medial axis is not new. To the best of our knowledge the first work that attempts such endeavour is [McAllister and Snoeyink, 2000], where the skeletons of river networks are used to identify opposite points in river banks, to connect the centreline of the networks with original river bank data, to calculate the areas of rivers and, finally, to automatically obtain the ordering of tributaries along the network. The problem of using the skeletons for the individuation and classification of water features (such as the identification of rivers and lakes in a single river network as proposed in the present paper), however, was only hinted in that work. We believe that the tasks of segmentation and classification of such features cannot be accomplished without a precise and consistent body of definitions such as those presented in sections 2.2 and 3.

In this work, we use a divergence-based method [Dimitrov *et al.*, 2000] as an off-the-shelf method for extracting the medial axis of river network snapshots. This method is based on considering a 2D region, representing the image of a shape boundary, as gradient fields of the region's distance transform. The skeletal points are, thus, identified as singularities in the *net outward flux* of the vector field relative to the region where each pair of points in opposite borders shrinks to a point.

Skeletons produced by the divergence-based method are transformed into an 8-connected list whereby each cell is subdivided into 9 registers. These registers encode each skeletal point in its middle register and pointers to any of its 8 cardinal neighbours. It is worth pointing out that, along with the skeletal point, the middle register also encodes the distance from this point to the nearest point in the shape's boundary (thus providing a measurement of the width of the water feature).

⁵A Jordan curve is any closed curve that does not intersect itself.

⁶A simple polygon is a single closed polygonal chain that do not intersect itself [de Berg *et al.*, 1997].

The task now is to sweep the 8-connected list, segmenting the river-network skeleton into stretches. These stretches are, thus, classified according to appropriate thresholds based on the water-feature ontology.

Integrated Classification and Individuation

Points where the skeleton branches and points where there are abrupt changes in the width of a river network (according to a threshold) are used to provide a means of obtaining a *proto-segmentation* of the data into *candidate water features*. The initial set of segments may further be concatenated or split according to the definitions in the ontology. This provides, as a result, the segmentation of the river network into water features, according to formal definitions.

In order to summarise these ideas in an algorithm we first need to introduce the ideas of *branching point*, *sample stretch* and *nexus point*.

A branching point is a discontinuity in the orientation of the boundary of a shape⁷. In a skeleton encoded as a 8-connected list, branching points are identified as a cell with more than two neighbours.

A *sample stretch* of a skeleton at a given point p is an ordered set of skeletal points with p as the mid-point and which extends a certain number of points either side of p . The *sample stretch ratio* is the ratio of the length of the sample stretch to the width of the water body at the sample point (i.e. the width at the mid-point of the stretch). We have found that a reasonable segmentation can be obtained by taking the sample stretch ratio as 1, so that the stretch has the same length as the width of the river.

We call a *nexus point* a point where two or more linear chains in the medial skeleton meet.

We can then define an (approximately) *linear sample stretch* of water as one where the minimum and maximum width along the stretch lie within a certain variation threshold relative to the width at the mid-point. More specifically a sample stretch s is linear according to a given *stretch linearity ratio threshold* (SLR_thresh) just in case:

$$(mid_width(s)/min_width(s) < SLR_thresh) \wedge (max_width(s)/mid_width(s) < SLR_thresh) \quad (1)$$

The lake-river segmentation and classification algorithm can be summarised as follows:

ALGORITHM: Lake-river segmentation and classification

INPUT: A bitmap depicting a river network

OUTPUT: A set of segments of the river network classified according to the water-feature ontology

- first, a matrix output by the divergence-based skeletonisation is converted to a set of Prolog facts – one for each point in the skeleton. These identify the coordinates of the skeleton point and its distance from the nearest edge point of the water body;

⁷Branching points are, in fact, *first-order shocks* as defined in [Siddiqi and Kimia, 1996].

- Prolog predicates are defined that test whether a point is an end point, a nexus point or a mid point of a chain. This is simply done by counting the number of points of the surrounding 8 pixes that are also skeleton points;
- For each chain of mid points the algorithm checks that the point is on a chain that extends for a distance in each direction that is at least equal to the width at that point;
- a list of points corresponding to the sample stretch is collected;
- Maximum and minimum widths along the sample stretch are computed;
- The ratio of the maximum and minimum widths to the width of the mid point are computed;
- If the ratio's lie within the given *stretch linearity ratio threshold* (according to statement 1) the point is marked as a point on a linear stretch; otherwise, it is marked as belonging to a non-linear segment.
- In the case of a linear segment, the linear-length/width thresholds are taken into account to decide whether the given segment is a river or a stream; in the case of a non-linear segment, the non-linear size thresholds are used to decide between a lakes or a pond.

Changes in the stretch linearity ratio threshold (and on the linear and non-linear size thresholds) allow for variations of the feature segmentation and classification. This provides a rigorous account of multiple standpoints in the classification of river, lakes, stream and pond features in a river network.

4.2 Prolog Implementation

The algorithm outlined in the previous section has been partially implemented in Prolog.

The following screen dump shows the output for an artificial example which was constructed to illustrate some key problems of lake/river discrimination. Note that the example contains a wide river and a much narrower river that could be interpreted either as a stream or as a river depending of an observer's standpoint. Moreover, the narrow river is broken by a small lake (or pond) and in this case the small lake/pond has a maximum diameter which is much smaller than the width of the wide river.

Figure 3 presents an output of the prototype implementation whereby linear stretches are indicated by thick black lines and non-linear parts of medial skeleton are shown as thin gray lines. A new classification, according to a distinct standpoint, is obtained by simply providing a new set of thresholds. Junctions between distinct rivers, however, were misclassified by the prototype, as shown by the junction between the thin and the wide rivers in Figure 3. [McAllister and Snoeyink, 2000] presents a possible solution for this problem, which shall be further exploited within the framework proposed in the present paper.

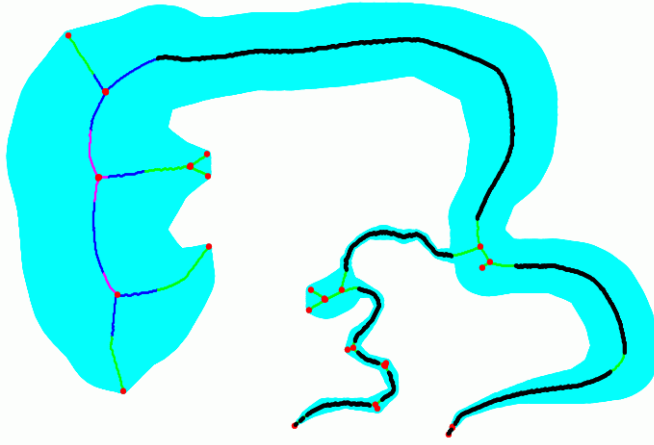


Figure 3: River stretch identification using sample stretches with sample length ratio 1 and sample variance ratio 1.5.

5 Further Work

Several aspects of the proposed ontology require further work to provide a more discriminating and flexible conceptualisation. In particular the articulation of size thresholds on water bodies requires further refinement to identify the relevant metrics. For instance, area and width measures should be distinguished.

More importantly, further work needs to be done on the integration between *individuation* and *classification*, which is only sketched in the present work. It will probably be necessary to introduce a further layer of definitions that allow features to be constructed by aggregating several components. Specifically, we would like to define the way in which a ‘river’ may be composed from several river stretches which may be joined in complex ways. An adequate treatment of this may require information on water flow to be integrated within the classification.

6 Conclusion

In this paper we presented an ontology for inland water features constructed using formal concept analysis and supervaluation semantics. The former allows a rigorous account of the domain structure by means of a complete lattice of concepts and their attributes. The latter formalises multiple interpretations of ontology terms owing to vagueness in the relative concept. The key for the integration of formal concept analysis and supervaluation semantics is the process of scaling. Scaling partitions many-value attributes of features into one-value ones, which permits a lattice representation of the domain. These partitions are defined by means of thresholds on the attribute values. Supervaluation semantics is used in this work to formalise variations on these thresholds modelling multiple possible interpretations of vague terms. Analysis of the hydrographic domain brings out the fact that vagueness not only affects the definitions of features but also how these features are individuated.

We proposed an algorithm for individuation and classification of water features according to the ontology presented.

The basic idea of this algorithm is to use the medial axis of a river network (extracted via an off-the-shelf divergence-based skeletonisation algorithm) to identify stretches of the network where there is relatively small variation in width. I.e., the process is to look at the variance along the axis checking that the maximum and minimum width values lie within some percentage variation threshold. This threshold can be regarded a supervaluation semantics parameter, and may vary according to any particular interpretation of the feature. This interaction between supervaluation semantics and the actual algorithmic feature individuation is the main contribution of this paper.

Acknowledgements

This work was partially supported by the Ordnance Survey.

Thanks to the five anonymous reviewers for the constructive criticisms on this paper.

References

- [Bennett, 2001a] B. Bennett. Application of supervaluation semantics to vaguely defined spatial concepts. In D.R. Montello, editor, *Spatial Information Theory: Foundations of Geographic Information Science; Proceedings of COSIT’01*, volume 2205 of *LNCS*, pages 108–123, Morro Bay, 2001. Springer.
- [Bennett, 2001b] B. Bennett. What is a forest? on the vagueness of certain geographic concepts. *Topoi*, 20(2):189–201, 2001.
- [Blum, 1973] H. Blum. Biological shape and visual science. *Journal of Theoretical Biology*, 38:205–287, 1973.
- [Chin et al., 1995] F. Chin, J. Snoeyink, and C.-A. Wang. Finding the medial axis of a simple polygon in linear time. In *Proc. of the 6th Annual International Symposium of Algorithms and Computation*, volume 1004 of *LNCS*, pages 382–391. Springer Verlag, 1995.
- [de Berg et al., 1997] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computation Geometry*. Springer Verlag, 1997.
- [Dimitrov et al., 2000] Pavel Dimitrov, Carlos Phillips, and Kaleem Siddiqi. Robust and efficient skeletal graphs. In *Proc. CVPR*, June 2000.
- [Elkan, 1993] C. Elkan. The paradoxical success of fuzzy logic. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-93)*, pages 698–703, 1993.
- [Fine, 1975] K. Fine. Vagueness, truth and logic. *Synthese*, 30:263–300, 1975.
- [Ganter and Wille, 1999] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer Verlag, 1999.
- [Loncaric, 1998] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.
- [McAllister and Snoeyink, 2000] Michael McAllister and Jack Snoeyink. Medial axis generalization of river networks. *CaGIS*, 27(2):129–138, 2000.
- [Siddiqi and Kimia, 1996] K. Siddiqi and B. Kimia. Toward a shock grammar for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

Aspects of Distributed and Modular Ontology Reasoning

Luciano Serafini
ITC-IRST
Via Sommarive 18
38050 Trento, Italy
serafini@itc.it

Alex Borgida
Department of Computer Science
Rutgers University
Piscataway, NJ 08855, USA
borgida@cs.rutgers.edu

Andrei Tamilin
DIT - University of Trento
Via Sommarive 14
38050 Trento, Italy
tamilin@dit.unitn.it

Abstract

We investigate a formalism for reasoning with multiple local ontologies, connected by directional semantic mappings. We propose: (1) a relatively small change of semantics which localizes inconsistency (thereby making unnecessary global satisfiability checks), and preserves directionality of “knowledge import”; (2) a characterization of inferences using a fixed-point operator, which can form the basis of a cache-based implementation for local reasoners; (3) a truly distributed tableaux algorithm for cases when the local reasoners use subsets of $\mathcal{SHIQ}$. Throughout, we indicate the applicability of the results to several recent proposals for knowledge representation and reasoning that support modularity, scalability and distributed reasoning.

1 Introduction

In applications ranging from information and service integration to the semantic web, it is expected that ontologies will be fragmented and distributed, rather than monolithic. Moreover, there are well-known advantages to breaking up a large ontology into modules (e.g., see [9]). Recently, there have been a number of proposals for systems supporting modularized ontologies, where one can have a localized view either by “importing” specific information [9; 3] or by distributing reasoning [8; 4; 10].

Distributed Description Logics (DDLs) [2] are a particular formalization of the notion of distributed ontology and reasoning, which starts from local T-boxes, and expresses semantic connections between them as “bridge rules” between pairs of connects. This paper contributes to the study of DDLs, with the hope of shedding light both on the semantic and the implementation aspects of *all* similar mechanisms.

In particular, the claimed contributions of the paper are three-fold. First, on the semantic front, we propose a new semantics for DDLs, which better captures the properties of *localized inconsistency* and *directionality*. According to the directionality property, semantic mappings have a direction from a source ontology to a target ontology, and support knowledge propagation only in such a direction. According

to localized inconsistency, the inconsistency in one component T-box, or in some subgroup of connected components, should not automatically render the *entire* system inconsistent. These properties are desirable for all multi-module ontologies, and we discuss how our results apply to other proposals, such as [9; 3; 8].

Second, we examine an approach which views the bridge rules connecting two local ontologies as describing an *operator* that propagates knowledge in the form of DL subsumption axioms. This is used as the basis of a characterization of distributed DL reasoning using a fixed point operator, which does forward-propagation of axioms. In addition to its intrinsic interest, as part of a syntactic characterization of the DDL consequence relation, this result can be used to extend the caching technique described in [9] to a sound and complete one.

The third contribution is a sound and complete distributed tableaux algorithm that determines the satisfiability of a $\mathcal{SHIQ}$ concept in the context of the local axioms of an ontology and the extra knowledge imparted by the bridge rules. This algorithm is the basis of a peer-to-peer implementation of a proof mechanism for checking concept subsumption in a distributed manner.

The rest of the paper is organized as follows. In Section 2 we review the formal framework of DDLs. In Section 3 we introduce the notion of inconsistent local interpretation that allows us to model partially inconsistent distributed ontologies, and show exactly how the new and old entailments are related. The new semantics is also used in Section 4 to characterize the knowledge flow induced by bridge rules, and in Section 5 to present a sound and complete distributed tableaux algorithm that computes satisfiability/subsumption.

2 Distributed Description Logics

We briefly recall the definition of DDL as given by Borgida and Serafini [2].

Syntax Given a non-empty set I of indexes, used to enumerate local ontologies, let $\{\mathcal{DL}_i\}_{i \in I}$ be a collection of description logics.¹ For each $i \in I$, let us denote a T-box of $\mathcal{DL}_i$ as

¹We assume the reader is familiar with description logics and related reasoning systems, as described in [1].

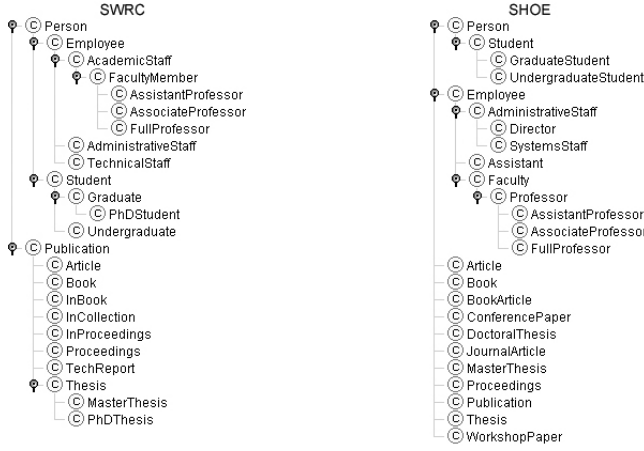


Figure 1: Extracts of the class hierarchies

$\mathcal{T}_i$.² To make every description D distinct, we will prefix it with the index of ontology it belongs to, as in $i : C$. We use $i : C \sqsubseteq D$ to say that $C \sqsubseteq D$ is being considered in the i -th ontology.

Semantic mappings between different ontologies are expressed via *bridge rules*. A bridge rule from i to j is an expression, which in this paper is restricted to being one of the following two forms:

$i : x \xrightarrow{\sqsubseteq} j : y$ — an *into-bridge rule*

$i : x \xrightarrow{\sqsupseteq} j : y$ — an *onto-bridge rule*

where x and y are concepts. The derived bridge rule $i : x \xrightarrow{\equiv} j : y$ can be defined as the conjunction the corresponding into and onto bridge rule.

Bridge rules from i to j express relations between i and j viewed from the *subjective* point of view of the j -th ontology.

For example, the into-bridge rule $i : C \xrightarrow{\sqsubseteq} j : D$ intuitively says that, from the j -th point of view, the individuals in concept C in i correspond (via an approximation introduced by an implicit semantic domain relation) to a subset of the individuals in its local concept D . Therefore, bridge rules from i to j provide the possibility of translating into j 's ontology some of the concepts of a foreign ontology i .

A *distributed T-box (DTBox)* $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \mathfrak{B} \rangle$ therefore consists of a collection of T-boxes $\{\mathcal{T}_i\}_{i \in I}$, and a collection of bridge rules $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i \neq j \in I}$ between them.

Example 2.1 Figure 1 shows fragments of class hierarchies from two ontologies, SWRC³ and SHOE⁴, available from the DAML on-line library. These can be viewed as local T-boxes. For the sake of demonstrating the value of mappings, we considered oversimplified SHOE ontology without imports. The

²We assume that a T-box will contain all the information necessary to define the terminology of a domain, including not just concept and role definitions, but also general axioms relating descriptions, as well as declarations such as the transitivity of certain roles. This is in keeping with the intent of the original paper introducing the terms T-box and A-box.

³www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml

⁴www.cs.umd.edu/projects/plus/DAML/onts/univ1.0.daml

following are examples of bridge rules from SWRC to SHOE:

$$\text{SWRC} : \text{Publication} \xrightarrow{\equiv} \text{SHOE} : \text{Publication} \quad (1)$$

$$\text{SWRC} : \text{InProceedings} \xrightarrow{\sqsubseteq} \text{SHOE} : \text{ConferencePaper} \sqcup \text{WorkshopPaper} \quad (2)$$

$$\text{SWRC} : \text{InBook} \xrightarrow{\sqsupseteq} \text{SHOE} : \text{BookArticle} \quad (3)$$

Semantics DDL semantics is a customization of the Local Models Semantics for Multi Context Systems [5; 6]. Each ontology $\mathcal{T}_i$ is *locally interpreted* by a standard DL interpretation $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, \mathcal{I}_i \rangle$. Since local domains may be heterogeneous (e.g., time may be represented by Rationals and Integers in two ontologies), we need relations that model semantic correspondences between heterogeneous domains. A *domain relation* r_{ij} from $\Delta^{\mathcal{I}_i}$ to $\Delta^{\mathcal{I}_j}$ is a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. For example, if $\Delta^{\mathcal{I}_1}$ and $\Delta^{\mathcal{I}_2}$ are the Rationals and the Naturals, then r_{12} could be the round-off function. We use $r_{ij}(d)$ to denote $\{d' \in \Delta^{\mathcal{I}_j} \mid \langle d, d' \rangle \in r_{ij}\}$; for $D \subseteq \Delta^{\mathcal{I}_i}$, we use $r_{ij}(D)$ for $\bigcup_{d \in D} r_{ij}(d)$.

A *distributed interpretation* $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ of a DTBox $\mathfrak{T}$ therefore combines the above two notions and is said to satisfy (written $\mathfrak{I} \models_d \mathfrak{T}$) the elements of $\mathfrak{T}$ if

1. $\mathcal{I}_i \models_d A \sqsubseteq B$ for all $A \sqsubseteq B$ in $\mathcal{T}_i$
2. $\mathfrak{I} \models_d i : x \xrightarrow{\sqsubseteq} j : y$, if $r_{ij}(x^{\mathcal{I}_i}) \subseteq y^{\mathcal{I}_j}$
3. $\mathfrak{I} \models_d i : x \xrightarrow{\sqsupseteq} j : y$, if $r_{ij}(x^{\mathcal{I}_i}) \supseteq y^{\mathcal{I}_j}$
4. $\mathfrak{I} \models_d \mathfrak{T}$, if for every $i, j \in I$, $\mathfrak{I} \models_d \mathcal{T}_i$ and $\mathfrak{I} \models_d \mathfrak{B}_{ij}$

Finally, $\mathfrak{T} \models_d i : C \sqsubseteq D$ (read as “ $\mathfrak{T}$ d-entails $i : C \sqsubseteq D$ ”) if for every $\mathfrak{I}$, $\mathfrak{I} \models_d \mathfrak{T}$ implies $\mathfrak{I} \models_d i : C \sqsubseteq D$. We say $\mathfrak{T}$ is *satisfiable* if there exists a $\mathfrak{I}$ such that $\mathfrak{I} \models_d \mathfrak{T}$. Concept $i : C$ is *satisfiable* with respect to $\mathfrak{T}$ if there is a $\mathfrak{I}$ such that $\mathfrak{I} \models_d \mathfrak{T}$ and $C^{\mathcal{I}_i} \neq \emptyset$.

On injective domain correspondences A key novelty of the semantic mappings in DDL is support for multiple levels of granularity and perspective: allowing individuals to be related using arbitrary binary relations rather than just bijection. For example, while it is traditional to state correspondences such as “Wife in Ontology 1 corresponds to Moglie in Ontology 2”, DDLs support domain correspondences that are general binary relations, so that one can say that “Husband and Wife in ontology 1 correspond to Couple in Ontology 2”, which can be formalized by using onto-bridge rules $\{1 : \text{Wife} \xrightarrow{\sqsupseteq} 2 : \text{Couple}, 1 : \text{Husband} \xrightarrow{\sqsupseteq} 2 : \text{Couple}\}$. In [4], DDLs are faulted because the collection of bridge rules $\{1 : \text{Bird} \xrightarrow{\sqsupseteq} 2 : \text{Penguin}, 1 : \neg \text{Fly} \xrightarrow{\sqsupseteq} 2 : \text{Penguin}\}$ do not render Penguin unsatisfiable even if Bird is subsumed by Fly in Ontology 1. As the example involving Couple shows, the general formal pattern is correct in some cases, so this is actually a problem of incomplete modeling.

In the case of Penguins, the extra information is that the domain relation is one-to-one. In such cases, one should also have added bridge rules stating that *non-birds* and *flying objects* are *non-penguins*: $\{1 : \neg \text{Bird} \xrightarrow{\sqsubseteq} 2 : \neg \text{Penguin}, 1 : \text{Fly} \xrightarrow{\sqsubseteq} 2 : \neg \text{Penguin}\}$. This would indeed lead to the conclusion $\mathfrak{T} \models_d 2 : \text{Penguin} \sqsubseteq \perp$.

Since the property that the domain relation is one-one over some concept B arises quite frequently, we might consider

adding a new kind of bridge rule to express it, writing something like $\xrightarrow{\leq 1}$ Penguin. Interestingly, it can be proven that in the context of DDLs, such rules can be eliminated by syntactically manipulating the DTBox, so that whenever $\xrightarrow{\leq 1} G$ and $A \xrightarrow{\geq} H$ are present, a new bridge rule $\neg A \xrightarrow{\leq} \neg(H \sqcap G)$ is added. The tableaux technique in Section 5 could however use such rules more efficiently without the encoding.

Properties and Desiderata for DDL We first give some basic ways in which subsumption and a combination of onto- and into-bridge rules allows to propagate subsumptions across ontologies.

Lemma 2.1 *If $\mathfrak{B}_{ij}$ contains $i : A \xrightarrow{\geq} j : G$ and $i : B \xrightarrow{\leq} j : H$, then $\mathfrak{T} \models_d i : A \sqsubseteq B \implies \mathfrak{T} \models_d j : G \sqsubseteq H$*

Thus, in Example 2.1, the subsumption $SHOE : BookArticle \sqsubseteq Publication$ can be inferred in DDL through bridge rules (1) and (3), and the subsumption $InBook \sqsubseteq Publication$ contained in $\mathcal{T}_{SWRC}$.

If the local languages support disjunction as a concept constructor then a more general form of propagation can occur:

Lemma 2.2 *If $\mathfrak{B}_{ij}$ contains $i : A \xrightarrow{\geq} j : G$ and $i : B_k \xrightarrow{\leq} j : H_k$ for $1 \leq k \leq n$ (with $n \geq 0$), then $\mathfrak{T} \models_d i : A \sqsubseteq \bigsqcup_{k=1}^n B_k \implies \mathfrak{T} \models_d j : G \sqsubseteq \bigsqcup_{k=1}^n H_k$*

Additional properties would be desirable for DDL entailment. In particular, since the intended meaning is that bridge rules $\mathfrak{B}_{ij}$ constitute a semantic channel which allows ontology j to access and import knowledge from ontology i , we want information flow to be “directional” in some sense. To express this, we first introduce the notion of bridge graph.

Definition 2.1 *The bridge graph $G_{\mathfrak{T}}$ of a DTBox $\mathfrak{T}$ is a directed graph with an arc from i to j exactly when the set of bridge rules $\mathfrak{B}_{ij}$ is non-empty.*

We can then state the main property we are looking for as:

Directionality desideratum If in $G_{\mathfrak{T}}$ there is no path from i to j , then $\mathfrak{T} \models_d j : A \sqsubseteq B$ if and only if $\mathfrak{T}' \models_d j : A \sqsubseteq B$, where $\mathfrak{T}'$ is obtained by removing $\mathfrak{T}_i$, $\mathfrak{B}_{ki}$, and $\mathfrak{B}_{ik}$ from $\mathfrak{T}$.

This says that knowledge is propagated *only* through bridge rules, so that if there are no bridge rules that go from i towards j , then j is not affected by i . The following two isolation properties are special cases of this:

Isolation 1 A T-box without incoming bridge rules is not affected by other T-boxes. (Formally, if $\mathfrak{B}_{ki} = \emptyset$ for all $k \neq i \in I$, then $\mathfrak{T} \models_d i : A \sqsubseteq B \implies \mathfrak{T}_i \models A \sqsubseteq B$)

Isolation 2 A T-box without outgoing bridge rules does not affect the other T-boxes.

Unfortunately, property *Isolation 1* does not always hold, because of onto-rules. Indeed, in the presence of onto-rule $1 : A \xrightarrow{\geq} 2 : G$ from T-box $\mathcal{T}_1$ to $\mathcal{T}_2$, if $\mathcal{T}_2$ entails $\top \sqsubseteq G$, then $1 : A$ cannot be empty according to DDL semantics, and so, for example, an inconsistency would be generated if $\mathcal{T}_1$ entails $A \sqsubseteq \perp$. This is despite the fact that the bridge rules are toward $\mathcal{T}_2$.

Property *Isolation 2* may also not hold. Indeed, if $\mathcal{T}_1$ is unsatisfiable, then $\mathcal{T}_2 \models_d 2 : X \sqsubseteq Y$ for every X, Y , even if there are no bridge rules connecting $\mathcal{T}_1$ with $\mathcal{T}_2$, because there are no satisfying distributed interpretations at all. Note that in a DDL, inconsistency may arise in a connected group of T-boxes even if each T-box is locally consistent; e.g., consider the case in the hypothesis of Lemma 2.1, when $\mathcal{T}_j \models \top \sqsubseteq G$ and $\mathcal{T}_j \models H \sqsubseteq \perp$.

This is a significant problem, because a localized inconsistency spreads and contaminates reasoning in *all* other local ontologies, even in the absence of connections to them, because there will be no satisfying distributed interpretation, and hence every statement about them is true, as usual in logic. This problem plagues all modular and distributed representation systems.

In the following section we propose an extension of the initial semantics in order to fix this problem.

3 Inconsistency in DDL

There are a number of possible approaches to handle the problem of inconsistency propagation.

(1) Define d-entailment in a 2-step manner, first eliminating local T-boxes that are inconsistent, and then using the standard definition. The problem with this approach is that it is non-monotonic, and it does not deal with cases where the inconsistency arises due to several connected local sources.

(2) Use some variant of a multi-modal epistemic semantics, which allows for models of even inconsistent knowledge in the case when the set of accessible worlds is empty. Such an approach was used in [6] for Distributed First Order Logics, but its computational complexity/decidability aspects are quite worrisome, and the precise impact of such non-standard semantics on logical consequences is hard to explain in an intuitive manner to users.

(3) Introduce some special interpretation, called a “hole” in [2], whose role is to interpret even inconsistent local T-boxes. We pursue this latter option.

Definition 3.1 *A hole for a T-box $\mathcal{T}$ is an interpretation $\mathcal{I}^\epsilon = \langle \emptyset, \cdot^\epsilon \rangle$, where the domain is empty.*

Of course, the important property of holes is that $\mathcal{I}^\epsilon \models X \sqsubseteq Y$ for every X and Y , since both sides are interpreted as the empty set.⁵ We will however continue to refer to T-boxes as “inconsistent/unsatisfiable” in case there are no interpretations *other* than $\mathcal{I}^\epsilon$ which satisfy all the axioms in it.

Let us extend the notion of d-entailment $\models_d$, obtaining the $\models_\epsilon$ relation, by also allowing holes as interpretations for local T-boxes. Note that now even if some local T-box $\mathcal{T}_i$ is inconsistent, we still have an interpretation for the whole DTBox: one that uses $\mathcal{I}^\epsilon$ to satisfy $\mathcal{T}_i$.

Properties of the semantics with holes First, the new semantics does the intended job:

Theorem 3.1 *The earlier-stated “directionality desideratum” holds for $\models_\epsilon$.*

⁵We are indebted to XXX for pointing out an earlier error, and to YYY referee for suggesting the empty domain.

Non-standard semantics (such as multivalued logics) can however distort the meaning of the original semantics in unpredictable ways. The following results should be reassuring in this respect.

For any $\mathcal{T}$ and any $i \in I$, let $\mathcal{T}(\epsilon_i)$ (the distributed T-box with the i -th local T-box viewed as inconsistent) be obtained by removing $\mathcal{T}_i$, $\mathcal{B}_{ij}$ and $\mathcal{B}_{ji}$ from $\mathcal{T}$, and by extending each $\mathcal{T}_j$ with the set of axioms $\{G \sqsubseteq \perp \mid i : A \xrightarrow{\sqsubseteq} j : G \in \mathcal{B}_{ij}\}$. For any finite set $J = \{i_1, \dots, i_n\}$, such that $J \subset I$, let $\mathcal{T}(\epsilon_J)$ be $\mathcal{T}(\epsilon_{i_1}) \dots (\epsilon_{i_n})$. (If J is empty $\mathcal{T}(\epsilon_J) = \mathcal{T}$.) The following then precisely characterizes the relationship of $\models_d$ and $\models_\epsilon$:

Proposition 3.1 $\mathcal{T} \models_\epsilon i : X \sqsubseteq Y$ if and only if for every subset $J \subseteq I$ not containing i , $\mathcal{T}(\epsilon_J) \models_d i : X \sqsubseteq Y$.

Moreover, in acyclic cases the relationship is even clearer:

Proposition 3.2 Let $\mathcal{T} = \langle \mathcal{T}_1, \mathcal{T}_2, \mathcal{B}_{12} \rangle$ be a DTBox. Then

- (i) if $\mathcal{T}_1$ is consistent, then for $j \in \{1, 2\}$, $\mathcal{T} \models_\epsilon j : X \sqsubseteq Y$ if and only if $\mathcal{T} \models_d j : X \sqsubseteq Y$.
- (ii) if $\mathcal{T}_1$ is inconsistent, then $\mathcal{T} \models_\epsilon 2 : X \sqsubseteq Y$ if and only if $\mathcal{T}_2 \cup \{G \sqsubseteq \perp \mid 1 : A \xrightarrow{\sqsubseteq} 2 : G \in \mathcal{B}_{12}\} \models_d X \sqsubseteq Y$

Application to Other Frameworks As noted earlier, the problem of local inconsistency polluting the inferences of all the modules in a modular representation is quite general. We examine how the approach presented here can be applied to two previously proposed schemes.

[9] proposes an elegant notion of modular ontology which starts from the semantic framework of DDLs, but restricts bridge rules to “identities” defining new local names $j : N$ using concepts $i : C$ from T-box i , modulo a semantic domain correspondence exactly like r_{ij} for DDLs.⁶ This can be modeled by replacing every definition $i : C \equiv j : N$ by the composed bridge rule $i : C \xrightarrow{\equiv} j : N$. Therefore the semantics involving holes introduced in the previous section can be applied to this approach, in order to localize inconsistencies in modules.

The Somewhere Peer Data Management System [8] consists of a collection of peers which maintain local ontologies, including repositories of “extensional data”. Peers are acquainted with neighbors, whose concepts they can use, and query processing involves intensional distributed reasoning for query rewriting. Since this reasoning is (semantically) based on a single global interpretation, it is subject to the above mentioned difficulties due to inconsistency. In fact, for completeness, a *global* consistency check, involving even unconnected peers, would be required. We would suggest adopting a distributed semantics with holes, such as that of DDL. In particular, current peer links in Somewhere can be reduced to subsumption expressions like $1 : C \sqcap 3 : D \sqsubseteq 2 : E$. A DDL can be constructed from this by replacing occurrences of $i : C$ in peer j by new, local symbol C_i , and adding

⁶Although [9] originally defines imported names using conjunctive queries over concepts and roles in T-box j , it then says that these can be “rolled up” into descriptions. Although this may in fact not always be doable, we will deal here with exactly those definitions for which this roll-up holds.

bridge rules $i : C \xrightarrow{\equiv} j : C_i$ and $i : \neg C \xrightarrow{\equiv} j : \neg C_i$. Our semantics then provides for directionality and locality, and the next section provides a distributed satisfiability testing algorithm for the semantics with holes.

Finally, the C-OWL [3] proposal for contextualized ontologies, uses a similar model theory as DDL. The only difference concerns the definition of hole, which, in C-OWL was defined on a non empty set. Those notion of hole supports directionality in all the cases except the case of presence of $i : \perp \xrightarrow{\sqsubseteq} j : \perp$ rule, which allows to propagate back inconsistency. The particular version of “holes” given in this paper gives to C-OWL the directionality property, in addition to the localized inconsistency it already had.

4 Fixed-point semantics of bridge rules

As we saw earlier, combinations of bridge rules allow the propagation of subsumptions across T-boxes. To better understand how this propagation happens, we will associate with a set $\mathcal{B}_{ij}$ of bridge rules an operator of the same name, which extends the j -th T-box with a set of subsumption axioms that are the transformation via bridge rules of subsumptions in the i -th T-box.

Before proceeding further, we need to introduce the concept of disjoint union for interpretations. To begin with, we define as usual the disjoint union of two, possibly overlapping sets S and T as $S \uplus T = (S \times \{\#\}) \cup (T \times \{\@\})$, where the values are distinguished by tupling with two discriminant symbols — $\#$ and $\@$, in this case. This is generalized to the disjoint union $\biguplus_{i \in K} S_i$ of a collection of sets $\{S_i\}_{i \in K}$ indexed with (possibly infinite) K , by using the indices i as the discriminants.

Definition 4.1 Given two interpretations $\mathcal{I} = \langle \Delta_{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle \Delta_{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ of the same language $\mathcal{L}$, the disjoint union of $\mathcal{I}$ and $\mathcal{J}$, denoted by $\mathcal{I} \uplus \mathcal{J}$, is $\langle \Delta_{\mathcal{I} \uplus \mathcal{J}}, \cdot^{\mathcal{I} \uplus \mathcal{J}} \rangle$, where:

1. $\Delta_{\mathcal{I} \uplus \mathcal{J}} = \Delta_{\mathcal{I}} \times \{\#\} \cup \Delta_{\mathcal{J}} \times \{\@\}$
2. for concept A , $(A)^{\mathcal{I} \uplus \mathcal{J}} = A^{\mathcal{I}} \times \{\#\} \cup A^{\mathcal{J}} \times \{\@\}$
3. for role R , $R^{\mathcal{I} \uplus \mathcal{J}} = \{ \langle (x, \#), (y, \#) \rangle \mid \langle x, y \rangle \in R^{\mathcal{I}} \} \cup \{ \langle (w, \@), (z, \@) \rangle \mid \langle w, z \rangle \in R^{\mathcal{J}} \}$

Disjoint union for interpretations $\biguplus_{k \in K} \mathcal{I}_k$ can similarly be generalized to the case of a sets. Intuitively the interpretation $\mathcal{I} \uplus \mathcal{J}$ is interpretation that is composed of two unrelated subparts one is isomorphic to $\mathcal{I}$ and the other to $\mathcal{J}$.

Definition 4.2 A description logic family $\mathcal{DL}$ has the disjoint union satisfiability property if $E^{\mathcal{I}'} = \biguplus_{k \in K} E^{\mathcal{I}_k}$ holds for all concepts and roles E over $\mathcal{DL}$, and for all interpretations $\mathcal{I}' = \biguplus_{k \in K} \mathcal{I}_k$.

Lemma 4.1 $\mathcal{SHIQ}$, and its sub-languages, have the distributed union satisfiability property.

On the other hand, languages that support nominals (such as OWL), or A-boxes do not have this property.

The bridge operator The bridge operator essentially applies generalized subsumption propagation Lemma 2.2, to find new subsumptions:

Definition 4.3 Given a set of bridge rules $\mathfrak{B}_{12}$ from DL_1 to DL_2 , the bridge operator $\mathfrak{B}_{12}(\cdot)$, taking as input a T-box in DL_1 and producing a T-box in DL_2 , is defined as follows:

$$\mathfrak{B}_{12}(\mathcal{T}_1) = \left\{ G \sqsubseteq \bigcup_{k=1}^n H_k \mid \begin{array}{l} \mathcal{T}_1 \models A \sqsubseteq \bigcup_{k=1}^n B_k, \\ 1 : A \xrightarrow{\exists} 2 : G \in \mathfrak{B}_{12}, \\ 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}, \\ \text{for } 1 \leq k \leq n, n \geq 0 \end{array} \right\}$$

(Notationally, $\bigcup_{k=1}^0 D_k$ denotes $\perp$.)

It is remarkable that these are essentially *all* the inferences that one can get, if we use the semantics with holes:

Theorem 4.1 Let $\mathcal{T}_{12} = \langle \mathcal{T}_1, \mathcal{T}_2, \mathfrak{B}_{12} \rangle$ be a distributed T-box. If DL_1 and DL_2 have the distributed union satisfiability property then:

$$\mathcal{T}_{12} \models_{\epsilon} 2 : X \sqsubseteq Y \iff \mathcal{T}_2 \cup \mathfrak{B}_{12}(\mathcal{T}_1) \models X \sqsubseteq Y$$

For any family $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i,j \in I}$ of bridge rules, we can combine these into a new operator $\mathfrak{B}$ on a family of T-boxes as follows:

$$\mathfrak{B}(\{\mathcal{T}_i\}_{i \in I}) = \left\{ \mathcal{T}_i \cup \bigcup_{j \neq i} \mathfrak{B}_{ji}(\mathcal{T}_j) \right\}_{i \in I}$$

If I is finite and each $\mathfrak{B}_{ij}$ is finite, then there is a positive integer b such that for every family of T-boxes $\mathbf{T}$, $\mathfrak{B}^b(\mathbf{T}) = \mathfrak{B}^{b+1}(\mathbf{T})$. Let us then define $\mathfrak{B}^*(\mathbf{T})$ as $\mathfrak{B}^b(\mathbf{T})$, where b is the first positive integer such that $\mathfrak{B}^b(\mathbf{T}) = \mathfrak{B}^{b+1}(\mathbf{T})$. Furthermore let $\mathfrak{B}^{b+1}(\mathbf{T})_i$, be the i -th T-box in $\mathfrak{B}^{b+1}(\mathbf{T})$.

Theorem 4.2 For every $\mathcal{T} = \langle \mathbf{T}, \mathfrak{B} \rangle$, $\mathcal{T} \models_{\epsilon} j : X \sqsubseteq Y$ if and only if the j -th T-box of $\mathfrak{B}^*(\mathbf{T})$ entails $X \sqsubseteq Y$.

Applications to Caching A number of researchers have considered the idea of caching locally the necessary information from the imported ontology $\mathcal{T}_{other}$, since this is assumed to be both more efficient (there is no need to interrupt local reasoning, while waiting for answers from the other ontology), and more prespicuous from the point of view of the local user: in order to understand an imported concept F , it is not necessary to understand *all* of $\mathcal{T}_{other}$, only the locally cached part, which is presumed to be much smaller. (This idea is also known as “subsetting”, and there is considerable research on this topic in the ontology community.)

Theorem 4.2 above indicates that it is possible to finitely pre-compile in a sound and complete manner the subsumption information imported into a local ontology $\mathcal{T}_j$ by the bridge rules in a DTBox $\mathcal{T}$: compute and store it.

In a similar vein, [9] takes the set of imported concept definitions $\{N_k \equiv other : D_k \mid k = 1, \dots, n\}$, and then computes and caches the subsumption hierarchy of the $\{N_k\}$. Since we have explained in Section 3 that the module mechanism in [9] can be represented as a DDL, Lemma 2.2 indicates that if the language contains at least $\mathcal{ALC}$, and if it is possible to ask subsumption queries about complex concepts composed using the imported definitions, then it is not sufficient to cache only subsumptions of the form $D_{k1} \sqsubseteq D_{k2}$, since there may be additional subsumptions entailed, involving disjunctions. On the other hand, by Theorem 4.2 it is *sufficient* to cache all subsumptions of the form

$N_k \sqsubseteq N_{k1} \sqcup \dots \sqcup N_{km}$, whose definitions satisfy the condition⁷ $\mathcal{T}_{other} \models D_k \sqsubseteq D_{k1} \sqcup \dots \sqcup D_{km}$.

5 A distributed tableaux algorithm for DDL

In this section we describe a tableaux-based decision procedure for $\mathcal{T} \models_{\epsilon} i : X \sqsubseteq Y$, for DTBoxes whose bridge graph $G_{\mathcal{T}}$ is acyclic. The cyclic case is left for future work, pending the identification of a loop blocking strategy that preserves the independence of the local proofs.

To simplify the description, we suppose that local ontologies are expressed in (a subset of) the *SHIQ* language — one of the most widely known DLs. Also, we will assume that the consequences of bridge rules are atomic names. (This condition can easily be achieved by introducing, through definitions, names for the consequent concepts.) We need the usual notion of axiom internalization, as in [7]: given a T-box $\mathcal{T}_i$, the concept $C_{\mathcal{T}_i}$ is defined as $C_{\mathcal{T}_i} = \bigcap_{E \sqsubseteq D \in \mathcal{T}_i} \neg E \sqcup D$; also, the role hierarchy $R_{\mathcal{T}_i}$ contains the role axioms of $\mathcal{T}_i$, plus additional axioms $P \sqsubseteq U$, for each role P of $\mathcal{T}_i$, with U some fresh role.

The algorithm for testing j -satisfiability of a concept expression X (i.e., checking $\mathcal{T} \not\models_{\epsilon} j : X \sqsubseteq \perp$) builds, as usual, a finite representation of a distributed interpretation $\mathcal{I}$, by running local *autonomous SHIQ* tableaux procedures to find each local interpretation $\mathcal{I}_i$ of $\mathcal{I}$.

Definition 5.1 For each $j \in I$, the function **DTab_j** takes as input a concept X and tries to build a representation of $\mathcal{I}_j$ with $X^{\mathcal{I}_j} \neq \emptyset$ (called a completion tree [7]) for the concept $X \sqcap C_{\mathcal{T}_j} \sqcap \forall U.C_{\mathcal{T}_j}$, using the *SHIQ* expansion rules, w.r.t. the role hierarchy $R_{\mathcal{T}_j}$, plus the following additional “bridge” expansion rules

Unsat- $\mathfrak{B}_{ij}$ -rule

- if 1. $G \in \mathcal{L}(x)$, $i : A \xrightarrow{\exists} j : G \in \mathfrak{B}_{ij}$, and
 2. $\text{IsSat}_i(A \sqcap \neg \bigcup \mathbf{B}') = \text{False}$, for some $\mathbf{H}' \not\sqsubseteq \mathcal{L}(x)$,
- then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\bigcup \mathbf{H}'\}$

New- $\mathfrak{B}_{ij}$ -rule

- if 1. $G \in \mathcal{L}(x)$, $i : A \xrightarrow{\exists} j : G \in \mathfrak{B}_{ij}$, and
 2. $\mathbf{B} \subseteq \{B \mid i : B \xrightarrow{\sqsubseteq} j : H \in \mathfrak{B}_{ij}\}$, and
 3. for no $\mathbf{B}' \subseteq \mathbf{B}$ is $\text{IsSat}_i(A \sqcap \neg \bigcup \mathbf{B}') = \text{False}$, and
 4. for no $\mathbf{B}' \supseteq \mathbf{B}$ is $\text{IsSat}_i(A \sqcap \neg \bigcup \mathbf{B}') = \text{True}$,
- then if **DTab_i**($A \sqcap \neg \bigcup \mathbf{B}$) = Satisfiable
then $\text{IsSat}_i(A \sqcap \neg \bigcup \mathbf{B}) = \text{True}$
else $\text{IsSat}_i(A \sqcap \neg \bigcup \mathbf{B}) = \text{False}$

The idea, inspired by bridge operator $\mathfrak{B}_{ij}(\cdot)$, is that whenever **DTab_j** encounters a node x that contains a label G which is a consequence of an onto-bridge rule, then if $G \sqsubseteq \bigcup \mathbf{H}$ is entailed by the bridge rules, the label $\bigcup \mathbf{H}$, is added to x . To determine if $G \sqsubseteq \bigcup \mathbf{H}$ is entailed by bridge rules $\mathfrak{B}_{ij}$, **DTab_j** invokes **DTab_i** on the satisfiability of the concept $A \sqcap \neg (\bigcup \mathbf{B})$. **DTab_i** will build (independently from **DTab_j**) an interpretation $\mathcal{I}_i$, as illustrated in Figure 2. To avoid redundant calls, **DTab_j** caches the calls to **DTab_i** in a data structure IsSat_i ,

⁷There is no need to iterate if we assume that imported names cannot be used in additional axioms of the local ontology — only for labeling information on the semantic web, for example.

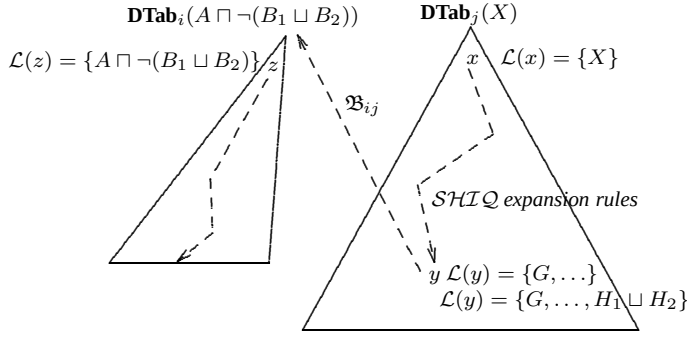


Figure 2: Subsumption propagation forced by bridge rules
 $i : A \sqsupseteq j : G$, $i : B_1 \sqsupseteq j : H_1$ and $i : B_2 \sqsupseteq j : H_2$

which caches the subsumption propagations that have been computed so far. Specifically, for every C , $IsSat_i(C)$ will be set to *True/False* whenever $\mathfrak{T} \not\models_{\epsilon} i : C \sqsubseteq \perp$ is determined.

Theorem 5.1 (Termination) *For any acyclic DTBox $\mathfrak{T}$ and for any SHIQ concept X , $\mathbf{DTab}_j(X)$ terminates.*

Theorem 5.2 (Soundness and completeness) *$j : X$ is satisfiable in $\mathfrak{T}$ if and only if $\mathbf{DTab}_j(X)$ can generate a complete and clash-free completion tree.*

Note that the construction of the distributed interpretation can be parallelized, as each local tableau procedure can run independently from the others, without checking for blocking conditions with nodes generated by the other local tableaux. The distributed tableaux proposed above has been implemented in a peer-to-peer setting, and is available for download. Some preliminary experiments have been done to see how distributed algorithm performs w.r.t. a global tableaux algorithm based on the encoding described in [2].

Example 5.1 *To clarify how $\mathbf{DTab}_j$ works, Table 1 reports the trace of $\mathbf{DTab}_{shoe}$ invocation for satisfiability test of $SHOE : BookArticle \sqcap \neg Publication$ from Example 2.1. Notice that at step (3) $\mathbf{DTab}_{shoe}$ applies the new- $\mathfrak{B}_{ij}$ -rule and invokes $\mathbf{DTab}_{src}$, using the bridge rules (1) and (3).*

$\mathbf{DTab}_{shoe}(BookArticle \sqcap \neg Publication)$			
Step	tree	labeling and <i>IsSat</i>	rule
(1)	x	$\mathcal{L}(x) = \{BookArticle \sqcap \neg Publication\}$ $IsSat = \{\}$	initial node
(2)	x	$\mathcal{L}(x) = \{BookArticle, \neg Publication\}$ $IsSat = \{\}$	$\sqcap$ -rule
(3*)	x	$\mathcal{L}(x) = \{BookArticle, \neg Publication\}$ $IsSat = \{InBook \sqcap \neg Publication = False\}$	new- $\mathfrak{B}_{ij}$ -rule
(4)	x	$\mathcal{L}(x) = \left\{ \begin{array}{l} BookArticle, \neg Publication \\ Publication \end{array} \right\}$ $IsSat = \{InBook \sqcap \neg Publication = False\}$	Unsat- $\mathfrak{B}_{ij}$ -rule
(5)	x	$\mathcal{L}(x) = \{Clash\}$ $IsSat = \{InBook \sqcap \neg Publication = False\}$	Clash-rule

$\mathbf{DTab}_{src}(InBook \sqcap \neg Publication)$			
Step	tree	labeling and <i>IsSat</i>	rule
(1)	x	$\mathcal{L}(x) = \{InBook \sqcap \neg Publication\}$ $IsSat = \{\}$	initial node
		$\vdots$	SHIQ-rules
(n)	x	$\mathcal{L}(x) = \{Clash\}$	Clash-rule

Table 1: Trace of $\mathbf{DTab}_{shoe}$ invocation

6 Conclusions

This paper has focused on the properties of a KR&R formalism that supports multiple, local knowledge bases connected by semantic mappings. It has proposed a *minor* modification to the semantics of DDL, which has the salutary effect of localizing inconsistency to the connected component of a distributed reasoner in which it occurs, and of making “knowledge import” be directional. This resolves a problem left open in [2]. Not only is the new semantics easy to explain, but so are its effect on the inferences that continue to be valid. We believe that these features are desirable for *all* distributed and modular KR&R formalisms, and we provide several such applications [9; 3; 8]. We note here that although the e-connections formalism, and its perspectival restriction [4], can simulate DDLs, their fundamental goal is to obtain hybrid logics, and in such situations there is no motivation for localizing inconsistency.

The same semantics forms the foundation of a fixed-point characterization of the formulas entailed by DDL, which can be used as the basis of a sound and complete cache-based solution to the problem of distributed reasoning. It is also the standard with respect to which one can prove the completeness (as well as, of course, soundness) of a distributed tableaux reasoner which we have implemented in a peer-to-peer setting.

References

- [1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P.F. Patel-Schneider, editors. *The Description Logic Handbook*, Cambridge University Press, 2003.
- [2] A. Borgida and L. Serafini. “Distributed description logics: Assimilating information from peer sources”, *J. of Data Semantics 1(1)*, pp.153–184, 2003.
- [3] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. “Contextualizing ontologies”, *J. of Web Semantics 1(4)*, 2004.
- [4] B. Cuenca Grau, B. Parsia, E. Sirin. “Working with Multiple Ontologies on the Semantic Web”, *Proc. ISWC’04*, LNCS vol.3298, pp.620–634, 2004.
- [5] C. Ghidini and F. Giunchiglia. “Local model semantics, or contextual reasoning = locality + compatibility”, *Artificial Intelligence*, 127(2):221–259, 2001.
- [6] C. Ghidini and L. Serafini. “Distributed first order logics”, *Proc. Frontiers of Combining Systems*, pp.121–139, 2000.
- [7] I. Horrocks, U. Sattler, and S. Tobies. A description logic with transitive and converse roles, role hierarchies and qualifying number restriction. Technical Report 99-08, Technische Universität Dresden, LTCS, 1999.
- [8] M-C. Rousset. “Small can be beautiful in the Semantic Web”, *Proc. ISWC’04*, LNCS vol.3298, pp.6–16, 2004.
- [9] H. Stuckenschmidt and M. Klein. “Integrity and change in modular ontologies”, *IJCAI’03*, pp.900-908, 2003.
- [10] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati. “Logical foundations of peer-to-peer data integration”, *PODS’04*, pp.241-251, 2004.

A Formal Investigation of Mapping Language for Terminological Knowledge

Luciano Serafini
ITC-IRST, Via Sommarive 18
38050 Povo, Trento, Italy
serafini@itc.it

Heiner Stuckenschmidt, Holger Wache
Vrije Universiteit Amsterdam
Artificial Intelligence Department
de Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
{heiner|holger}@cs.vu.nl

Abstract

The need to represent mappings between different ontologies has been recognized as a result of the fact that different ontologies may partially overlap, or even represent the same domain from different points of view. Unlike ontology languages, work on languages to represent ontology mappings has not yet reached a state where a common understanding of the basic principles exists. In this paper we propose a formal comparison of existing mapping languages by translating them into distributed first order logic. This allows us to analyze underlying assumptions and differences in the interpretation of ontology mappings.

1 Motivation

The benefits of using ontologies as explicit models of the conceptualization underlying information sources has widely been recognized. Meanwhile, a number of logical languages for representing and reasoning about ontologies have been proposed and there are even language standards now that guarantee stability and homogeneity on the language level. At the same time, the need to represent mappings between different ontologies has been recognized as a result of the fact that different ontologies may partially overlap or even represent the same domain from different points of view [Bouquet *et al.*, 2004]. As a result, a number of proposals have been made for extending ontology languages with notions of mappings between different models. Unlike for the case of ontology languages, work on languages to represent ontology mappings has not yet reached a state where a common understanding of the basic principles exists. As a consequence, existing proposals show major differences concerning almost all possible aspects of such languages. This makes it difficult to compare approaches and to make a decision about the usefulness of a particular approach in a given situation.

The purpose of this work is to provide a better understanding of the commonalities and differences of existing proposals for ontology mapping languages. We restrict our attention to logic-based approaches that have been defined as extensions of existing formalisms for representing Terminological Knowledge. In particular, we chose approaches that extend description logics (DL) with notions of mappings between

different T-boxes. The rationale for this choice is the fact that DLs are a widely agreed standard for describing terminological knowledge. In particular, DLs have gained a lot of attention as a standardized way of representing ontologies on the Semantic Web [Horrocks *et al.*, 2003].

Approach and Contributions

We encode the different mapping languages in an extended version of distributed first-order logic (DFOL), a logical framework for representing distributed knowledge systems [Ghidini and Serafini, 2000]. DFOL consists of two components: a family of first order theories and a set of axioms describing the relations between these theories. As most proposals for mapping languages are based on a subset of first-order logic for describing local models and mappings with a particular semantics for the connections between models, these mapping languages can be expressed in distributed first order logic in the following way:

- restrictions on the use of first order sentences for describing domain models
- the form of axioms that can be used for describing relations between domain models
- axioms describing the assumptions that are encoded in the specific semantics of mappings

Encoding the different mapping approaches in first-order logic in this way has several advantages for an analysis and comparison of existing work. In particular it allows us to do a formal analysis and comparison of different approaches in a uniform logical framework. In the course of the investigations, we make the following contributions to the state of the art in distributed knowledge representation and reasoning:

- we show how the DFOL formalism can be used to model relations between heterogeneous domains (Proposition 1)
- we encode existing mapping approaches in a common framework, making them more comparable
- we make hidden assumptions explicit in terms of distributed first order logic axioms
- we provide first results on the relative expressiveness of the approaches and identify shared fragments

The paper is structured as follows. In section 2 we introduce distributed first order logic as a general model for de-

scribing distributed knowledge systems. We explain the intuition of the logic and introduce its syntax and semantics. In section 3 we describe how the different mapping approaches can be encoded in distributed first order language. Here we will focus on the representation of mappings and the encoding of hidden assumptions. In section 4 we compare the different approaches based on their encoding in DFOL and discuss issues such as relative expressiveness and compatibility of the different approaches and conclude with a summary of our findings and open questions.

2 Distributed First-Order Logic

This section introduces distributed first order logic as a basis for modeling distributed knowledge bases. More details about the language including a sound and complete calculus can be found in [Ghidini and Serafini, 2005].

Let $\{L_i\}_{i \in I}$ (in the following $\{L_i\}$) be a family of first order languages with equality defined over a non-empty set I of indexes. Each language L_i is the language used by the i -th knowledge base (ontology). The signature of L_i is extended with a new set of symbols used to denote objects which are related with other objects in different ontologies. For each variable, and each index $j \in I$ with $j \neq i$ we have two new symbols $x \rightarrow^j$ and $x^{j \rightarrow}$, called *arrow variables*. Terms and formulas of L_i , also called *i -terms* and *i -formulas* and are defined in the usual way. Quantification on arrow variables is not permitted. The notation $\phi(\mathbf{x})$ is used to denote the formula ϕ and the fact that the free variables of ϕ are $\mathbf{x} = \{x_1, \dots, x_n\}$. In order to distinguish occurrences of terms and formulas in different languages we label them with their index. The expression $i : \phi$ denotes the formula ϕ of the i -th knowledge base.

The semantics of DFOL is an extension of Local Models Semantics defined in [Ghidini and Giunchiglia, 2001]. Local models are defined in terms of first order models. To capture the fact that certain predicates are completely known by the i -th sub-system we select a sub-language of L_i containing the equality predicate, denoted as L_i^c , which we call the *complete fragment* of L_i . *Complete terms* and *complete formulas* are terms and formula of L_i^c and vice versa.

Definition 1 (Set of local Models). A *set of local models* of L_i are a set of first order interpretations of L_i , on a domain $\mathbf{dom}_i$, which agree on the interpretation of L_i^c , the complete fragment of L_i .

As noted in [Franconi and Tessaris, 2004] there is a foundational difference between approaches that use epistemic states and approaches that use a classical model theoretic semantics. The two approaches differ as long as there is more than one model m . Using the notion of complete sublanguage L_c , however, we can force that the set of local models is either a singleton or the empty set by enforcing that $L^c = L$. Under this assumption the two ways of defining the semantics of submodels are equivalent. Using this assumption, we are therefore able to simulate both kinds of semantics in DFOL.

Two or more models can carry information about the same portion of the world. In this case we say that they *semantically overlap*. Overlapping is unrelated to the fact that the

same constant appears in two languages, as from the local semantics we have that the interpretation of a constant c in L_i is independent from the interpretation of the very same constant in L_j , with $i \neq j$. Overlapping is also unrelated to the intersection between the interpretation domains of two or more contexts. Namely if $\mathbf{dom}_1 \cap \mathbf{dom}_2 \neq \emptyset$ it does not mean that L_1 and L_2 overlap. Instead, DFOL explicitly represents semantic overlapping via a domain relation.

Definition 2 (Domain relation). A *domain relation* from $\mathbf{dom}_i$ to $\mathbf{dom}_j$ is a binary relation $r_{ij} \subseteq \mathbf{dom}_i \times \mathbf{dom}_j$.

A domain relation from i to j represents the capability of the j -th sub-system to represent in its domain the domain of the i -th subsystem. A pair $\langle d, d' \rangle$ being in r_{ij} means that, from the point of view of j , d in $\mathbf{dom}_i$ is the representation of d' in $\mathbf{dom}_j$. We use the functional notation $r_{ij}(d)$ to denote the set $\{d' \in \mathbf{dom}_j \mid \langle d, d' \rangle \in r_{ij}\}$. The domain relation r_{ij} formalizes j 's subjective point of view on the relation between $\mathbf{dom}_i$ and $\mathbf{dom}_j$ and not an absolute objective point of view. Or in other words $r_{ij} \neq r_{ji}$ because of the non-symmetrical nature of mappings. Therefore $\langle d, d' \rangle \in r_{ij}$ must not be read as if d and d' were the same object in a domain shared by i and j . This fact would indeed be formalized by some observer which is external (above, meta) to both i and j . Using the notion of domain relation we can define the notion of a model for a set of local models.

Definition 3 (DFOL Model). A DFOL *model* $\mathcal{M}$ is a pair $\langle \{\mathcal{M}_i\}, \{r_{ij}\} \rangle$ where, for each $i \neq j \in I$: $\mathcal{M}_i$ is a set of local models for L_i , and r_{ij} is a domain relation from $\mathbf{dom}_i$ to $\mathbf{dom}_j$.

We extend the classical notion of assignment (e.g., the one given for first order logic) to deal with arrow variables using domain relations. In particular, an assignment a , provides for each system i , an interpretation for all the variables, and for *some* (but not necessarily all) arrow variables, as the domain relations might be such that there is no consistent way to assign arrow variables. For instance if $a_i(x) = d$ and $r_{ij}(d) = \emptyset$, then a_j cannot assign anything to $x^{i \rightarrow}$.

Definition 4 (Assignment). Let $\mathcal{M} = \langle \{\mathcal{M}_i\}, \{r_{ij}\} \rangle$ be a model for $\{L_i\}$. An *assignment* a is a family $\{a_i\}$ of partial functions from the set of variables and arrow variables to $\mathbf{dom}_i$, such that:

1. $a_i(x) \in \mathbf{dom}_i$;
2. $a_i(x^{j \rightarrow}) \in r_{ji}(a_j(x))$;
3. $a_j(x) \in r_{ij}(a_i(x \rightarrow^j))$;

An assignment a is *admissible* for a formula $i : \phi$ if a_i assigns all the arrow variables occurring in ϕ . Furthermore, a is admissible for a set of formulas Γ if it is admissible for any $j : \phi \in \Gamma$. An assignment a is *strictly admissible* for a set of formulas Γ if it is admissible for Γ and assigns only the arrow variables that occur in Γ .

Using the notion of an admissible assignment given above, satisfiability in distributed first order logic is defined as follows:

Definition 5 (Satisfiability). Let $\mathcal{M} = \langle \{\mathcal{M}_i\}, \{r_{ij}\} \rangle$ be a model for $\{L_i\}$, $m \in \mathcal{M}_i$, and a an assignment. An i -formula ϕ is *satisfied* by m , w.r.t. a , in symbols $m \models_D \phi[a]$ if

- a) $\mathcal{M} \models i : P(x^{\rightarrow j}) \rightarrow j : Q(x)$ iff For all $d \in \|P\|_i$ and for all $d' \in r_{ij}(d)$, $d' \in \|Q\|_j$
- b) $\mathcal{M} \models i : P(x) \rightarrow j : Q(x^{i \rightarrow})$ iff For all $d \in \|P\|_i$ there is a $d' \in r_{ij}(d)$, s.t., $d' \in \|Q\|_j$
- c) $\mathcal{M} \models j : Q(x^{i \rightarrow}) \rightarrow i : P(x)$ iff For all $d \in \|Q\|_j$ and for all d' with $d \in r_{ij}(d')$, $d' \in \|P\|_i$
- d) $\mathcal{M} \models j : Q(x) \rightarrow i : P(x^{\rightarrow j})$ iff For all $d \in \|Q\|_j$ there is a d' with $d \in r_{ij}(d')$, s.t., $d' \in \|P\|_i$

Figure 1: Implicit Quantification of Arrow Variables in Interpretation Constraints

1. a is admissible for $i : \phi$ and
2. $m \models \phi[a_i]$, according to the definition of satisfiability for first order logic.

$\mathcal{M} \models \Gamma[a]$ if for all $i : \phi \in \Gamma$ and $m \in \mathcal{M}_i$, $m \models_D \phi[a_i]^1$.

Mappings between different knowledge bases are formalized in DFOL by a new form of constraints that involves more than one knowledge base. These formulas that will be the basis for describing different mapping approaches are called interpretation constraints and are defined as follows:

Definition 6 (Interpretation constraint). An *interpretation constraint* from $i_1, \dots, i_n$ to i with $i_k \neq i$ for $1 \leq k \leq n$ is an expression of the form

$$i_1 : \phi_1, \dots, i_n : \phi_n \rightarrow i : \phi \quad (1)$$

The interpretation constraint (1) can be considered as an axiom that restricts the set of possible DFOL models to those which satisfy it. Therefore we need to define when a DFOL model satisfies an interpretation constraint.

Definition 7 (Satisfiability of interpretation constraints). A model $\mathcal{M}$ satisfies the interpretation constraint (1), in symbols $\mathcal{M} \models i_1 : \phi_1, \dots, i_n : \phi_n \rightarrow i : \phi$ if for any assignment a strictly admissible for $\{i_1 : \phi_1, \dots, i_n : \phi_n\}$, if $\mathcal{M} \models i_k : \phi_k[a]$ for $1 \leq k \leq n$, then a can be extended to an assignment a' admissible for $i : \phi$ and such that $\mathcal{M} \models i : \phi[a']$.

Notice that, depending on whether an arrow variable $x^{\rightarrow}$ occurs on the left or on the right side of the constraint, $x^{\rightarrow}$ has a universal or an existential reading. Figure 1 summarizes the different possible readings that will reoccur later. Notationally, for any predicate P , $\|P\|_i = \bigcap_{m \in \mathcal{M}_i} m(P)$, where $m(P)$ is the interpretation of P in m .

By means of interpretation constraints on equality, we can formalize possible relations between heterogeneous domains.

$$\begin{aligned}
F_{ij} &= \{i : x^{\rightarrow j} = y^{\rightarrow j} \rightarrow j : x = y\} \\
INV_{ij} &= \left\{ \begin{array}{l} i : x = y^{j \rightarrow} \rightarrow j : x^{i \rightarrow} = y \\ j : x = y^{i \rightarrow} \rightarrow i : x^{j \rightarrow} = y \end{array} \right\} \\
OD_{ij} &= F_{ij} \cup F_{ji} \cup INV_{ij} \\
ED_{ij} &= OD_{ij} \cup \{i : x = x \rightarrow j : x^{i \rightarrow} = x^{i \rightarrow}\} \\
ID_{ij} &= ED_{ij} \cup ED_{ji} \\
RD_{ij} &= \left\{ \begin{array}{l} i : x = c \rightarrow j : x^{i \rightarrow} = c \\ j : x = c \rightarrow i : x^{j \rightarrow} = c \end{array} \middle| c \in L_i \cap L_j \right\} \\
IP_{ij} &= i : \perp \rightarrow j : \perp
\end{aligned}$$

¹Since it will be clear from the context, in the remainder we will use the classical satisfiability symbol $\models$ instead of $\models_D$ and we will write $m \models \phi[a]$ to mean that an i -formula ϕ is satisfied by m . In writing $m \models \phi[a]$ we always mean that a is admissible for $i : \phi$ (in addition to the fact that m classically satisfies ϕ under the assignment a)

Proposition 1. Let $\mathcal{M}$ be a DFOL model and $i \neq j \in I$.

1. $\mathcal{M} \models F_{ij}$ iff r_{ij} is a partial function.
2. $\mathcal{M} \models INV_{ij}$ iff r_{ij} is the inverse of r_{ji} .
3. $\mathcal{M} \models OD_{ij}$ if $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between a subset of dom_i and a subset of dom_j . I.e., dom_i and dom_j (isomorphically) overlap.
4. $\mathcal{M} \models ED_{ij}$ iff $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between dom_i and a subset of dom_j . I.e., dom_i is (isomorphically) embedded in dom_j .
5. $\mathcal{M} \models ID_{ij}$ iff $r_{ij}(= r_{ji}^{-1})$ is an isomorphism between dom_i and dom_j . I.e., dom_i is isomorphic to dom_j .
6. $\mathcal{M} \models RD$, if for every constant c of L_i and L_j , if c is interpreted in d for all $m \in \mathcal{M}_i$ then c is interpreted in $r_{ij}(d)$ for all models of $m \in \mathcal{M}_j$, and vice-versa. I.e., the constant c is rigidly interpreted by i and j in two corresponding objects.
7. Finally $\mathcal{M} \models IP_{ij}$ iff $\mathcal{M}_i = \emptyset$ implies that $\mathcal{M}_j = \emptyset$. I.e., inconsistency propagates from i to j .

3 Modeling Mapping Languages in DFOL

Formalisms for mapping languages are based on four main parameters: local languages and local semantics used to specify the local knowledge, and mapping languages and semantics for mappings, used to specify the semantic relations between the local knowledge. In this section we focus on the second pairs and as far as local languages and local semantics it is enough to notice that

Local languages In all approaches local knowledge is expressed by a suitable fragment of first order languages.

Local semantics with the notable exception of [Franconi and Tessaris, 2004], where authors propose an *epistemic approach* to information integration, all the other formalisms for ontology mapping assume that each local knowledge is interpreted in a (partial) state of the world and not into an epistemic state. This formally corresponds to the fact that each local knowledge base is associated with *at most one* FOL interpretation.

The first assumption is naturally captured in DFOL, by simply considering L_i to be an adequately restricted FOL language. Concerning the local semantics, in DFOL models each L_i is associated with a *set of interpretations*. To simulate the single local model assumption, in DFOL it is enough to declare each L_i to be a *complete* language. This implies that all the $m \in M_i$ have to agree on the interpretation of L_i -symbols.

Notationally, $\phi, \psi, \dots$ will be used to denote both DL expressions and FOL open formulas. If ϕ is a DL concept, $\phi(x)$ (or $\phi(x_1, \dots, x_n)$) will denote the corresponding translation of ϕ in FOL as described in [Borgida, 1996]. If ϕ is a role R then $\phi(x, y)$ denotes its translation $P(x, y)$, and if ϕ is a

constant c , then $\phi(x)$ denote its translation $x = c$. Finally we use $\mathbf{x}$ to denote a set $x_1, \dots, x_n$ of variables.

3.1 Distributed Description Logics/C-OWL

The approach presented in [Borgida and Serafini, 2003] extends DL with a local model semantics similar to the one introduced above and so-called bridge rules to define semantic relations between different T-Boxes. A distributed interpretation for DDL on a family of DL languages $\{L_i\}$, is a family $\{\mathcal{I}_i\}$ of interpretations, one for each L_i plus a family $\{r_{ij}\}_{i \neq j \in I}$ of domain relations. While the original proposal only considered subsumption between concept expressions, the model was extended to a set of five semantic relations discussed below. The semantics of the five semantic relations defined in C-OWL is the following:

Definition 8 ([Bouquet *et al.*, 2004]). Let ϕ and ψ be either concepts, or individuals, or roles of the descriptive languages L_i and L_j respectively².

1. $\mathcal{I} \models i:\phi \xrightarrow{\sqsubseteq} j:\psi$ if $r_{ij}(\phi^{\mathcal{I}_i}) \subseteq \psi^{\mathcal{I}_j}$;
2. $\mathcal{I} \models i:\phi \xrightarrow{\sqsupseteq} j:\psi$ if $r_{ij}(\phi^{\mathcal{I}_i}) \supseteq \psi^{\mathcal{I}_j}$;
3. $\mathcal{I} \models i:\phi \xrightarrow{=} j:\psi$ if $r_{ij}(\phi^{\mathcal{I}_i}) = \psi^{\mathcal{I}_j}$;
4. $\mathcal{I} \models i:\phi \xrightarrow{\perp} j:\psi$ if $r_{ij}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} = \emptyset$;
5. $\mathcal{I} \models i:\phi \xrightarrow{*} j:\psi$ if $r_{ij}(\phi^{\mathcal{I}_i}) \cap \psi^{\mathcal{I}_j} \neq \emptyset$;

An interpretation for a context space is a model for it if all the bridge rules are satisfied.

From the above satisfiability condition one can see that the mapping $i:\phi \xrightarrow{=} j:\psi$ is equivalent to the conjunction of the mappings $i:\phi \xrightarrow{\sqsubseteq} j:\psi$ and $i:\phi \xrightarrow{\sqsupseteq} j:\psi$. The mapping $i:\phi \xrightarrow{\perp} j:\psi$ is equivalent to $i:\phi \xrightarrow{\sqsubseteq} j:\neg\psi$. And finally the mapping $i:\phi \xrightarrow{*} j:\psi$ is the negation of the mapping $i:\phi \xrightarrow{\sqsubseteq} j:\psi$. For the translation we will therefore consider only the primitive mappings. As the underlying notion of a model is the same as for DFOL, we can directly try to translate bridge rules into interpretation constraints. In particular, there are no additional assumptions about the nature of the domains that have to be modeled. The translation is the following:

C-OWL	DFOL
$i:\phi \xrightarrow{\sqsubseteq} j:\psi$	$i:\phi(x \rightarrow^j) \rightarrow j:\psi(x)$
$i:\phi \xrightarrow{\sqsupseteq} j:\psi$	$j:\psi(x) \rightarrow i:\phi(x \rightarrow^j)$
$i:\phi \xrightarrow{\perp} j:\psi$	No translation

We see that a bridge rule basically corresponds to the interpretation a) and d) in Figure 1. The different semantic relations correspond to the usual readings of implications. Finally negative information about mappings (i.e., $i:\phi \xrightarrow{\perp} j:\psi$ is not representable by means of DFOL interpretation constraints.

²In this definition, to be more homogeneous, we consider the interpretations of individuals to be sets containing a single object rather than the object itself.

3.2 Ontology Integration Framework (OIS)

Calvanese and colleagues in [Calvanese *et al.*, 2002b] propose a framework for mappings between ontologies that generalizes existing work on view-based schema integration [Ullman, 1997] and subsumes other approaches on connecting DL models with rules. In particular, they distinguish global centric, local centric and the combined approach. These approaches differ in the types of expressions connected by mappings. With respect to the semantics of mappings, they do not differ and we therefore treat them as one.

OIS assumes the existence of a global model g into which all local models s are mapped. On the semantic level, the domains of the local models are assumed to be embedded in a global domain. Further, in OIS constants are assumed to rigidly designate the same objects across domain. Finally, global inconsistency is assumed, in the sense that the inconsistency of a local knowledge base makes the whole system inconsistent. As shown in Proposition 1, we can capture these assumptions by the set of interpretation constraints ED_{sg} , RD_{sg} , and IP_{sg} , where s is the index of any source ontology and g the index of the global ontology.

According to these assumptions mappings are described in terms of correspondences between a local and the global model. The interpretation of these correspondences are defined as follows:

Definition 9 ([Calvanese *et al.*, 2002b]). Correspondences between source ontologies and global ontology are of the following four forms

1. $\mathcal{I}$ satisfies $\langle \phi, \psi, \text{sound} \rangle$ w.r.t. the local interpretation $\mathcal{D}$, if all the tuples satisfying ψ in $\mathcal{D}$ satisfy ϕ in $\mathcal{I}$
2. $\langle \phi, \psi, \text{complete} \rangle$ w.r.t. the local interpretation $\mathcal{D}$, if no tuple other than those satisfying ψ in $\mathcal{D}$ satisfies ϕ in $\mathcal{I}$,
3. $\langle \phi, \psi, \text{exact} \rangle$ w.r.t. the local interpretation $\mathcal{D}$, if the set of tuples that satisfies ψ in $\mathcal{D}$ is exactly the set of tuples satisfying ϕ in $\mathcal{I}$.

From the above semantic conditions, $\langle \phi, \psi, \text{exact} \rangle$ is equivalent to the conjunction of $\langle \phi, \psi, \text{sound} \rangle$ and $\langle \phi, \psi, \text{complete} \rangle$. It's therefore enough to provide the translation of the first two correspondences. The definitions 1 and 2 above can directly be expressed into interpretation constraints (compare Figure 1) resulting in the following translation:

GLAV Correspondence	DFOL
$\langle \phi, \psi, \text{sound} \rangle$	$s:\psi(\mathbf{x}) \rightarrow g:\phi(\mathbf{x} \rightarrow^s)$
$\langle \phi, \psi, \text{complete} \rangle$	$g:\phi(\mathbf{x}) \rightarrow s:\psi(\mathbf{x} \rightarrow^g)$

The translation shows that there is a fundamental difference in the way mappings are interpreted in C-OWL and in OIS. While C-OWL mappings correspond to a universally quantified reading (Figure 1 a), OIS mappings have an existentially quantified readings (Figure 1 b/d). We will come back to this difference later.

3.3 DL for Information Integration (DLII)

A slightly different approach to the integration of different DL models is described in [Calvanese *et al.*, 2002a]. This approach assumes a partial overlap between the domains of the models M_i and M_j , rather than a complete embedding of

them in a global domain. This is captured by the interpretation constraint OD_{ij} . The other assumptions (rigid designators and global inconsistency) are the same as for OIS.

An interpretation $\mathcal{I}$ associates to each M_i a domain Δ_i . These different models are connected by interschema assertions. Satisfiability of interschema assertions is defined as follows³

Definition 10 (Satisfiability of interschema assertions). If $\mathcal{I}$ is an interpretation for M_i and M_j we say that $\mathcal{I}$ satisfies the interschema assertion

$$\begin{aligned} \phi \sqsubseteq_{\text{ext}} \psi, & \text{ if } \phi^{\mathcal{I}} \subseteq \psi^{\mathcal{I}} & \phi \not\sqsubseteq_{\text{ext}} \psi, & \text{ if } \phi^{\mathcal{I}} \not\subseteq \psi^{\mathcal{I}} \\ \phi \equiv_{\text{ext}} \psi, & \text{ if } \phi^{\mathcal{I}} = \psi^{\mathcal{I}} & \phi \not\equiv_{\text{ext}} \psi, & \text{ if } \phi^{\mathcal{I}} \neq \psi^{\mathcal{I}} \\ \phi \sqsubseteq_{\text{int}} \psi, & \text{ if } \phi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} \subseteq \psi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} & & \\ \phi \equiv_{\text{int}} \psi, & \text{ if } \phi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} = \psi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} & & \\ \phi \not\sqsubseteq_{\text{int}} \psi, & \text{ if } \phi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} \not\subseteq \psi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} & & \\ \phi \not\equiv_{\text{int}} \psi, & \text{ if } \phi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} \neq \psi^{\mathcal{I}} \cap \tau_{nij}^{\mathcal{I}} & & \end{aligned}$$

As before $\equiv_{\text{ext}}$ and $\equiv_{\text{int}}$ are definable as conjunctions of $\sqsubseteq_{\text{ext}}$ and $\sqsubseteq_{\text{int}}$, so we can ignore them for the DFOL translation. Furthermore, a distinction is made between extensional and intentional interpretation of interschema assertions, which leads to different translations into DFOL.

interschema assertions	DFOL
$\phi \sqsubseteq_{\text{ext}} \psi$	$i: \phi(\mathbf{x}) \rightarrow j: \psi(\mathbf{x}^{i \rightarrow})$
$\phi \not\sqsubseteq_{\text{ext}} \psi, \phi \not\equiv_{\text{ext}} \psi$	No translation
$\phi \sqsubseteq_{\text{int}} \psi$	$i: \phi(\mathbf{x}^{i \rightarrow j}) \rightarrow j: \psi(\mathbf{x})$
$\phi \not\sqsubseteq_{\text{int}} \psi, \phi \not\equiv_{\text{int}} \psi$	No translation

While the extensional interpretation corresponds to the semantics of mappings in OIS, the intentional interpretation corresponds to the semantics of mappings in C-OWL. Thus using the distinction made in this approach we get an explanation of different conceptualizations underlying the semantics of C-OWL and OIS that use an extensional and an intentional interpretation, respectively.

3.4 ϵ -connections

A different approach for defining relations between DL knowledge bases has emerged from the investigation of so-called ϵ -connections between abstract description systems [Kutz *et al.*, 2004]. Originally intended to extend the decidability of DL models by partitioning them into a set of models that use a weaker logic, the approach has recently been proposed as a framework for defining mappings between ontologies [Grau *et al.*, 2004].

In the ϵ -connections framework, for every pair of ontologies ij there is a set ϵ_{ij} of *links*, which represents binary relations between the domain of the i -th ontology and the domain of the j -th ontology. Links from i to j can be used to define i -concepts, in a way that is analogous to how roles are used to define concepts. In the following table we report the syntax and the semantics of i -concept definitions based on links. (E denotes a link from i to j and C denotes a concept in j . The only assumption about the relation between domains is global inconsistency, see above).

³To simplify the definition we introduce the notation $\tau_{nij}^{\mathcal{I}} = \tau_{ni}^{\mathcal{I}} \cap \tau_{nj}^{\mathcal{I}}$ for any $n \geq 1$. Notice that $\tau_{nij}^{\mathcal{I}} = \Delta_i^n \cap \Delta_j^n$.

In DFOL we have only one single relation from i to j , while in ϵ -connection there are many possible relations. However, we can use a similar trick as used in [Borgida and Serafini, 2003] to map relations to interschema relations: each of the relations in ϵ_{ij} acts as a r_{ij} . To represent ϵ -connections it is therefore enough to label each arrow variable with the proper link name. The arrow variable $x \xrightarrow{\text{Own}_{ij}} j$ is read as the arrow variable $x \xrightarrow{i} j$ where r_{ij} is intended to be the interpretation of Own_{ij} . With this syntactic extension of DFOL, concept definitions based on links (denoted as E) can be codified in DFOL as follows:

ϵ -connections	DFOL
$\phi \sqsubseteq \exists E.\psi$	$i: \phi(x) \rightarrow j: \psi(x \xrightarrow{i} j)$
$\phi \sqsubseteq \forall E.\psi$	$i: \phi(x \xrightarrow{i} j) \rightarrow j: \psi(x)$
$\phi \sqsubseteq \geq n E.\psi$	$i: \bigwedge_{k=1}^n \phi(x_1) \rightarrow$ $j: \bigwedge_{k \neq h=1}^n \psi(x_k \xrightarrow{i} j) \wedge x_k \neq x_h$
$\phi \sqsubseteq \leq n E.\psi$	$i: \phi(x) \wedge \bigwedge_{k=1}^{n+1} x = x_k \xrightarrow{i} j \rightarrow$ $j: \bigvee_{k=1}^{n+1} (\psi(x_k) \supset \bigvee_{h \neq k} x_h = x_k)$

We see that like OIS, links in the ϵ -connections framework have an extensional interpretation. The fact, that the framework distinguishes between different types of domain relations, however, makes it different from all other approaches.

Another difference to the previous approaches is that new links can be defined on the basis of existing links, similar to complex roles in DL. Syntax and semantics for link constructors is defined in the usual way: $(E^-)^I = (E^I)^{-1}$ (Inverse), $(E \sqcap F)^I = E^I \cap F^I$ (Conjunction), $(E \sqcup F)^I = E^I \cup F^I$ (Disjunction), and $(\neg E)^I = (\Delta_i \times \Delta_j) \setminus E^I$ (Complement). Notice that, by means of inverse links we can define mappings of the b and d type. E.g., the ϵ -connection statement $\phi \sqsubseteq \exists E^- \psi$ corresponds to the DFOL bridge rule $\phi(x) : i \rightarrow \psi x \xrightarrow{i} j$ which is of type b). Similarly the ϵ -connection $\phi \sqsubseteq \forall E^- \psi$ corresponds to a mapping of type d).

As the distinctions between different types of links is only made on the model theoretic level, it is not possible to model Boolean combinations of links. Inverse links, however, can be represented by the following axiom:

$$\begin{aligned} i: y = x \xrightarrow{i} j & \rightarrow j: y \xrightarrow{j} i = x \\ j: y \xrightarrow{j} i = x & \rightarrow i: y = x \xrightarrow{i} j \end{aligned}$$

Finally the inclusion axioms between links, i.e., axioms of the form $E \sqsubseteq F$ where E and F are homogeneous links, i.e., links of the same ϵ_{ij} , can be translated in DFOL as follows:

$$i: x = y \xrightarrow{i} j \rightarrow j: x \xrightarrow{j} i = y$$

We can say that the ϵ -connections framework significantly differs from the other approaches in terms of the possibilities to define and combine mappings of different types.

4 Discussion and Conclusions

The encoding of different mapping approaches in a common framework has two immediate advantages. The first one is the

ability to reason across the different frameworks. This can be done on the basis of the DFOL translation of the different approaches using the sound and complete calculus for DFOL [Ghidini and Serafini, 2000]. As there are not always complete translations, this approach does not cover all aspects of the different approaches, but as shown above, we can capture most aspects. There are only two aspects which cannot be represented in DFOL, namely “non mappings” ($i:\phi \xrightarrow{*} j:\psi$ in C-OWL, $\phi \not\sqsubseteq_{int} \psi$ etc. in DLII) and “complex mappings” such as complex links in ϵ -connection. The second benefit is the possibility to compare the expressiveness of the approaches. We have several dimensions along which the framework can differ:

Arity of mapped items⁴ C-OWL allows only to align constants, concepts and roles (2-arity relations), ϵ -connections allow to align only 1-arity items, i.e., concepts, while DLII and OIS allow to integrate n -arity items.

Positive/negative mappings Most approaches state positive facts about mapping, e.g that two elements are equivalent. The DLII and C-OWL frameworks also allow to state that two elements do not map ($\phi \not\equiv \psi$).

Domain relations The approaches make different assumptions about the nature of the domain. While C-OWL and ϵ -connections do not assume any relation between the domains, DLII assumes overlapping domains and OIS assumes local domains that are embedded in a global domain.

Multiple mappings Only the ϵ -connection approach supports the definition of different types of mappings between ontologies that partition the inter-domain relations.

Local inconsistency Some approaches provide a consistent semantics also in the case in which some of the ontologies or mappings are inconsistent.

We summarize the comparison in the following table.

	Int. constr. (cf. fig. 1)				Mapping type			Domain relation	Arity	Local $\perp$
	a)	b)	c)	d)	Pos.	Neg.	Mult.			
C-OWL	x			x	x	x		Het.	2	x
OIS		x		x	x			Incl.	n	
DLII	x	x			x	x		Emb.	n	
ϵ -Conn.	x	x	x	x	x	x	x	Het.	1	

We conclude that existing approaches make choices along a number of dimensions. These choices are obviously influenced by the intended use. Approaches intended for database integration for example will support the mapping of n -ary items that correspond to tuples in the relational model. Despite this fact, almost no work has been done on charting the landscape of choices to be made when designing a mapping approach, and for adapting the approach to the requirements of an application. The work reported in this paper provides the basis for this kind of work by identifying the possible choices on a formal level. An important topic of future work is to identify possible combinations of features for mapping languages on a formal level in order to get a more complete picture of the design space of mapping languages.

⁴Due to limited space we did not discuss the encoding of mapped items in this paper

References

- [Borgida and Serafini, 2003] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.
- [Borgida, 1996] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82:353–367, 1996. Research Note.
- [Bouquet et al., 2004] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. *Journal on Web Semantics*, 1(4):xx–xx, 2004.
- [Calvanese et al., 2002a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2002.
- [Calvanese et al., 2002b] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A framework for ontology integration. In Isabel Cruz, Stefan Decker, Jerome Euzenat, and Deborah McGuinness, editors, *The Emerging Semantic Web*, pages 201–214. IOS Press, 2002.
- [Franconi and Tessaris, 2004] E. Franconi and S. Tessaris. Rules and queries with ontologies: a unified logical framework. In *Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR’04)*, 2004.
- [Ghidini and Giunchiglia, 2001] C. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.
- [Ghidini and Serafini, 2000] C. Ghidini and L. Serafini. Distributed first order logics. In D.M. Gabbay and M. De Rijke, editors, *Frontiers of Combining Systems 2*, pages 121–139. Research Studies Press Ltd., 2000.
- [Ghidini and Serafini, 2005] Chiara Ghidini and Luciano Serafini. Distributed first order logic - revised semantics. Technical report, ITC-irst, January 2005.
- [Grau et al., 2004] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *Proceedings of the Third International Semantic Web Conference (ISWC2004)*, volume 3298 of *Lecture Notes in Computer Science*, 2004.
- [Horrocks et al., 2003] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
- [Kutz et al., 2004] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyashev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.
- [Ullman, 1997] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of the 6th International Conference on Database Theory*, volume 1186 of *Lecture Notes In Computer Science*, pages 19 – 40, 1997.

Goal Change

Steven Shapiro

Computer Science Inst.

University of Leipzig

Leipzig 04109

Germany

shapiro@informatik.uni-leipzig.de

Yves Lespérance

Dept. of Computer Science

York University

Toronto, ON

M3J 1P3 Canada

lesperan@cs.yorku.ca

Hector J. Levesque

Dept. of Computer Science

University of Toronto

Toronto, ON

M5S 3G4 Canada

hector@cs.toronto.edu

Abstract

Although there has been much discussion of belief change (e.g., [Gärdenfors, 1988; Spohn, 1988]), goal change has not received much attention. In this paper, we propose a method for goal change in the framework of Reiter's [2001] theory of action in the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998], and investigate its properties. We extend the framework developed by Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled, but goal contraction was not.

1 Introduction

One of the commonsense abilities people possess is the ability to predict the behaviour of others. We attribute goals and beliefs to others and assume they will act on their beliefs to achieve their goals. One strategy for endowing machines with a similar ability is to explicitly model the beliefs and goals of agents and assume they are acting to achieve their goals in accordance with their beliefs. In dynamic environments, the beliefs and goals of agents are likely to change. Although there has been much discussion of belief change (e.g., [Gärdenfors, 1988; Spohn, 1988]), goal change has not received much attention. In this paper, we propose a method for specifying and reasoning about goal change in the framework of Reiter's [2001] theory of action in the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998]. We thus inherit Reiter's solution to the frame problem, and iterated goal change is achieved seamlessly using sequences of actions. Our approach to goal contraction is simpler than most approaches to belief revision and relies on using the history of requests rather than a specification of entrenchment. This work extends the framework developed by Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled but goal contraction was not.

We model agents that adopt goals if they are requested to do so and they do not already have a conflicting goal. They maintain their goals unless they are explicitly requested to drop them by the agent who requested the goal in the first

place. We first define goals using an accessibility relation, W , which picks out the situations that the agent wants to be the case. We then give a successor state axiom for W that is affected by two communicative actions: REQUEST actions, which cause agents to adopt goals, and CANCELREQUEST actions, which cause agents to drop goals that had been adopted following prior requests. We then consider some properties of our axiomatization: expansion, contraction, and persistence. Finally, we identify the restrictions on the accessibility relations that give us positive and negative introspection of goals, and show that these restrictions persist, if they are asserted of the initial situations.

The remainder of the paper is organized as follows. In Sec. 2, we give a brief review of the situation calculus, Reiter's theory of action, and Shapiro *et al.*'s [1998] framework for representing knowledge expansion in the situation calculus. In Sec. 3, we introduce our framework for goal change, and in Sec. 4, we investigate properties of this framework. In Sec. 5, we present an example, and in Sec. 6, we conclude, discuss related work, and suggest avenues for future work.

2 Representation of Action and Knowledge

The basis of our framework for goal change is an action theory [Reiter, 2001] based on the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998]. The situation calculus is a predicate calculus language for representing dynamically changing domains. A situation represents a possible state of the domain. There is a set of initial situations corresponding to the ways the agents believe the domain might be initially. The actual initial state of the domain is represented by the distinguished initial situation constant, S_0 . The term $do(a, s)$ denotes the unique situation that results from the agent doing action a in situation s . Thus, the situations can be structured into a set of trees, where the root of each tree is an initial situation and the arcs are actions. The sequence of actions that produces a situation is called the *history* of the situation. $s \preceq s'$ ($s \prec s'$, resp.) means there is a (nonempty, resp.) path from situation s to situation s' . The initial situations are defined as those having an empty history:

$$Init(s) \stackrel{\text{def}}{=} \neg \exists a, s'. s = do(a, s').$$

Predicates and functions whose value may change from situation to situation (and whose last argument is a situation) are called *fluents*. For instance, we might use the fluent

$\text{INROOM}(\text{Agt}, R_1, S)$ to represent the fact that agent Agt is in room R_1 in situation S . The effects of actions on fluents are defined using successor state axioms [Reiter, 2001], which provide a succinct representation for both effect axioms and frame axioms [McCarthy and Hayes, 1969].

We will be quantifying over formulae, so we assume that we have an encoding of formulae as first-order terms (we can adapt De Giacomo *et al.*'s [2000] axioms for this purpose, but we omit the details here). Therefore, we will freely quantify over formulae here. To simplify notation, we ignore the details of the encoding and use formulae directly instead of the terms that represent them. We use ψ to denote a formula that can contain the situation constants *Now* and *Then*, and ϕ to denote a formula that can contain *Now* but not *Then*. We use $\phi[s]$ ($\psi[s, s']$, resp.) to denote the formula that results from substituting s for *Now* in ϕ (s for *Now* and s' for *Then*, resp.).

To axiomatize a dynamic domain in the situation calculus, we use Reiter's [2001] action theory, which consists of (1) successor state axioms for each fluent (including K and W introduced below); (2) initial state axioms, which describe the initial state of the domain and the initial mental states of the agents; (3) precondition axioms, which specify the conditions under which the distinguished predicate $\text{Poss}(A, S)$ holds, i.e., when it is physically possible to execute an action A in situation S ; (4) unique names axioms for the actions, and domain-independent foundational axioms (we adopt the ones given by Levesque *et al.* [1998] which accommodate multiple initial situations, but we do not describe them further here); and (5) the axioms to encode formulae as terms.

Moore [1985] defined a possible-worlds semantics for a logic of knowledge in the situation calculus by treating situations as possible worlds. Scherl and Levesque [2003] adapted this to Reiter's theory of action. Scherl and Levesque gave a successor state axiom for K that states how actions, including sensing actions, affect knowledge. Shapiro *et al.* [1998] adapted this axiom to handle multiple agents and INFORM actions, which we adopt here. $K(\text{agt}, s', s)$ will be used to denote that in situation s , agt thinks that situation s' might be the actual situation. Note that the order of the situation arguments is reversed from the convention in modal logic for accessibility relations. $\text{INFORM}(\text{infr}, \text{agt}, \phi)$ is the action of the agent *infr* informing another agent, agt , that ϕ holds. Here is Shapiro *et al.*'s [1998] successor state axiom for K .¹

Axiom 2.1

$$\begin{aligned} K(\text{agt}, s'', \text{do}(a, s)) \equiv \\ \exists s'. K(\text{agt}, s', s) \wedge s'' = \text{do}(a, s') \wedge \text{Poss}(a, s') \wedge \\ \forall \text{infr}, \phi. a = \text{INFORM}(\text{infr}, \text{agt}, \phi) \supset \phi[s'] \end{aligned}$$

First note that for any action other than an INFORM action directed towards agt , the specification ensures that the

¹Since we are using terms for formulae in our underlying representation, $\text{INFORM}(\text{infr}, \text{agt}, \phi) = \text{INFORM}(\text{infr}, \text{agt}, \phi')$, only if ϕ and ϕ' are the same term. It would not be difficult to modify our representation so that the equality holds only if ϕ and ϕ' are the representations of equivalent formulae. We adopt the convention that unbound variables are universally quantified in the widest scope.

only change in knowledge that occurs in moving from s to $\text{do}(a, s)$ is that it is known (by all agents) that the action a has been successfully executed. This is true because the K -alternative² situations to $\text{do}(a, s)$ are the situations that result from doing a in a K -alternative situation to s where a is possible to execute (denoted by $\text{Poss}(a, s')$). For the action $\text{INFORM}(\text{infr}, \text{agt}, \phi)$, the idea is that in moving from s to $\text{do}(\text{INFORM}(\text{infr}, \text{agt}, \phi), s)$, agt not only knows that the action has been executed (as above), but it also knows that ϕ holds. In this case, the K -alternative situations to $\text{do}(a, s)$ for agt are the situations that result from performing a in a K -alternative situation to s where a is possible to execute, except the ones where ϕ is false.³ As usual, we say that an agent knows a formula ϕ in s , if ϕ holds in all situations K -accessible from s , i.e., $\mathbf{Know}(\text{agt}, \phi, s) \stackrel{\text{def}}{=} \forall s'. K(\text{agt}, s', s) \supset \phi[s']$.

Following Shapiro *et al.* [1998], we say that an agent can only execute an inform action if it knows that the content of the action is true.

Axiom 2.2

$$\text{Poss}(\text{INFORM}(\text{infr}, \text{agt}, \phi), s) \equiv \mathbf{Know}(\text{infr}, \phi, s).$$

Since we are dealing with knowledge here, not belief, we assert that K is (initially) reflexive.⁴

Axiom 2.3 $\text{Init}(s) \supset K(\text{agt}, s, s)$.

Following Scherl and Levesque, we also assert that initial situations can only be K -related to other initial situations. We use an initial state axiom for this purpose:

Axiom 2.4 $K(\text{agt}, s', s) \wedge \text{Init}(s) \supset \text{Init}(s')$.

This axiom and the successor state axiom for K imply that only situations with the same history can be K -related. Introspection will be discussed in Sec. 4.

3 Goal Change

Just as an agent's beliefs can change, its goals should also be able to change. For example, if an agent's owner requests it to do something, the agent's goals should change to reflect the request. We extend the framework developed by Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled but not goal contraction.

Following Cohen and Levesque [1990], we model goals using an accessibility relation over possible worlds (situations,

²We say that S' is a K -alternative situation to S for Agt or S' is K -related to S for Agt , if $K(\text{Agt}, S', S)$ holds. We may drop Agt when it is understood from the context.

³Note that here all agents are aware of all actions that occur, including inform actions, so we have a broadcast model of communication. In Shapiro and Lespérance [2001], encrypted speech acts were introduced, ensuring that only the intended recipient of a message could know its contents.

⁴We do not handle belief revision here, but a treatment of belief revision in our framework was discussed in [Shapiro *et al.*, 2000].

in our case). The accessible worlds are the ones that are compatible with what the agent *wants* to be the case. Cohen and Levesque's accessibility relation for goals, G , was a subset of their accessibility relation for beliefs. This constraint precludes an agent wanting something that it believes is impossible (unless its goals are inconsistent). In our case, we define G in terms of a more primitive accessibility relation, which we call W and is intended to model what the agent wants independently of what it knows. We can then define Cohen and Levesque's goal accessibility relation, G , to be the intersection⁵ of W and K .

An agent's goals are future oriented. For example, an agent might want some property to hold eventually, i.e., the agent's goal is of the form **Eventually**(ψ), for some formula ψ . We evaluate formulae such as these with respect to a path of situations rather than a single situation, and we call such formulae *path formulae*. Cohen and Levesque used infinite time-lines to evaluate such formulae, but for simplicity, we evaluate path formulae with respect to finite paths of situations which we represent by pairs of situations, (*Now*, *Then*), such that $\text{Now} \preceq \text{Then}$. *Now* corresponds to the "current time" on the path of situations defined by the sequence of situations in the history of *Then*. Path formulae may contain two situation constants, *Now* and *Then*. For example, $\exists r. \text{INROOM}(\text{JOHN}, r, \text{Now}) \wedge \neg \text{INROOM}(\text{JOHN}, r, \text{Then})$ could be used to denote the goal that John eventually leaves the room he is in currently.

Our W relation, like our K relation, is a relation on situations. While K only relates situations with the same histories, W can relate situations with different histories. Intuitively, $W(\text{Agt}, S', S)$ holds if in situation S , Agt considers that in S' everything that it wants to be true is actually true. For example, if the agent wants to become a millionaire in S , then in all situations W -related to S , the agent is a millionaire, but these situations can be arbitrarily far in the future.

Recall that the situations in the W accessibility relation are the ones that the agent wants to actualize independently of what it knows. Following Cohen and Levesque, we want the goals of the agent to be compatible with what it knows. The situations that the agent wants to actualize should be on a path from a situation that the agent considers possible. Therefore, the situations that will be used to determine the goals of an agent will be the W -accessible situations that are also compatible with what the agent knows, in the sense that there is K -accessible situation in their history. We will say that $S' K_{\text{Agt}, S}$ -intersects S'' if $K(\text{Agt}, S'', S)$ and $S'' \preceq S'$. We will suppress Agt or S if they are understood from the context. We define the goals of agt in s to be those formulae that are true in all the situations s' that are W -accessible from s and that K -intersect some situation, s'' :

$$\text{Goal}(\text{agt}, \psi, s) \stackrel{\text{def}}{=} \forall s', s''. W(\text{agt}, s', s) \wedge K(\text{agt}, s'', s) \wedge s'' \preceq s' \supset \psi[s'', s'].$$

Note that s'' corresponds to the "current situation" (or the current time) in the path defined by s' .

⁵As we will see below, the operation used on W and K to obtain G is not quite intersection, but a related operation.

As noted by Konolige and Pollack [1993], one has to be careful when using this definition of a goal. Suppose an agent has a conjunctive goal, $\psi \wedge \psi'$. According to this definition of **Goal**, both ψ and ψ' are also goals. Therefore, we could imagine a rational agent working to achieve one of the conjuncts as a subgoal. But it is easy to think of circumstances where achieving only one component of a conjunctive goal is undesirable. For example, I may have as a goal to be holding the bomb *and* that the bomb be defused. However, I want these to be true simultaneously and I do not have the goal to be holding the bomb if it is not also defused. As Konolige and Pollack did for their intention modality (I), we consider formulae that are an agent's "only goals", i.e., formulae that are true in all and *only* the W -paths:

$$\text{OGoal}(\text{agt}, \psi, s) \stackrel{\text{def}}{=} \forall s', s''. K(\text{agt}, s'', s) \wedge s'' \preceq s' \supset (W(\text{agt}, s', s) \equiv \psi[s'', s']).$$

We now give the successor state axiom for W . $W^+(\text{agt}, a, s', s)$ ($W^-(\text{agt}, a, s', s)$, resp.), which is defined below, denotes the conditions under which s' is added to (dropped from, resp.) W due to action a :

Axiom 3.1

$$W(\text{agt}, s', \text{do}(a, s)) \equiv (W^+(\text{agt}, a, s', s) \vee (W(\text{agt}, s', s) \wedge \neg W^-(\text{agt}, a, s', s))).$$

An agent's goals are expanded when it is requested to do something by another agent. After the $\text{REQUEST}(\text{requester}, \text{agt}, \psi)$ action occurs, agt should adopt the goal that ψ , unless it currently has a conflicting goal (i.e., we assume agents are maximally cooperative). Therefore, the $\text{REQUEST}(\text{requester}, \text{agt}, \psi)$ action should cause agt to drop any paths in W where ψ does not hold. This action is taken into account in the definition of W^- :

$$W^-(\text{agt}, a, s', s) \stackrel{\text{def}}{=} \exists \text{requester}, \psi, s''. a = \text{REQUEST}(\text{requester}, \text{agt}, \psi) \wedge \neg \text{Goal}(\text{agt}, \neg \psi, s) \wedge K(\text{agt}, s'', s) \wedge s'' \preceq s' \wedge \neg \psi[\text{do}(a, s''), s'].$$

According to this definition, s' will be dropped from W , if for some requester and ψ , a is the $\text{REQUEST}(\text{requester}, \text{agt}, \psi)$ action and agt does not have a conflicting goal, i.e., the goal that $\neg \psi$ in s^6 , and s' K -intersects some s'' such that ψ does not hold on the path $(\text{do}(a, s''), s')$. The reason that we check whether $\neg \psi$ holds at $(\text{do}(a, s''), s')$ rather than at (s'', s') is to handle goals that are relative to the current time. If, for example, ψ states that the very next action should be to get some coffee, then we need to check whether the next action after the request is getting the coffee. If we checked $\neg \psi$ at (s'', s') , then the next action would be the REQUEST action.

If the agent gets a request for ψ and it already has the goal that $\neg \psi$, then it does not adopt the goal that ψ , otherwise its

⁶Note that according to the definition of **Goal**, if the agent has a goal that implies $\neg \psi$, then it also has the goal that $\neg \psi$, i.e., if $\text{Goal}(\text{Agt}, \psi', S)$ holds and $\forall s, s'. \psi'[s, s'] \supset \neg \psi[s, s']$ holds, then $\text{Goal}(\text{Agt}, \neg \psi, S)$ also holds. In other words, if the agent has a conflicting goal that implies $\neg \psi$, then ψ will not be adopted as a goal.

goal state would become inconsistent and it would want everything. This is a simple way of handling goal conflicts. A more interesting method would be to give more credence to requests from certain individuals, or requests of certain types. For example, if an agent gets a request from its owner that conflicts with a previous request from someone else, it should drop the previous request and adopt its owner's request instead. We reserve a more sophisticated handling of conflicting requests for future work.

We assume that request actions are always possible to execute, although executability conditions could easily be added.

Axiom 3.2 $\text{Poss}(\text{REQUEST}(\text{reqr}, \text{agt}, \psi), s)$.

We now turn our attention to goal contraction. Suppose that the owner of an agent asks it to do ψ and later changes his mind. The owner should be able to tell the agent to stop working on ψ . We use the action $\text{CANCELREQUEST}(\text{requester}, \text{agt}, \psi)$ for this purpose. This action causes agt to drop the goal that ψ . A CANCELREQUEST action can only be executed if a corresponding REQUEST action has occurred in the past.

Axiom 3.3 $\text{Poss}(\text{CANCELREQUEST}(\text{reqr}, \text{agt}, \psi), s) \equiv \text{do}(\text{REQUEST}(\text{reqr}, \text{agt}, \psi), s') \preceq s$

We handle CANCELREQUEST actions by determining what the W relation would have been if the corresponding REQUEST action had never happened. Suppose a $\text{CANCELREQUEST}(\text{Requester}, \text{Agt}, \psi)$ action occurs in situation S . We restore the W relation to the way it was before the corresponding REQUEST occurred. Then, starting just after the REQUEST , we look at all the situations $\text{do}(A^*, S^*)$ in the history of S and remove from W any situation S' that satisfies $W^-(\text{Agt}, A, S', S^*)$; such situations S' should be removed to reflect the adoption of a subsequent request. We first define the predicate $\text{CANCELS}(a, a')$, which says that action a cancels the action a' . In our case, only CANCELREQUEST actions cancel the corresponding REQUEST 's:

$$\begin{aligned} \text{CANCELS}(a, a') &\stackrel{\text{def}}{=} \\ &\exists \text{reqr}, \psi. a = \text{CANCELREQUEST}(\text{reqr}, \text{agt}, \psi) \wedge \\ &a' = \text{REQUEST}(\text{reqr}, \text{agt}, \psi). \end{aligned}$$

We now define W^+ which states the conditions under which s' is added to W^+ after a is executed in s :

$$\begin{aligned} W^+(\text{agt}, a, s', s) &\stackrel{\text{def}}{=} \\ &(\exists s_i. W(\text{agt}, s', s_i) \wedge \\ &(\exists a_1. \text{do}(a_1, s_i) \preceq s \wedge \text{CANCELS}(\text{agt}, a, a_1) \wedge \\ &\quad \forall a^*, s^*. \text{do}(a_1, s_i) \prec \text{do}(a^*, s^*) \preceq s \supset \\ &\quad \neg W^-(\text{agt}, a^*, s', s^*))) \end{aligned}$$

To help explain this definition, we will refer to Fig. 1, where a segment of a situation forest is shown. The situation S' is not W -related to S . However, S' was W -related to S_1 , which is in the history of S . The next action after S_1 in the history of S was $\text{REQUEST}(\text{Requester}, \text{Agt}, \psi)$. We suppose S' was dropped from W after the REQUEST . If none of the actions between $\text{do}(\text{REQUEST}(\text{Requester}, \text{Agt}, \psi), S_1)$ and

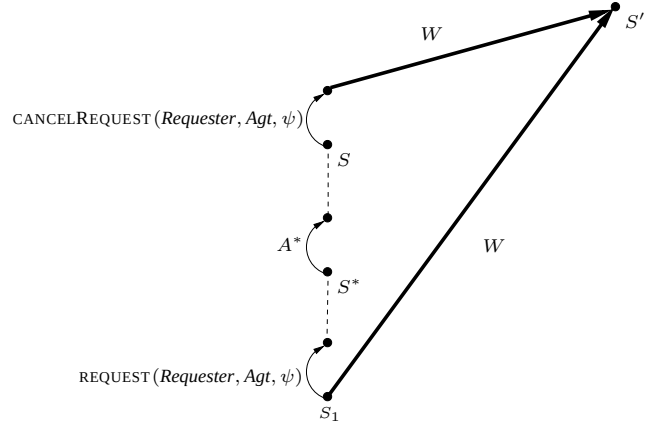


Figure 1: An example of goal contraction.

S also cause S' to be dropped from W , then S' is returned to W after the $\text{CANCELREQUEST}(\text{Requester}, \text{Agt}, \psi)$ action is executed in S . In other words, $W^+(\text{Agt}, \text{CANCELREQUEST}(\text{Requester}, \text{Agt}, \psi), S', S)$ holds because the following hold: $W(\text{Agt}, S', S_1)$, $\text{do}(\text{REQUEST}(\text{Requester}, \text{Agt}, \psi), S_1) \preceq S$, $\text{CANCELREQUEST}(\text{Requester}, \text{Agt}, \psi)$ cancels $\text{REQUEST}(\text{Requester}, \text{Agt}, \psi)$, and for every situation $\text{do}(A^*, S^*)$ between $\text{do}(\text{REQUEST}(\text{Requester}, \text{Agt}, \psi), S_1)$ and S , $\neg W^-(\text{Agt}, A^*, S', S^*)$ holds. Note that for our definition of W^+ to work properly, we must assume that there is only one REQUEST action in the history that is cancelled by each CANCELREQUEST . This assumption will be relaxed in future work.

4 Properties

We now consider some properties of goal change. Let Σ consist of the foundational, encoding, and unique names axioms, and the axioms from Sections 2 and 3. First, we show that our theory supports goal expansion. If an agent does not have $\neg\psi$ as a goal, then a request for ψ causes it to adopt ψ as a goal.

Theorem 4.1

$$\begin{aligned} \Sigma \models \forall \text{agt}, \psi, \text{requester}, s. \neg \text{Goal}(\text{agt}, \neg\psi, s) \supset \\ \text{Goal}(\text{agt}, \psi, \text{do}(\text{REQUEST}(\text{requester}, \text{agt}, \psi), s)) \end{aligned}$$

We also show the conditions under which an agent's only goals are expanded. If an agent has the only goal that ψ , ψ' is requested of the agent, the agent knows that the request does not affect the truth value of ψ , and ψ' does not conflict with the agent's goals, then its only goals expand to include ψ' .

Theorem 4.2

$$\begin{aligned} \Sigma \models \forall a, \text{agt}, \psi, \psi', \text{reqr}, s. \\ \text{OGoal}(\text{agt}, \psi, s) \wedge a = \text{REQUEST}(\text{reqr}, \text{agt}, \psi') \wedge \\ \text{Know}(\text{agt}, [\forall \text{then}. \text{Poss}(a, \text{Now}) \supset \\ (\psi[\text{Now}, \text{then}] \equiv \psi[\text{do}(a, \text{Now}), \text{then}])], s) \wedge \\ \neg \text{Goal}(\text{agt}, \neg\psi', s) \supset \\ \text{OGoal}(\text{agt}, \psi \wedge \psi', \text{do}(a, s)). \end{aligned}$$

Next, we examine the persistence of goals. A goal ψ persists over an action a , if a is not a cancel request, and the agent knows that if ψ holds then a does not change its value.

Theorem 4.3

$$\begin{aligned} \Sigma \models & \forall a, agt, \psi, s. \\ & \mathbf{Goal}(agt, \psi, s) \wedge \\ & (\forall reqr, \psi'. a \neq \text{CANCELREQUEST}(reqr, agt, \psi')) \wedge \\ & \mathbf{Know}(agt, [\forall then. \text{Poss}(a, Now) \wedge \psi[Now, then] \supset \\ & \quad \psi[do(a, Now), then]], s) \supset \\ & \mathbf{Goal}(agt, \psi, do(a, s)). \end{aligned}$$

We have a similar result for “only goals”.

Theorem 4.4

$$\begin{aligned} \Sigma \models & \forall a, agt, \psi, s. \\ & \mathbf{OGoal}(agt, \psi, s) \wedge \\ & [\forall requester, \psi'. \\ & \quad (a = \text{REQUEST}(requester, agt, \psi') \supset \\ & \quad \mathbf{Goal}(agt, \neg\psi', s)) \wedge \\ & \quad a \neq \text{CANCELREQUEST}(requester, agt, \psi')] \wedge \\ & \mathbf{Know}(agt, [\forall then. \text{Poss}(a, Now) \supset \\ & \quad (\psi[Now, then] \equiv \psi[do(a, Now), then]), s) \supset \\ & \mathbf{OGoal}(agt, \psi, do(a, s)). \end{aligned}$$

We now show a property about only goal contraction. Suppose that an agent only has the goal that ψ in s , and just after s , $reqr$ requests ψ' of the agent. In some future situation s'' , $reqr$ cancels its request that ψ' , yielding the situation s' . We assume that there was only one request of the agent for ψ' from $reqr$, and that no other REQUEST or CANCELREQUEST actions occurred between s and s' . We also assume that ψ always persists (this condition can be weakened; we assumed it to simplify the proof). Then, we can show that the agent only has the goal that ψ in s . In other words, the request for ψ' was indeed cancelled, and the other “only goals” persisted.

Theorem 4.5

$$\begin{aligned} \Sigma \models & \forall agt, \psi, \psi', reqr, s, s', s''. \mathbf{OGoal}(agt, \psi, s) \wedge \\ & do(\text{REQUEST}(reqr, agt, \psi'), s) \preceq s' \wedge \\ & s' = do(\text{CANCELREQUEST}(reqr, agt, \psi'), s'') \wedge \\ & [\forall s_1, s_2. do(\text{REQUEST}(reqr, agt, \psi'), s_1) \preceq s'' \wedge \\ & \quad do(\text{REQUEST}(reqr, agt, \psi'), s_2) \preceq s'' \supset \\ & \quad s_1 = s_2] \wedge \\ & [\forall s^*, a^*. \\ & \quad do(\text{REQUEST}(reqr, agt, \psi'), s) \prec do(a^*, s^*) \wedge \\ & \quad do(a^*, s^*) \preceq s'' \supset \\ & \quad (\neg \exists reqr', \psi''. \\ & \quad \quad a^* = \text{CANCELREQUEST}(reqr', agt, \psi'') \vee \\ & \quad \quad a^* = \text{REQUEST}(reqr', agt, \psi''))] \wedge \\ & [\forall a, s_1, s_2. \psi(s_1, s_2) \equiv \psi(do(a, s_1), s_2)] \supset \\ & \mathbf{OGoal}(agt, \psi, s'). \end{aligned}$$

Just as agents introspect their knowledge, we want agents to be able to introspect their goals, i.e., if an agent has a goal (does not have a goal, resp.) that ψ , it should know that it has (does not have, resp.) ψ as a goal. We identify constraints that yield these properties. They are constraints on K and

W . We will need the following definitions of transitivity and Euclideaness, starting in a situation s . Note that the order of the situations is reversed from the traditional definitions.

$$\begin{aligned} Ktrans(agt, s) & \stackrel{\text{def}}{=} \forall s_1, s_2. K(agt, s_1, s) \wedge K(agt, s_2, s_1) \supset \\ & \quad K(agt, s_2, s). \\ Keuc(agt, s) & \stackrel{\text{def}}{=} \forall s_1, s_2. K(agt, s_1, s) \wedge K(agt, s_2, s) \supset \\ & \quad K(agt, s_2, s_1). \end{aligned}$$

Theorem 6 of Scherl and Levesque [2003] showed that if K is initially reflexive, and transitive or Euclidean, then the successor state axiom for K guarantees that the properties are preserved over executable sequences of actions. This result carries over here. First, we show that reflexivity is preserved. Recall that we asserted that K was initially reflexive in Axiom 2.3. Our successor state axiom for K preserves reflexivity over all executable sequences of actions.

Theorem 4.6

$$\Sigma \models \forall agt, s. Executable(s) \supset K(agt, s, s),$$

where $Executable(s) \stackrel{\text{def}}{=} \forall a, s'. do(a, s') \preceq s \supset \text{Poss}(a, s')$.

Similarly, if K is initially transitive and Euclidean, then it remains so over any executable sequence of actions.

Theorem 4.7

$$\begin{aligned} \Sigma \models & (\forall agt, s. \text{Init}(s) \supset Ktrans(agt, s) \wedge Keuc(agt, s)) \supset \\ & (\forall agt, s. Executable(s) \supset Ktrans(agt, s)) \wedge \\ & (\forall agt, s. Executable(s) \supset Keuc(agt, s)), \end{aligned}$$

For positive introspection of goals, we need a constraint similar to transitivity, but that involves both K and W . We call this constraint *cross-transitivity*:

$$\begin{aligned} CrossTrans(agt, s) & \stackrel{\text{def}}{=} \\ & \forall s_1, s_2, s_3. K(agt, s_1, s) \wedge K(agt, s_2, s_1) \wedge \\ & \quad W(agt, s_3, s_1) \wedge s_2 \preceq s_3 \supset \\ & \quad W(agt, s_3, s). \end{aligned}$$

If this constraint is satisfied, and K is transitive, then the agents will have positive introspection of goals:

Theorem 4.8

$$\begin{aligned} \Sigma \models & \forall agt, s, \psi. Ktrans(agt, s) \wedge CrossTrans(agt, s) \supset \\ & (\mathbf{Goal}(agt, \psi, s) \supset \\ & \quad \mathbf{Know}(agt, \mathbf{Goal}(agt, \psi), s)). \end{aligned}$$

For negative introspection of goals, we need a constraint similar to Euclideaness, which we call *cross-Euclideaness*:

$$\begin{aligned} CrossEuc(agt, s) & \stackrel{\text{def}}{=} \\ & \forall s_1, s_2, s_3. K(agt, s_1, s) \wedge K(agt, s_2, s) \wedge \\ & \quad W(agt, s_3, s) \wedge s_2 \preceq s_3 \supset \\ & \quad W(agt, s_3, s_1). \end{aligned}$$

If this constraint is satisfied, and K is Euclidean, then the agents will have negative introspection of goals:

Theorem 4.9

$$\Sigma \models \forall agt, s. Keuc(agt, s) \wedge CrossEuc(agt, s) \supset (\neg \mathbf{Goal}(agt, \psi, s) \supset \mathbf{Know}(agt, \neg \mathbf{Goal}(agt, \psi), s)).$$

$CrossTrans$ and $CrossEuc$ persist if they hold of the initial situations, and K is initially transitive and Euclidean.

Theorem 4.10

$$\Sigma \models [\forall s'. Init(s') \supset (Ktrans(agt, s') \wedge Keuc(agt, s') \wedge CrossTrans(agt, s') \wedge CrossEuc(agt, s'))] \supset \forall agt, s. Executable(s) \supset CrossTrans(agt, s) \wedge CrossEuc(agt, s).$$

It follows that positive and negative goal introspection persist, if these constraints hold initially.

5 Meeting Scheduler Example

To illustrate how such a formalization of goal change is useful in specifying multiagent applications, we now summarize the meeting scheduler example from Shapiro [2005] that is based on the one given by Shapiro *et al.* [1998], but was modified to include goal contraction using CANCELREQUEST.

We specify the behavior of agents with the notation of the programming language ConGolog [De Giacomo *et al.*, 2000], the concurrent version of Golog [Levesque *et al.*, 1997]. Here are the ConGolog constructs we will use:

a ,	primitive action
$\phi?$,	wait for a condition
$\delta_1; \delta_2$,	sequence
$\delta_1 \mid \delta_2$,	nondeterministic choice of programs
if ϕ then δ_1 {else} δ_2 endIf ,	conditional
for $x \in L$ do δ endFor ,	for loop
while ϕ do δ endWhile	while loop

In the above, a denotes a situation calculus action; δ , δ_1 , and δ_2 stand for programs; and L is a finite list.

In the meeting scheduler example [Shapiro *et al.*, 1998; Shapiro and Lespérance, 2001; Shapiro, 2005], there are meeting organizer agents, which are trying to schedule meetings with personal agents, which manage the schedules of their (human) owners. To schedule a meeting, an organizer agent requests of each of the personal agents of the participants in the meeting to adopt the goal that its owner attend the meeting during a given time period. If a personal agent does not have any goals that conflict with its owner attending the meeting (i.e., it has not previously scheduled a conflicting meeting), it adopts the goal that its owner attend this meeting and informs the meeting organizer that it has adopted this goal, i.e., that it accepts the meeting request. Otherwise, the personal agent informs the meeting organizer that it has not

adopted the goal that its owner attend the meeting, i.e., that it declines the meeting request.

The procedure that defines the behaviour of meeting organizer agents (organizeMeeting) is given in Fig. 2.⁷ Their task is to organize a meeting for a given period and set of participants on behalf of the chair of the meeting. They do this by asking the personal agent of each participant to have their owner meet during the given period. The participants' personal agent will then reply whether they have declined the meeting. The behaviour of the personal agents is specified by the MANAGESCHEDULE procedure which is not shown here, due to space constraints. In that procedure, a personal agent responds to a meeting request by adopting the goal that its owner be at the requested meeting, unless it has already scheduled a conflicting meeting, and then informs the requester whether it has adopted the requested goal. The meeting organizer waits until it knows whether one of the participants' agent declined the meeting. This will happen when either someone has declined the meeting or everyone has agreed to it. If the organizer discovers that someone declined the meeting, it cancels its request for a meeting with the personal agents of all the participants in the meeting. The personal agents that agreed to the meeting will then drop the goal that their owners be at the requested meeting.

6 Conclusions and Future Work

In this paper, we defined the goals and “only goals” of an agent, and examined properties of these definitions: expansion, contraction, and persistence. We identified the restrictions on K and W that give us introspection of goals, and showed that these restrictions persist, if they hold initially. Note that our approach is quite different and simpler than most existing approaches to belief change. These require a specification of the plausibilities of alternate situations. Our approach avoids this and instead relies on the history to undo previous goal expansions. However, we do not handle arbitrary goal contractions, only cancellations of previous requests. Note there are some related approaches in the belief change literature [Booth *et al.*, 2005; Roorda *et al.*, 2003]. This raises the question of using belief change frameworks to handle goal change, or applying our goal change approach to belief change. It would be interesting to investigate whether there are essential differences between belief revision and goal revision that warrant different solutions, or whether the same solutions can be applied to both problems. Also, it would be interesting to identify postulates for goal change and examine how they differ from belief change postulates [Gärdenfors, 1988]. Our handling of the cancelling of requests needs further work and analysis. First of all, it does not properly handle nested requests for the same goal. For the successor state axiom to work properly for goal contraction, we must assume that there is only one REQUEST

⁷Note that for us (unlike De Giacomo *et al.*) procedures are simply program definitions and cannot be recursive. In the definition of this procedure, $PAG(p)$ is a function that maps an owner to its personal agent, $MEMBER(p, parts)$ holds if p is a member of the list $parts$, and $\mathbf{KWhether}(agt, \phi, s) \stackrel{\text{def}}{=} \mathbf{Know}(agt, \phi, s) \vee \mathbf{Know}(agt, \neg \phi, s)$.

$\text{INFORMWHETHER}(agt, agt', \phi) \stackrel{\text{def}}{=} [\text{Know}(agt, \phi)?; \text{INFORM}(agt, agt', \phi)] \mid$
 $[\text{Know}(agt, \neg\phi)?; \text{INFORM}(agt, agt', \neg\phi)] \mid [\neg\text{KWhether}(agt, \phi)?; \text{INFORM}(agt, agt', \neg\text{KWhether}(agt, \phi))]$

$\text{SomeoneDeclined}(per, chair, parts, s) \stackrel{\text{def}}{=} \exists p. \text{MEMBER}(p, parts) \wedge \neg\text{Goal}(\text{PAG}(p), \text{During}(per, \text{ATMEETING}(p, chair)), s)$

$\text{organizeMeeting}(oa, chair, parts, per) \stackrel{\text{def}}{=}$
for $p \in parts$ **do**
 $\text{REQUEST}(oa, \text{PAG}(p), \text{During}(per, \text{ATMEETING}(p, chair)))$ **endFor**;
 $\text{KWhether}(oa, \text{SomeoneDeclined}(per, chair, parts))?$;
if $\text{Know}(oa, \text{SomeoneDeclined}(per, chair, parts))$ **then**
for $p \in parts$ **do**
 $\text{cancelRequest}(oa, \text{PAG}(p), \text{During}(per, \text{ATMEETING}(p, chair)))$ **endFor** **endif**;
 $\text{INFORMWHETHER}(oa, chair, \text{SomeoneDeclined}(per, chair, parts))$

Figure 2: Procedure run by the meeting organizer agents.

action in the history that is cancelled by each CANCELREQUEST. This would not be too difficult to remedy, but it leads us to another difficulty. The definition of W^+ is already quite complex, which makes proving properties about it somewhat awkward. Since the definition refers to a sequence of situations, the proofs will often involve inductions. We would like to investigate the possibility of simplifying the definition.

References

- [Booth *et al.*, 2005] R. Booth, S. Chopra, A. Ghose, and T. Meyer. Belief liberation (and retraction). *Studia Logica*, 79:47–72, 2005.
- [Cohen and Levesque, 1990] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
- [De Giacomo *et al.*, 2000] G. De Giacomo, Y. Lespérance, and H. J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. The MIT Press: Cambridge, MA, 1988.
- [Konolige and Pollack, 1993] K. Konolige and M. E. Pollack. A representationalist theory of intention. In *Proc. IJCAI-93*, pages 390–395, 1993.
- [Levesque *et al.*, 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.
- [Levesque *et al.*, 1998] H. J. Levesque, F. Pirri, and R. Reiter. Foundations for a calculus of situations. *Electronic Transactions of AI (ETAI)*, 2(3–4):159–178, 1998.
- [McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, 1969.
- [Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Common Sense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1985.
- [Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.
- [Roorda *et al.*, 2003] J.-W. Roorda, W. van der Hoek, and J.-J. Meyer. Iterated belief change in multi-agent systems. *Logic Journal of the IGPL*, 11(2):223–246, 2003.
- [Scherl and Levesque, 2003] R. B. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.
- [Shapiro and Lespérance, 2001] S. Shapiro and Y. Lespérance. Modeling multiagent systems with the cognitive agents specification language — a feature interaction resolution application. In C. Castelfranchi and Y. Lespérance, editors, *Proc. ATAL-2000*, pages 244–259. Springer-Verlag, Berlin, 2001.
- [Shapiro *et al.*, 1998] S. Shapiro, Y. Lespérance, and H. J. Levesque. Specifying communicative multi-agent systems. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multi-Agent Systems — Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, Berlin, 1998.
- [Shapiro *et al.*, 2000] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. J. Levesque. Iterated belief change in the situation calculus. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proc. KR-2000*, pages 527–538, 2000.
- [Shapiro, 2005] S. Shapiro. PhD thesis. To appear, 2005.
- [Spohn, 1988] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Kluwer Academic Publishers, 1988.

First-Order Logical Filtering

Afsaneh Shirazi and Eyal Amir

Computer Science Department, University of Illinois at U-C
Urbana, IL 61801, USA {hajiamin,eyal}@cs.uiuc.edu

Abstract

Logical filtering is the process of updating a belief state (set of possible world states) after a sequence of executed actions and perceived observations. In general, it is intractable in dynamic domains that include many objects and relationships. Still, potential applications for such domains (e.g., semantic web, autonomous agents, and partial-knowledge games) encourage research beyond immediate intractability results.

In this paper we present polynomial-time algorithms for filtering belief states that are encoded as First-Order Logic (FOL) formulae. We sidestep previous discouraging results, and show that our algorithms are exact in many cases of interest. These algorithms accept belief states in full FOL, which allows natural representation with explicit references to unidentified objects, and partially known relationships. Our algorithms keep the encoding compact for important classes of actions, such as STRIPS actions. These results apply to most expressive modeling languages, such as partial databases and belief revision in FOL.

1 Introduction

Many everyday scenarios are dynamic and partially observable: a robot in one room cannot see the state of another room, a camera overlooking a bookshelf cannot detect the title of a book that is obscured, and one cannot readily observe the amount of money an agent has. Many applications in such domains compute information about the current world state (*belief state*, i.e., set of possible states or a distribution over such a set) after actions and observations. This computation is called *filtering* (also, state estimation, belief update, and database progression). They use this information to make decisions, answer questions, and explore.

Filtering is intractable in general for discrete domains [Eiter and Gottlob, 1992], and much research is dedicated to its approximation in stochastic domains (e.g., [Boyer and Koller, 1998]). Still, these approximations introduce unbounded errors many times, take unbounded computation time in others, and are not usable in most deterministic domains. Recent progress on logical methods for filtering of

propositional belief states (sets of states) [Amir and Russell, 2003] with actions and observations has shown that *exact* filtering is tractable when belief states are represented as propositional formulae, and certain natural assumptions are met. Still, many domains have propositional encodings that are too large or are not possible (e.g., large numbers of objects, unknown number of objects, and observations with partial knowledge about identity of objects).

In this paper we present tractable algorithms and theoretical results for updating belief states that are represented in First-Order Logic (FOL). These representations permit belief states of infinite sizes, uncertainty about the number of objects and their identity, and observations that do not distinguish between some objects. It also enables more compact representations than those of propositional logic.

We show that when actions map states 1:1, then we can update FOL belief-state formulae efficiently (linear time in the representation size), after prior compilation. We also show that the representation remains of bounded polynomial size for two classes of actions, including those for which actions have simple case preconditions and actions with STRIPS-like preconditions (non-conditional, and observed success/failure). For those we also present filtering algorithms that do not require precompilation for efficient update. This is in surprising contrast to the common belief that FOL cannot be used efficiently for representing and updating partial knowledge [Winslett, 1990; Lin and Reiter, 1997].

On the way to these contributions we form the foundations and provide a theory for FOL belief update. We relate deterministic Situation Calculus [Reiter, 2001] with a first-order transition model [Blass and Gurevich, 2000]. There, every belief state is a set of FOL models over the FOL language of a state. We show that filtering such belief states can be captured exactly by *deduction in FOL*, if the result of the filtering is definable in FOL (this is the best we can hope for [Lin and Reiter, 1997]). Also, we show that deduction can be carried out one time step at a time.

Most work related to ours is limited to the propositional case (e.g., [Amir and Russell, 2003; Boyen and Koller, 1998]). Important exceptions are [Cravo *et al.*, 2001; Dixon and Wobcke, 1993] (First-Order AGM belief revision), [Winslett, 1990] (belief update and revision in simple subclasses of FOL), and [Lin and Reiter, 1997] (progression in

Situation Calculus). The important difference that we draw with these works is that ours provides efficient inference procedures, while others focus on the use of general-purpose theorem provers, and intractability results.

2 Semantics of First-Order Filtering

In this section, we study logical filtering with first-order structures. A *first-order language* has as nonlogical symbols, the variables, the function symbols and the predicate symbols. A 0-ary function symbol is called *constant*. Note that among the binary predicate symbols must be the equality symbol $=$. We define the *terms* and *formulas* by the generalized inductive definition. Variables and functions are terms. Predicates are atomic formulas.

A *first-order language* is a language in which the symbols and formulas are as described above. We now turn to a description of the semantics of first-order languages. A *structure* S for a first-order language consists of:

1. $|S|$, the nonempty *universe* or *domain* of the structure S . The elements of $|S|$ are called the *individuals* of S .
2. For each n -ary predicate symbol p , $p^S \subseteq |S|^n$. These tuples of the universe are those tuples on which p is true.
3. For each n -ary function symbol f , $f^S : |S|^n \rightarrow |S|$. (In particular, for each constant e , e^S is an individual of S)

When a sentence ψ is *true* in a structure S , we denote it by $\models_S \psi$. For example, suppose that in structure S , $|S| = \{B, R\}$, for predicate *in*, $in^S = \{\langle B, R \rangle\}$, for constant *CS-R*, $CS-R^S = \{R\}$, and for constant *CS-B*, $CS-B^S = \{B\}$. This world has a CS room (CS-R), a CS book (CS-B), and a predicate *in* which indicates whether a book is in a room. By this definition, sentence *in*(CS-B, CS-R) is true in S .

We define logical filtering using situation calculus. The one that we use is compatible with the *basic action theory* of [Reiter, 2001]. The basic action theory has the form $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{s_0}$ where:

- Σ are the foundational axioms for situations.
- $\mathcal{D}_{ss}$ is a set of successor state axioms.
- $\mathcal{D}_{ap}$ is a set of action precondition axioms.
- $\mathcal{D}_{una}$ is a set of unique name axioms for actions. $\mathcal{A}(\vec{x}) \neq \mathcal{A}'(\vec{y})$, $\mathcal{A}(\vec{x}) = \mathcal{A}(\vec{y}) \Rightarrow \vec{x} = \vec{y}$ where $\mathcal{A}$ and $\mathcal{A}'$ are action symbols.
- $\mathcal{D}_{s_0}$ (the initial database) is a finite set of first-order sentences that are *uniform* in s_0

Definition 2.1 (Uniform Formula). A formula is *uniform* in s if s is the only term of sort *situation* mentioned by that formula.

We define precondition axioms and successor state axioms as a part of basic action theory as follows.

Definition 2.2 (Action Precondition Axioms). An action precondition axiom is a sentence of the form:¹

$$Poss(a(x_{1:n}), s) \Leftrightarrow precond_a(x_{1:n}, s)$$

where a is an n -ary action symbol, and $precond_a(x_{1:n}, s)$ is a formula that is *uniform* in s and whose free variables are among $x_1, \dots, x_n, s$.

¹ $x_{1:n}$ is the abbreviation for $x_1, \dots, x_n$

Generally, the values of relations and functions in a dynamic world will vary from one situation to the next. Relations whose truth values vary from situation to situation are called *relational fluents*. They are denoted by predicate symbols taking a situation term as their last argument. Same argument is true about functions.

Definition 2.3 (Successor State Axioms). Successor state axioms is defined for either a relational fluent or a functional fluent. A successor state axiom for an n -ary relational fluent p is a sentence of the form:

$$Poss(a(x_{1:n}), s) \Rightarrow \forall y_1, \dots, \forall y_m \\ (p(y_{1:m}, do(a(x_{1:n}), s)) \Leftrightarrow succ_{p,a}(x_{1:n}, y_{1:m}, s))$$

where a is an action symbol, and $succ_{p,a}(x_{1:n}, y_{1:m}, s)$ is a formula that is *uniform* in s and whose free variables are among $x_1, \dots, x_n, y_1, \dots, y_m, s$. We define a successor state axiom for a functional fluent in a similar way.

All changes to the world are the result of named actions. An action may be parameterized. For example, *move*(b, r_1, r_2) stands for the action of moving object b from room r_1 to room r_2 . The intended interpretation is that situations are finite sequences of actions, and $do(a, s)$ denotes the sequence formed by adding action a to the sequences s . In other words, $do(a, s)$ is the successor situation resulting from performing the action a . We use situation calculus as foundations and semantics. Later (section 3 onwards) we do not mention it because we always focus on belief states. However, it is used in the proofs of theorems and our results are applicable to it.

Example Consider a book keeper robot who lives in a world consisting of rooms. When the robot is in a room, it can make observations about the books in that room. It can move a book between rooms, it can return a book to the library from an arbitrary room or it can put a borrowed book in a room. So possible actions are *move*(b, r_1, r_2), *return*(b, r) or *borrow*(b, r). The actions which are executable in a world state can change the value of different predicates or functions. Predicates are *room*(r), *book*(b) or *in*(b, r). There are no functional fluents except constants.

We define a precondition axiom and a successor state axiom for action *move* and skip the others.

- **Precondition Axiom:**
 $Poss(move(b, r_1, r_2), s) \Leftrightarrow book(b, s) \wedge room(r_1, s) \wedge room(r_2, s) \wedge in(b, r_1, s)$
- **Successor State Axioms:**
 $Poss(move(b, r_1, r_2), s) \Rightarrow$
 $\forall b', r' (in(b', r', do(move(b, r_1, r_2), s)) \Leftrightarrow$
 $((b' = b) \wedge (r' = r_1)) \Rightarrow false$
 $\wedge ((b' = b) \wedge (r' = r_2)) \Rightarrow true$
 $\wedge (\neg((b' = b) \wedge (r' = r_1)))$
 $\wedge \neg((b' = b) \wedge (r' = r_2))) \Rightarrow in(b', r', s)))$ ■

The definition of progression and filtering semantics are as follows.

Definition 2.4 (Transition Relation of Structures). For an action theory $\mathcal{D}$ and a structure S , we define a transition re-

lation $\mathcal{R}_D(S, a, S')$ as follows.

$$\begin{aligned}\mathcal{R}_D &= \{ \langle S, a(u_{1:n}), S' \rangle \mid \models_S \text{precond}_a(u_{1:n}), |S'| = |S|, \\ p^{S'} &= \{ \langle s_{1:m} \rangle \in |S|^m \mid \models_S \text{succ}_{p,a}(u_{1:n}, s_{1:m}) \}, \\ f^{S'} &= \{ \langle s_{1:m}, s_r \rangle \in |S|^{m+1} \mid \\ &\quad \models_S \text{succ}_{f,a}(u_{1:n}, s_{1:m}, s_r) \} \end{aligned}$$

We use $[x/v]$ as a notion of substitution in which x is a vector of variables and v is a vector of variables and constants. $[x/v]$ is a shorthand for $[x_1/v_1, \dots]$ in which $[x_1/v_1]$ means replacing all instances of symbol x_1 by symbol v_1 .

Definition 2.5 (Logical Filtering Semantics). Let σ be a set of first-order structures. We refer to it as belief state. The filtering of a sequence of actions (ground or not) and observations $\langle a_1, o_1, \dots, a_t, o_t \rangle$ is defined as follows (ϵ refers to the empty sequence).

1. $\text{Filter}[\epsilon](\sigma) = \sigma$;
2. $\text{Filter}[a](\sigma) = \{ S' \mid S \in \sigma, \langle S, \hat{a}, S' \rangle \in \mathcal{R}_D, \hat{a} = a_{[x/v]} \}$;
3. $\text{Filter}[o](\sigma) = \{ S \in \sigma \mid \models_S o \}$;
4. $\text{Filter}[\langle a_i, o_i, \dots, a_t, o_t \rangle](\sigma) = \text{Filter}[\langle a_{i+1}, o_{i+1}, \dots, a_t, o_t \rangle](\text{Filter}[a_i](\sigma))$.

We call Step 2 progression with a and Step 3 filtering with o .

3 Filtering of FOL Formulae

In the above definition, filtering is applied to a set of first-order structures. In this section we use FOL to represent belief states and we update this representation without explicit reference to the structures in a belief state.

3.1 FOL Theories and Belief States

A belief state formula is a first-order theory (possibly infinite) that represents belief state. A structure is in belief state if and only if it satisfies the belief state formula. The use of a theory instead of a formula is required because the set of first-order consequences of a first-order formula is infinite and may not be representable by a FOL formula.

For simplicity, we use the same logical connectives that we have for FOL formulas. The meaning of those connectives on first-order theories is as follows:

If φ, ψ are theories of infinite size, then

- $\varphi \wedge \psi$ will mean $\varphi \cup \psi$.
- $\varphi \vee \psi$ will mean $\{ \alpha \vee \beta \mid \alpha \in \varphi, \beta \in \psi \}$ (similar to de-morgan law).
- Whenever applying negation ($\neg$) we will assume that φ is a finite theory (thus, a logical formula)
- Same for $\exists$ (can replace $\exists$ with a new constant symbol)

Thus, in the rest of the paper, whenever we say "belief state formula", we refer to a FOL theory, unless sanctioned otherwise as above. From now on, we assume that our first-order language has no function symbols except constants.

3.2 Basic Algorithm

In this section, we show how we can progress an initial database represented by a logical formula after applying a single action or observation. The result of progression is a new database that progression algorithm can use afterwards.

We define a predicate corresponding to each relational fluent whose truth value does not depend on the situation. The snapshot of system at time t only shows the truth values of predicates. The truth values of predicates would change while moving from one situation to the next. We represent predicates with the same symbol as relational fluents but with different arity.

Suppose that $\mathcal{P} = \{g_1, \dots, g_r\}$ is the set of all constants and predicates. We define a new set of symbols $\mathcal{P}' = \{g'_1, \dots, g'_r\}$ such that $g'_i(y_{1:n}) = g_i(y_{1:n})_{[\mathcal{P}/\mathcal{P}]}$ where $[\mathcal{P}/\mathcal{P}']$ is a shorthand for $[g_1/g'_1, \dots, g_r/g'_r]$. We view $\mathcal{P}$ as the set of predicates in situation s , and $\mathcal{P}'$ as the set of predicates in situation $do(a, s)$.

We filter a belief-state formula as follows. (We reuse $\text{Filter}[\cdot](\cdot)$ for filtering a belief-state formula.) Let ψ be a belief state formula, $a(u_{1:n})$ be a grounded action, $Cn(\Psi)$ be the set of logical consequences of Ψ (i.e. formulae ϕ such that $\Psi \models \phi$), and $Cn^L(\Psi)$ be the set of logical consequences of Ψ in the language $\mathcal{L}$. We write $Cn^L(\Psi)$, when L is a set of symbols, to mean $Cn^{\mathcal{L}(L)}(\Psi)$.

1. $\text{Filter}[a(u_{1:n})](\psi) = (Cn^{\mathcal{P}'}(\psi \wedge \text{precond}_a(u_{1:n})) \wedge \bigwedge_i \forall y_{1:m}, p'_i(y_{1:m}) \Leftrightarrow \text{succ}_{p_i,a}(u_{1:n}, y_{1:m}) \wedge \bigwedge_i \forall y_{1:m} \forall z, f'_i(y_{1:m}) = z \Leftrightarrow \text{succ}_{f_i,a}(u_{1:n}, y_{1:m}, z))_{[\mathcal{P}'/\mathcal{P}]}$
2. $\text{Filter}[o](\psi) = \psi \wedge o$ (1)

When we filter with action a we assert implicitly that its precondition held in the last world state. If the action is not executable on the belief state, the new belief state would be false which indicates an empty set. We prove in the following theorem that this definition of filtering approximates the semantics of definition 2.5.

Theorem 3.1. Let ψ be a belief state formula and a be an action, then

$$\text{Filter}[a](\{s \mid \models_s \psi\}) \subseteq \{s' \mid \models_{s'} \text{Filter}[a](\psi)\}$$

PROOF See section A.1 ■

[Lin and Reiter, 1997] showed that progression is not always first-order definable. However, they proved that progression always exists as a set of second order sentences for finite initial databases. Therefore, the two sides in theorem 3.1 are not equivalent since formula (1) is in FOL. In other words, FOL is not strong enough to model the progression of the initial database. However, the following corollary shows that the two sides of theorem 3.1 would be equal if the progression of a database is FOL definable.

Corollary 3.2. Let ψ be a first-order belief state formula. If FOL can represent the progression of ψ after performing action a , then

$$\text{Filter}[a](\{s \mid \models_s \psi\}) = \{s' \mid \models_{s'} \text{Filter}[a](\psi)\}$$

From this point, we assume that progression is first-order definable. Our basic algorithm computes $\text{Filter}[\langle a_1, o_1, \dots, a_t, o_t \rangle](\psi)$ by iteratively applying filtering of a belief-state formula with an action and an observation. It sets $\psi_0 = \psi$ and $\psi_i = \text{Filter}[o_i](\text{Filter}[a_i](\psi_{i-1}))$

recursively for $i > 0$ using the equations defined above. This algorithm is correct, as shown by corollary 3.2. It can be implemented using a first-order consequence finder.

3.3 Sequences of Actions and Observations

This section shows that iterative applications of progression steps lose no information. Thus, we can throw away the previous database and start working with the new one after performing each action.

We break our action theory $\mathcal{D}$ into two parts, the initial database $\mathcal{D}_{s_0}$ and the rest $\mathcal{D}_g$. Therefore, $\mathcal{D} = \mathcal{D}_g \cup \mathcal{D}_{s_0}$. Now we define the language of an action theory as follows.

Definition 3.3. The language of $\mathcal{D}$, $\mathcal{L}(\mathcal{D})$, is a set of first-order formulae whose predicate and function symbols occur in $\mathcal{D}$.

For instance, if $\mathcal{D}_{s_0} = \text{put}(A, B) \wedge \forall x \text{ box}(x)$, then $\text{put}(A, A)$ and $\forall x \exists y \text{ put}(x, y)$ are in $\mathcal{L}(\mathcal{D}_{s_0})$ but $\text{box}(A, B)$ is not.

In progression we assume that a ground action a is performed, and we are looking for a set of sentences $\mathcal{D}_{s_a}$ that can serve as a new initial database (s_a denotes the situation term $\text{do}(a, s)$). Unfortunately [Lin and Reiter, 1997] showed that $\mathcal{D}_{s_a}$ is not always first-order definable.

We define $\mathcal{F}_{s_a}$ as the set of first-order sentences uniform in s_a entailed by $\mathcal{D}$. If we use $\mathcal{F}_{s_a}$ instead of $\mathcal{D}_{s_a}$, for every first-order sentence ψ about the future of s_a , $\mathcal{F}_{s_a} \cup \mathcal{D}_g \models \psi$ iff $\mathcal{D} \models \psi$. The following theorem states this result.

Note that the intersection of all consequences of the action theory with $\mathcal{L}(\mathcal{D}_g \cup \{s_a\})$ is uniform in s_a .

Theorem 3.4. Let $\mathcal{D}_0$ be an action theory, and define $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}'_2$ as follows.

$$\begin{aligned} \mathcal{D}_1 &= Cn(\mathcal{D}_0 \cup \{s_1 = \text{do}(a_1, s_0)\}) \cap \mathcal{L}(\mathcal{D}_g \cup \{s_1\}) \\ \mathcal{D}_2 &= Cn(\mathcal{D}_1 \cup \{s_2 = \text{do}(a_2, s_1)\}) \cap \mathcal{L}(\mathcal{D}_g \cup \{s_2\}) \\ \mathcal{D}'_2 &= Cn(\mathcal{D}_0 \cup \{s_1 = \text{do}(a_1, s_0), s_2 = \text{do}(a_2, s_1)\}) \\ &\quad \cap \mathcal{L}(\mathcal{D}_g \cup \{s_2\}) \end{aligned}$$

(a_1 and a_2 are two actions in $\mathcal{D}_0$, not necessarily different. s_0, s_1 and s_2 do not occur in $\mathcal{D}_g$.) Then, $\mathcal{D}_2 = \mathcal{D}'_2$.

PROOF See section A.2 ■

For instance in our book keeper example, if $\mathcal{D}_{s_0}$ is $\{\text{book}(B, s_0), \text{room}(R, s_0), \text{in}(B, R, s_0)\}$ and the first action is $\text{return}(B, R)$, $\mathcal{D}_{s_1}$ would be $\{\text{book}(B, s_1), \text{room}(R, s_1), \neg \text{in}(B, R, s_1)\}$.

4 Factored Inference

Several distribution properties hold for logical filtering. We can decompose the filtering of a formula φ along logical connectives $\wedge, \vee, \neg, \forall, \exists$.

Theorem 4.1. Let a be an action, and let φ, ψ be first-order formulae. Then,

1. $\text{Filter}[a](\varphi \vee \psi) \equiv \text{Filter}[a](\varphi) \vee \text{Filter}[a](\psi)$
2. $\models \text{Filter}[a](\varphi \wedge \psi) \Rightarrow \text{Filter}[a](\varphi) \wedge \text{Filter}[a](\psi)$
3. $\models \text{Filter}[a](\neg \varphi) \Leftarrow \neg \text{Filter}[a](\varphi) \wedge \text{Filter}[a](\text{TRUE})$
4. $\text{Filter}[a](\exists x \varphi(x)) \equiv \exists x \text{Filter}[a](\varphi(L))_{[L/x]}$

(L is a fresh constant symbol.) 3, 4 hold only when $\text{Filter}[a](\varphi)$ is finite.

We can say something stronger for actions that act as *permutations* on the structures in which they are executable.

Definition 4.2 (Permuting Actions). Action a is permuting (1:1) if for every structure S' there is at most one S such that $\mathcal{R}_{\mathcal{D}}(S, a, S')$.

Domains that only include permuting actions are called *permutation domains*.

Theorem 4.3 (Distribution for Permutation Domains). Let a be a permuting action, and let φ, ψ be formulae. Then,

1. $\text{Filter}[a](\varphi \vee \psi) \equiv \text{Filter}[a](\varphi) \vee \text{Filter}[a](\psi)$
2. $\text{Filter}[a](\varphi \wedge \psi) \equiv \text{Filter}[a](\varphi) \wedge \text{Filter}[a](\psi)$
3. $\text{Filter}[a](\neg \varphi) \equiv \neg \text{Filter}[a](\varphi) \wedge \text{Filter}[a](\text{TRUE})$
4. $\text{Filter}[a](\exists x \varphi(x)) \equiv \exists x \text{Filter}[a](\varphi(L))_{[L/x]}$

We can decompose every first-order formula into a set of single literals by using distribution properties proved above. For instance, $\forall x (\varphi(x) \wedge \psi(x))$ is equivalent to $\forall x \varphi(x) \wedge \forall x \psi(x)$ so rule 2 can break it into two parts. Also $\forall x \neg \varphi(x)$ is equivalent to $\neg \exists x \varphi(x)$ so rule 3 and rule 4 can be used, and $\forall x (\varphi(x) \vee \psi(x))$ is equivalent to $\neg \exists x (\neg \varphi(x) \wedge \neg \psi(x))$ so rule 3, rule 4, and rule 2 can be used.

In permutation domains, we decompose the formula down to a set of grounded first-order single literals, and for filtering a single literal we use formula (1).

Our factored filtering (FF) algorithm for permutation domains is presented in Figure 1. It relies on theorems 3.2, 4.1, and 4.3. The number of different grounded single literals would be finite, if the number of objects is finite. Therefore, we can calculate filtering of all single literals as a preprocessing step and retrieve it later in finite domains.

Note that the arguments of these literals are either the constants associated to existential quantifiers or the constants which are mentioned in the initial belief state, the set of axioms or the observations.

PROCEDURE FF($\langle a_i, o_i \rangle_{0 < i \leq t}, \psi$) $\forall i, a_i$ an action, o_i an observation, ψ a belief-state formula. 1. if $t = 0$, return ψ. 2. return $o_t \wedge \text{FF-Step}(a_t, \text{precond}_{a_t} \wedge \text{FF}(\langle a_i, o_i \rangle_{0 < i \leq (t-1)}, \psi))$.
--

PROCEDURE FF-Step(a, ψ) a an action. ψ a belief-state formula. 1. if ψ is a single literal, then return Single-Literal-Filtering(a, ψ). 2. else, use distribution properties, call FF-Step recursively on sub-formulas of ψ.
--

Figure 1: Filtering of a FOL formula when all the actions are permuting actions.

Theorem 4.4. The algorithm FF is correct, and if the filtering of all single literals are given, the algorithm FF would run in time $O(|\text{precond}_a \wedge \psi|)$, where ψ is a belief state formula.

Our factored filtering algorithm uses consequence finding tools. Since it is part of preprocessing, it does not affect the

runtime of the system. In open systems the time is different since new objects may be added during the operation of the system. In these systems filtering of new single literals should be computed while system is running.

5 Filtering Algorithms for Different Domains

Our naive filtering algorithm uses consequence finding tools which do not scale to large domains. The following theorem suggests a different reasoning procedure.

Theorem 5.1. *Let a be an action, ψ be a belief state formula, and $\Phi(\varphi_{1:n})$ be a first-order logical formula whose atomic subformulas are among $\varphi_1, \dots, \varphi_n$. Then,*

$$\text{Filter}[a](\psi) \equiv \{\Phi(p_{1:n}) \mid \text{FOL formula } \Phi, \\ \psi \wedge \text{precond}_a \models \Phi(\text{succ}_{p_1,a}, \dots, \text{succ}_{p_n,a})\} \quad (2)$$

In this formula, all possible Φ s should be considered. In general, generating all Φ s is impossible because there are infinitely many such Φ s. In the following sections, we provide simpler closed-form solutions for two special cases of dynamic domains. These give rise to practical (polynomial) algorithms.

5.1 Unit-Case Successor State Axioms

By definition of successor state axioms, for every pair of actions and predicates exactly one successor state axiom is provided. The successor state axiom for action a and predicate p_i can be rewritten as:

$$\text{Poss}(a(x_{1:n}), s) \Rightarrow \forall y_1, \dots, \forall y_m (p_i(y_{1:m}, \text{do}(a, s)) \Leftrightarrow \\ (case_i^1 \Rightarrow \phi_i^1) \wedge \dots \wedge (case_i^{l_i} \Rightarrow \phi_i^{l_i}) \\ \wedge (\neg case_i^1 \wedge \dots \wedge \neg case_i^{l_i} \Rightarrow \phi_i^{l_i+1}))$$

where $case_i^j$ is of the form $(y_{j_1} = x_{j_1}) \wedge \dots \wedge (y_{j_k} = x_{j_k})$ (variable x_{j_k} is an argument of action a and variable y_{j_1} is an argument of predicate p) and each variable assignment satisfies at most one of the cases. A successor state axiom is called *unit-case successor state axiom* if it can be rewritten in a form where every ϕ_i^j ($1 \leq j \leq l_i + 1$) is a unit clause.

We break a unit-case successor state axiom into multiple instantiated axioms. Instantiated successor state axioms for predicate p_i are:

- $\text{Poss}(a(x_{1:n}), s) \Rightarrow (p_i(y_{1:m}, \text{do}(a, s)) \Leftrightarrow \phi_i^j)_{[y_i^j/x_i^j]}$ for all $1 \leq j \leq l_i$
- $\text{Poss}(a(x_{1:n}), s) \Rightarrow \forall y_{1:m} (\neg case_i^1 \wedge \dots \wedge \neg case_i^{l_i} \Rightarrow (p_i(y_{1:m}, \text{do}(a, s)) \Leftrightarrow \phi_{l_i+1}))$

$[y_i^j/x_i^j]$ is the substitution corresponding to $case_i^j$ (y_i^j and x_i^j are sequences of variables). This process is called *breaking into cases*. Note that all instantiated successor state axioms are in the form $\text{Poss}(a) \Rightarrow (cond_i \Rightarrow (p_i \Leftrightarrow \phi_i))$ where in some of them $cond_i$ is true (i is an enumeration of all instantiated successor state axioms of action a).

Figure 2 shows the unit-case filtering (UCFilter) algorithm. This algorithm is applicable on permutation domains whose successor state axioms are unit-case. The algorithm UCFilter is actually a way to compute every $\Phi(\text{subsucc}_a^1, \dots, \text{subsucc}_a^k)$ in formula (2). In permutation domains, the head of entailment in formula (2) is a single literal (action precondition can be considered as a conjunct to

PROCEDURE UCFilter ($\langle a_i, o_i \rangle_{0 \leq i \leq t}, \psi$) $\forall i, a_i$ an action, o_i an observation, ψ a belief-state formula. 1. if $t = 0$, return ψ . 2. $\psi_{t-1} = \text{UCFilter}(\langle a_i, o_i \rangle_{0 \leq i \leq (t-1)}, \psi)$. 3. return $o_t \wedge \text{Filter-True}(a_t) \wedge \text{UCStep}(a_t, \text{precond}_{a_t} \wedge \psi_{t-1})$.
PROCEDURE Filter-True (a) a an action. 1. $\text{poss}(a) \Rightarrow (cond_i \Rightarrow (p_i \Leftrightarrow \phi_i))$ an instantiated successor state axiom ($1 \leq i \leq k$). 2. $S = \emptyset$ 3. for all $1 \leq i, j \leq k$, (a) if $\phi_i = \text{true}$, add $case_i \Rightarrow p_i$ to S (b) elseif $\phi_i = \text{false}$, add $case_i \Rightarrow \neg p_i$ to S (c) elseif unifiable(ϕ_i, ϕ_j), add $((cond_i \wedge cond_j) \Rightarrow (p_i \Leftrightarrow p_j))_{mgu(\phi_i, \phi_j)}$ to S (d) elseif unifiable($\phi_i, \neg \phi_j$), add $((cond_i \wedge cond_j) \Rightarrow (p_i \Leftrightarrow \neg p_j))_{mgu(\phi_i, \phi_j)}$ to S (e) elseif $\phi_i = \forall x q(x), \phi_j = q(t)$, add $(cond_i \wedge cond_j) \Rightarrow (\neg p_i \vee p_j)$ to S (f) elseif $\phi_i = \exists x q(x), \phi_j = q(t)$, add $(cond_i \wedge cond_j) \Rightarrow (p_i \vee \neg p_j)$ to S 4. return $\bigwedge_{\varphi \in S} \varphi$.
PROCEDURE UCStep (a, ψ) a an action, ψ a belief-state formula. 1. if ψ is a single literal, then (a) $\text{poss}(a) \Rightarrow (cond_i \Rightarrow (p_i \Leftrightarrow \phi_i))$ an instantiated successor state axiom ($1 \leq i \leq k$). (b) $S = \emptyset$ (c) for all $1 \leq i \leq k$, i. if unifiable(ϕ_i, ψ), add $(case_i \Rightarrow p_i)_{mgu(\phi_i, \psi)}$ to S ii. elseif unifiable($\phi_i, \neg \psi$), add $(case_i \Rightarrow \neg p_i)_{mgu(\phi_i, \psi)}$ to S (d) return $\bigwedge_{\varphi \in S} \varphi$. 2. else, use distribution properties, call UCStep recursively on sub-formulae of ψ .

Figure 2: Unit-Case Filtering.

the belief state formula, and the distribution properties can be used). Consequently, $\Phi(\text{subsucc}_a^1, \dots, \text{subsucc}_a^k)$ is either equivalent to that literal or a tautology. A tautology is at most of size two when unit-case successor state axioms are used. Therefore, we can compute all desired Φ s in a finite number of steps.

Theorem 5.2. *Let k be the number of successor state axioms after breaking into cases, and ψ be the belief state formula. If each predicate has arity at most R , then algorithm UCFilter returns the filtering of ψ with action a in time $O(R \cdot k^2 + R \cdot k \cdot |\psi \wedge \text{precond}_a|)$. The length of new belief state formula is $O(R \cdot k^2 + R \cdot k \cdot |\psi \wedge \text{precond}_a|)$.*

Corollary 5.3. *Given a sequence of t actions and observations, algorithm UCFilter returns the filtering of ψ_0 in time $O(t^2 \cdot R^t \cdot k^{t+1} + t \cdot R^t \cdot k^t \cdot |\psi_0|)$. The length of belief state formula after t step is $\psi_t = O(t \cdot R^t \cdot k^{t+1} + R^t \cdot k^t \cdot |\psi_0|)$. (If the length of all observations and preconditions of actions are negligible compared to the length of belief state formula)*

5.2 STRIPS Domains

In STRIPS domains every action has no conditional effects. It means that the value of each predicate either changes to true, changes to false, or remains the same. STRIPS actions are not necessarily permuting. Consequently STRIPS successor state axioms can not be treated by algorithm UCFILTER even though they are unit-case. Successor state axioms in STRIPS domains are of the form:

$$Poss(a(x_{1:n}), s) \Rightarrow \forall y_1, \dots, \forall y_m (p_i(y_{1:m}, do(a, s)) \Leftrightarrow (case_i^1 \Rightarrow \phi_i^1) \wedge \dots \wedge (case_i^{l_i} \Rightarrow \phi_i^{l_i}) \wedge (\neg case_i^1 \wedge \dots \wedge \neg case_i^{l_i} \Rightarrow p_i(y_{1:m}, s)))$$

where ϕ_i^j ($j \leq l_i$) is either true or false.

A STRIPS action affects some of the instantiated predicates and keeps the value of the others. We refer to the set of affected instantiated predicates as $Eff(a)$.

$$Eff(a) = \{p(\vec{v}) \mid \text{action } a \text{ affects every instance of } p(\vec{v})\}$$

where v is a sequence of variables and constant symbols.

The first-order STRIPS filtering (FOSF) algorithm is presented in figure 3. The belief state formula which is an input to this algorithm should be in the form $\exists^* \forall^* \phi$ (EAFOL) (EAFOL is a first-order formula in which there is no existential quantifier inside a universal one). A clause in the clausal form of ϕ would be splitted into multiple instantiated clauses if one of its literals has some instances in $Eff(a)$ and some out of $Eff(a)$. This step is called *splitting into pure literal clauses*. We split every clause $p(x) \vee \varphi$ in which some instantiations of $p(x)$ are affected and some are not, into:

$$\{p(v_1) \vee \varphi, \dots, p(v_r) \vee \varphi, ((x \neq v_1 \wedge \dots \wedge x \neq v_r) \Rightarrow p(x)) \vee \varphi\}$$

where $p(v_1), \dots, p(v_r)$ are in $Eff(a)$. Note that we treat $((x \neq v_1 \wedge \dots \wedge x \neq v_r) \Rightarrow p(x))$ as a single literal.

The new clausal form of ϕ is divided into two parts, clauses with no literal in $Eff(a)$ and clauses that have at least one literal in $Eff(a)$. The first part directly goes to the new belief state formula. The algorithm adds all the consequences of the second part in which no affected literals exist, to the new belief state formula. All literals in $Eff(a)$ are also added to new belief state formula as their values can be determined unconditionally after applying the action.

Theorem 5.4. *Given action a , observation o , and belief state formula ψ in EAFOL, algorithm FOSF returns the filtering of ψ with a and o in time $O(R \cdot |E| \cdot \min((\frac{|E|}{2})^{2|Eff(a)|}, |Eff(a)|2^{|E| \cdot R}) + |\psi|)$ where E is the set of all clauses with any literal in $Eff(a)$ after splitting into pure literal clauses and each clause has length at most R .*

Theorem 5.5. *If ψ is in 2-FO-CNF² then the time complexity of FOSF after filtering one action is $O(|Eff(a)| \cdot |E|^2 + |\psi|)$. The formula length is $O(|Eff(a)| + |E|^2 + |\psi|)$.*

Extended Example

Consider our previous book-keeping robot. Suppose that we have two rooms, a CS room and an ECE room, and two books,

²A first-order formula is in k-FO-CNF if in clausal form the size of each clause is at most k.

PROCEDURE FOSF($\langle a_i, o_i \rangle_{0 < i \leq t}, \psi$) $\forall i, a_i$ an action, o_i an observation and ψ a belief state formula. o_i and ψ in EAFOL 1. if $t = 0$, return ψ. 2. return Move-Quan^a($o_t \wedge \text{FO-STRIPS-Step}(a_t, \text{Move-Quan}(\text{precond}_{a_t} \wedge \text{FOSF}(\langle a_i, o_i \rangle_{0 < i \leq (t-1)}, \psi)))$) ^aMoves all the quantifiers to the front with fresh variable names
PROCEDURE FO-STRIPS-Step(a, ψ) a an action, $\psi = \exists^* \forall^* \bigwedge_i c_i$ a belief-state formula. 1. if $\psi = \exists x \phi(x)$, return $\exists x \text{FO-STRIPS-Step}(a, \phi(L))_{[L/x]}$^a 2. elseif $\psi = \forall x \phi(x)$, return $\forall x \text{FO-STRIPS-Step}(a, \phi(x))$ 3. else, (a) $\psi' \leftarrow$ split every clause in ψ into pure literal clauses (b) $E \leftarrow$ all clauses in ψ' with any literal in $Eff(a)$ (c) $S \leftarrow$ all clauses in ψ' with no literal in $Eff(a)$ (d) if $E \neq \emptyset$ - for all $l \in Eff(a)$ $E \leftarrow \text{resolve-out}(l, E)$ (e) $\phi = \bigwedge_{c_i \in E \cup S} c_i$ (f) $Eff^+(a) \leftarrow$ literals affected to true (g) $Eff^-(a) \leftarrow$ literals affected to false (h) return $\phi \wedge \bigwedge_{p(v) \in Eff^+(a)} p(v) \wedge \bigwedge_{p(v) \in Eff^-(a)} \neg p(v)$. ^aL is a fresh constant that does not appear in the language

Figure 3: First-Order STRIPS Filtering.

a CS book and an ECE book, and our belief state formula is $in(CS-B, CS-R) \wedge in(ECE-B, ECE-R)$. ECE department needs the CS book for a while, so the book keeper moves it to ECE room. The action is $a = move(CS-B, CS-R, ECE-R)$. (This example has unit-case successor state axioms.)

First, we add precondition to belief state formula. The new belief state is $in(CS-B, CS-R) \wedge in(ECE-B, ECE-R) \wedge book(CS-B) \wedge room(CS-R) \wedge room(ECE-R)$. We calculate the filtering of all the single literals of the belief state formula separately and compute the result by using distribution properties. What follows is the formula for one of these literals based on the algorithm presented before.

$$Filter[move(CS-B, CS-R, ECE-R)](in(ECE-B, ECE-R)) \equiv in(CS-B, ECE-R) \wedge \neg in(CS-B, CS-R) \wedge in(ECE-B, ECE-R)$$

Now suppose that instead of applying an action we filter the belief state based on an observation. The robot enters the CS room and observes that there is only one book in the room. The perfect filtering algorithm guarantees that in such cases the book is the same book that the robot has put in the room before.

Assume that the belief state formula is $in(CS-B, CS-R) \wedge in(ECE-B, ECE-R)$. The observation is $\forall x in(x, CS-R) \Rightarrow x = TheBook$. $Filter[o](\psi) \models TheBook = CS-B$, so we can replace every instance of $TheBook$ in the new belief state formula by $CS-B$.

6 Conclusions

In this paper we presented semantics and methodology for filtering in domains that include many objects whose identity is not certain. We generalized this problem to filtering with FOL

formulae. We showed that this problem is solvable in polynomial time when actions map states 1:1, or the actions are STRIPS. We showed that 1:1 actions allow us to filter first-order belief state formulae in linear time (in the size of representation), if we can perform a precompilation step. When actions are STRIPS or Unit-Case, we can filter these belief state formulae efficiently without precompilation. In some cases, we showed that the belief state formulae is guaranteed to remain compactly represented. Those cases permit filtering of actions and observations indefinitely in polynomial time (in the number of predicates and objects). As a result, we can use our algorithm for many interesting applications, such as semantic web, autonomous agents, robot motion control, and partial knowledge games.

7 Acknowledgements

We wish to thank Megan Nance for useful discussions on related topics. We also wish to acknowledge support from DAF Air Force Research Laboratory Award FA8750-04-2-0222 (DARPA REAL program).

References

- [Amir and Russell, 2003] Eyal Amir and Stuart Russell. Logical filtering. In *IJCAI '03*, pages 75–82. MK, 2003.
- [Blass and Gurevich, 2000] A. Blass and Y. Gurevich. Background, Reserve, and Gandy Machines. In P. Clote and H. Schwichtenberg, editors, *Computer Science Logic (Proceedings of CSL 2000)*, volume 1862 of *LNCS*, pages 1–17. Springer, 2000.
- [Boyer and Koller, 1998] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proc. UAI '98*, pages 33–42. MK, 1998.
- [Cravo et al., 2001] Maria R. Cravo, João P. Cachopo, Ana C. Cachopo, and João P. Martins. Permissive belief revision. In *EPIA*, pages 335–348, 2001.
- [Dixon and Wobcke, 1993] Simon Dixon and Wayne Wobcke. The implementation of a first-order logic agm belief revision system. In *ICTAI*, pages 40–47, 1993.
- [Eiter and Gottlob, 1992] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *AIJ*, 57(2-3):227–270, 1992.
- [Lin and Reiter, 1997] Fangzhen Lin and Ray Reiter. How to Progress a Database. *AIJ*, 92(1-2):131–167, 1997.
- [Reiter, 2001] Raymod Reiter. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.
- [Winslett, 1990] Mary-Anne Winslett. *Updating Logical Databases*. Cambridge U. Press, 1990.

A Proofs

A.1 Proof of Theorem 3.1: Filtering Algorithm

PROOF Take $\hat{S} \in \text{Filter}[a](\{s \mid \models_s \psi\})$. We need to show that $\models_{\hat{S}} \text{Filter}[a](\psi)$. From Definition 2.5 there should be S such that $S \in \{s \mid \models_s \psi\}$ and $\langle S, \hat{a}, \hat{S} \rangle \in \mathcal{R}_{\mathcal{D}}$. In other words, there should be S such that $\models_S \psi$ and $\langle S, \hat{a}, \hat{S} \rangle \in \mathcal{R}_{\mathcal{D}}$.

To prove $\models_{\hat{S}} \text{Filter}[a](\psi)$ we need to show that $\psi \wedge \text{precond}_a(u_{1:n}) \wedge \bigwedge_i \forall y_{1:m}, p'_i(y_{1:m}) \Leftrightarrow \text{succ}_{p_i,a}(u_{1:n}, y_{1:m}) \wedge \bigwedge_i \forall y_{1:m} \forall z, f'_i(y_{1:m}) = z \Leftrightarrow \text{succ}_{f_i,a}(u_{1:n}, y_{1:m}, z)$ with $\hat{S}$ as structure for $\mathcal{P}'$ and S as structure for $\mathcal{P}$. In other words the truth assignment $\hat{S}$ to all predicates in some situation satisfies this formula together with the truth assignment S to all predicates in following situation. It is not satisfying this formula only if one of the conjuncts ψ , $\text{precond}_a(u_{1:n})$, $\forall y_{1:m}, p'_i(y_{1:m}) \Leftrightarrow \text{succ}_{p_i,a}(u_{1:n}, y_{1:m})$, or $\forall y_{1:m} \forall z, f'_i(y_{1:m}) = z \Leftrightarrow \text{succ}_{f_i,a}(u_{1:n}, y_{1:m}, z)$ is falsified. This is not the case for first two by our choice of S .

Assume by contradiction that this is the case for some i . Then, the truth assignments sanction that $p'_i(y_{1:m}) \Leftrightarrow \text{succ}_{p_i,a}(u_{1:n}, y_{1:m})$ does not hold. From the way we defined $\mathcal{R}$ this is never the case. This contradicts our assumption. The same argument is true for functions. Thus, we get that $\models_{\hat{S}} \text{Filter}[a](\psi)$. ■

A.2 Proof of Theorem 3.4: Progression Possibility

PROOF We show that the two sets of world structures have the same elements. We first show that the left-hand side of the equality is contained in the right-hand side.

Take $\phi \in \mathcal{D}_2$. We show that ϕ should be in $\mathcal{D}'_2$. We can plug the definition of $\mathcal{D}_1$ in the definition of $\mathcal{D}_2$.

$$\begin{aligned} \mathcal{D}_2 &= C_n(C_n(\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\}) \cap \mathcal{L}(\mathcal{D}_g \cup \{s_1\})) \\ &\quad \cap \mathcal{L}(\mathcal{D}_g \cup \{s_2\}) \\ &\subseteq C_n(C_n(\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\})) \cap \mathcal{L}(\mathcal{D}_g \cup \{s_2\}) \\ &= C_n(\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\}) \cap \mathcal{L}(\mathcal{D}_g \cup \{s_2\}) \end{aligned}$$

In other words, $\mathcal{D}_2 \subseteq \mathcal{D}'_2$.

For the opposite direction (showing the right-hand side is contained in the left-hand side), suppose that $\phi \in \mathcal{D}'_2$. We show that $\phi \in \mathcal{D}_2$.

From the definition of $\mathcal{D}'_2$, we know that $\phi \in \mathcal{D}'_2$ iff $\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0), s_2 = do(a_2, s_1)\} \models \phi$. So, with the same explanation, we show that $\mathcal{D}_1 \cup \{s_2 = do(a_2, s_1)\} \models^? \phi$.

$$\begin{aligned} \phi &\in \mathcal{D}'_2 \\ \mathcal{D}_0 \cup \{s_1 = do(a_1, s_0), s_2 = do(a_2, s_1)\} &\models \phi \\ \mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\} &\models (s_2 = do(a_2, s_1)) \Rightarrow \phi \end{aligned}$$

On the other hand we know that,

$$\begin{aligned} \mathcal{L}(\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\}) &= \mathcal{L}(\mathcal{D}_0 \cup \{s_0, s_1\}) \\ &\subseteq \mathcal{L}(\mathcal{D}_g \cup \{s_0, s_1\}) \\ \mathcal{L}(s_2 = do(a_2, s_1) \Rightarrow \phi) &= \mathcal{L}(\mathcal{D}'_2 \cup \{s_1, s_2\}) \\ &\subseteq \mathcal{L}(\mathcal{D}_g \cup \{s_1, s_2\}) \end{aligned}$$

As we know, there is no s_i in $\mathcal{D}_g$ so we can compute the intersection of the two side of equation.

$$\begin{aligned} \mathcal{L}(\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\}) \cap \mathcal{L}(s_2 = do(a_2, s_1) \Rightarrow \phi) \\ \subseteq \mathcal{L}(\mathcal{D}_g \cup \{s_1\}) \end{aligned}$$

Now, by applying Craig's interpolation theorem for FOL, we get that there should exist a $\gamma \in \mathcal{L}(\mathcal{D}_g \cup \{s_1\})$ such that $\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\} \models \gamma$ and $\gamma \models (s_2 = do(a_2, s_1)) \Rightarrow \phi$. From $\gamma \in \mathcal{L}(\mathcal{D}_g \cup \{s_1\})$ and $\mathcal{D}_0 \cup \{s_1 = do(a_1, s_0)\} \models \gamma$ we can conclude that $\gamma \in \mathcal{D}_1$. So,

$$\mathcal{D}_1 \models (s_2 = do(a_2, s_1)) \Rightarrow \phi$$

$$\mathcal{D}_1 \cup (s_2 = do(a_2, s_1)) \models \phi$$

and the opposite direction is done. ■

Combining Learning Constraints and Numerical Regression

Dorian Šuc^{1,2}, Ivan Bratko²

¹National ICT Australia, Sydney Laboratory at UNSW, NSW 2052, Australia

²Faculty of Computer and Information Science, University of Ljubljana

Tržaška 25, 1000 Ljubljana, Slovenia

{dorian.suc, ivan.bratko}@fri.uni-lj.si

Abstract

Usual numerical learning methods are primarily concerned with finding a good numerical fit to data and often make predictions that do not correspond to qualitative laws in the domain of modelling or expert intuition. In contrast, the idea of Q^2 learning is to induce qualitative constraints from training data, and use the constraints to guide numerical regression. The resulting numerical predictions are consistent with a learned qualitative model which is beneficial in terms of explanation of phenomena in the modelled domain, and can also improve numerical accuracy. This paper proposes a method for combining the learning of qualitative constraints with an arbitrary numerical learner and explores the accuracy and explanation benefits of learning monotonic qualitative constraints in a number of domains. We show that Q^2 learning can correct for errors caused by the bias of the learning algorithm and discuss the potentials of similar hierarchical learning schemes.

1 Introduction

Learning understandable models is one of the main goals of machine learning, but has been recently overshadowed by methods that mainly concentrate on classification or regression accuracy. Less effort has been devoted to improving the explanation strength of machine learning methods. Induced models are often too complex and overly detailed to provide an understandable explanation of phenomena in a modelled domain. This is particularly notable with methods that achieve excellent accuracy by constructing ensembles of classifiers (overview in [Dietterich, 1998]). Another problem, illustrated and discussed in [Šuc *et al.*, 2004], is that state-of-the-art numerical machine learning methods often make predictions that a knowledgeable user finds obviously incorrect – not so much in numerical, but in qualitative terms. Such qualitative errors of numerical predictors are undesirable particularly because they make numerical results difficult to interpret. The underlying mechanism in the domain is usually best explained in qualitative terms. However, this is obscured by qualitative errors in numerical predictions.

Qualitatively faithful quantitative learning, called Q^2 learning for short, was proposed [Šuc *et al.*, 2004] to rectify the qualitative problems of numerical learning. Q^2 learning combines qualitative and numerical learning to give numerical predictions that both fit the data well and are consistent with an induced qualitative model. The qualitative consistency is beneficial in terms of explanation of phenomena in a modelled domain. Quite surprisingly, a case study with Q^2 learning shows that induced qualitative constraints can also improve numerical accuracy. This paper extends the previous work in several directions.

One contribution of this paper is a Q^2 learning scheme that combines learning of monotonic qualitative constraints with an arbitrary numerical learner and enables us to study accuracy benefits of the induced constraints. Qualitative learning has been previously used in a number of applications that are mainly tied to dynamic systems and control. In these applications, advantages in terms of explanation and in terms of the control performance of the induced qualitative models were observed. Since these models define only constraints on a class variable, a direct assessment of their accuracy benefits was previously not possible. The second contribution is an empirical evaluation in a number of domains and a demonstration that such regression, guided by induced qualitative constraints, often increases numerical accuracy.

We analyze the reasons for these accuracy improvements and show that Q^2 learning corrects for errors caused by the bias of a learner. In this respect Q^2 learning is similar to ensembles of classifiers, in particular approaches that combine classifiers constructed by different learning algorithms, e.g. combining instance and model-based learning [Quinlan, 1993] or stacking and its variations [Wolpert, 1992; Gama and Brazdil, 2000; Todorovski and Džeroski, 2003]. An important distinction of Q^2 learning is the (qualitative) *consistency of models at different levels of abstraction*. We discuss advantages of similar hierarchical learning schemes and we demonstrate that explanation improvements do not necessarily come at the price of lower accuracy.

In Section 2 we describe the Q^2 learning scheme proposed in this paper. The elements and the details of Q^2 learning are described in Section 3. Then we give experimental results in various domains and study accuracy improvements using bias-variance decomposition. Section 5 discusses results and benefits of Q^2 learning, and gives directions for future work.

2 Q^2 Learning with QUIN and Qfilter

The idea of Q^2 learning is to combine qualitative and numerical learning to find a regression function that fits the data well and is consistent with an induced qualitative model. In this paper, Q^2 learning consists of two stages:

1. Program *QUIN*¹ (described in Section 3.1) induces a qualitative tree from numerical training examples. Qualitative trees are similar to decision trees, but have monotonic qualitative constraints in their leaves.
2. Algorithm *Qfilter* (described in Section 3.2) uses the induced qualitative tree, plus training examples, plus class predictions of an arbitrary numerical learner, to alter the predictions to respect the induced qualitative tree. *Qfilter* is an optimization procedure that finds the minimal quadratic changes in class values that achieve consistency with the qualitative tree. In our experiments, the numerical learners, also called *base-learners*, are regression trees, model trees and locally weighted regression.

In this paper, a Q^2 learner consists of a qualitative constraints learner and a numerical base-learner, and can be denoted by $Q^2(\text{qual-learner}, \text{base-learner})$. We abbreviate the common case $Q^2(\text{QUIN}, \text{base-learner})$ simply as $Q^2\text{base-learner}$. This learning scheme is particularly interesting because base-learner's predictions are changed optimally in the sense of squared error. Therefore, the differences between base-learner's predictions and Q^2 predictions come just from induced qualitative constraints.

3 Elements of Q^2 Learning

3.1 Monotonic Qualitative Constraints, Qualitative Trees and QUIN

Monotonic qualitative constraints (MQCs) are a kind of monotonicity constraints that are widely used in the field of qualitative reasoning and are a generalization of monotonic function constraint used in QSIM [Kuipers, 1994]. A simple example of an MQC is: $Y = M^+(X)$. This says that Y is monotonically increasing in its dependence on X , i.e. whenever X increases, Y also increases. In general, MQCs can have more than one argument. For example, $Z = M^{+, -}(X, Y)$ says that Z is monotonically increasing in X and monotonically decreasing in Y . If both X and Y increase, then according to this constraint, Z may increase, decrease or stay unchanged. In such a case, an MQC cannot make an unambiguous prediction of the qualitative change in Z .

Qualitative trees are similar to decision trees, but have monotonic qualitative constraints in the leaves. Figure 1 gives an example of a simple qualitative tree. This qualitative tree is a qualitative model of the function $Y = L \sin(\Phi)$, where $0 \leq \Phi \leq \pi$ and $L > 0$. It describes how Y qualitatively depends on attributes Φ and L . The tree partitions the attribute space into two regions that correspond to the two leaves of

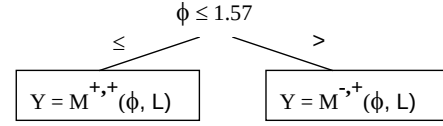


Figure 1: A qualitative tree induced from examples for the function $Y = L \sin(\Phi)$, where $0 \leq \Phi \leq \pi$ and $L > 0$. The right leaf, which applies when $\Phi > \pi/2$, says that Y is monotonically decreasing in Φ and monotonically increasing in L .

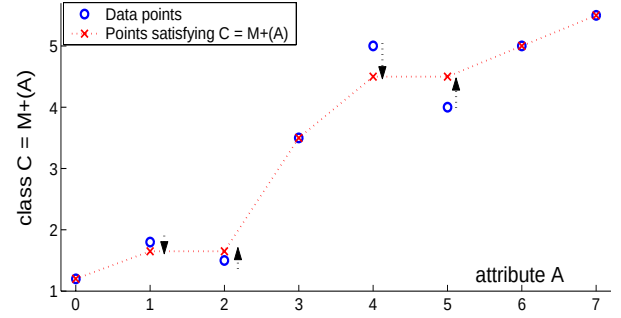


Figure 2: Achieving consistency with MQC $C = M^+(A)$: class values c_i (denoted by circles) are changed into $c_i + d_i$ (denoted by crosses) by minimizing the sum of squared changes d_i . The arrows denote the class changes d_i .

the tree. Note that a simple qualitative tree can describe a relatively complicated nonlinear function. Such qualitative trees are induced from numerical data by learning program QUIN [Šuc, 2003].

QUIN constructs a qualitative tree in a top-down greedy fashion, similar to decision or regression tree learning algorithms. The split selection criterion is based on the minimum description length principle and takes into account the encoding complexity of the subtrees, the consistency of the MQCs in the subtrees with the corresponding data, and the “ambiguity” of the MQCs with respect to the data (the more unambiguous qualitative predictions the MQC can make, the better).

3.2 Qfilter Algorithm

Qfilter handles each leaf of a qualitative tree separately. It first splits the examples according to the qualitative tree and then changes class values to achieve consistency with MQCs in corresponding leaves.

Let us first observe a simple example in Figure 2. We have eight examples (a_i, c_i) , $i=0, 1, \dots, 7$. Class C has values c_i and attribute A has values $a_i=i$. The examples are not consistent with the given MQC $C = M^+(A)$, because the MQC requires that $c_{i+1} > c_i$ which is violated at $i = 1$ and $i = 4$.

To achieve consistency with $C = M^+(A)$, class values should be changed into $c_i + d_i$, where the unknown parameter d_i denotes the change in the i -th class value. Class changes d_i are constrained by MQC-imposed inequalities: $c_{i+1} + d_{i+1} > c_i + d_i$ where $i = 0, 1, \dots, 6$. These inequalities can be formulated in matrix notation as $\mathbf{A}\mathbf{d} > \mathbf{b}$, where $\mathbf{d}$ is a vector of unknown parameters d_i , vector $\mathbf{b}$ has ele-

¹QUIN with a graphical user interface, *Qfilter* and locally weighted regression as a base-learner are available at <http://ai.fri.uni-lj.si/dorian/q2/quin.htm>.

ments $b_i = c_i - c_{i+1}$, and matrix $\mathbf{A}$ has elements $a_{i,i} = -1$, $a_{i,i+1} = 1$ and zeros elsewhere. In general, $\mathbf{b}$ and $\mathbf{A}$ depend on the MQC-imposed inequalities, which in turn depend on the MQC and on the ordering of attributes' values.

Therefore, finding minimal quadratic changes in class values that achieve consistency with a given MQC can be posed as the *quadratic programming optimization problem*: find vector $\mathbf{d}$ that minimizes the criterion function $\mathbf{d}^T \mathbf{H} \mathbf{d}$ such that $\mathbf{A} \mathbf{d} > \mathbf{b}$. In the above formulation matrix $\mathbf{H}$ is the identity matrix. In general $\mathbf{H}$ can be changed to differently penalize the changes in class values as described in Section 3.3.

Since the criterion function with diagonal matrix $\mathbf{H}$ is a convex function, and because the linear constraints $\mathbf{A} \mathbf{d} > \mathbf{b}$ define a convex hull, a local minimum of the criterion function is a globally optimal solution. A more elaborate description of Qfilter, defining also the appropriate ordering of class values when more than one attributes are used in an MQC, is given in [Šuc and Bratko, 2003]. In previous work Qfilter was used with qualitative trees derived manually from domain knowledge. Here we use it in a different and a more challenging context, where qualitative trees are induced from data.

3.3 Qfilter for Q^2 Learning

Qfilter is supplied with a qualitative tree, training examples with their class values and test examples with the base-learner's class predictions. Qfilter then adjusts the class values of the training and the test examples to achieve consistency with the qualitative tree.

One improvement of Qfilter is to use the base-learner's confidence estimates in its predictions. In this case, Qfilter makes smaller adjustments to the class values with higher confidences at the expense of larger changes of class values that have lower confidences. This is achieved by changing the quadratic programming criterion function. Namely, matrix $\mathbf{H}$ is changed from the identity to a diagonal matrix with elements $h_{i,i} = w_i$. Weight w_i is computed from the base-learner's confidence estimate in the i -th class value. Of course, the computation of weight w_i depends on a type and a scale of confidence estimates, but would generally be larger if a numerical predictor is more confident in the i -th class prediction.

Based on this idea, we used a heuristic weighting function: $w_i = 1 + \text{sign}(\mu - c_i)(1 - \exp(-\frac{(c_i - \mu)^2}{2\sigma}))$, where c_i denotes base-learner's confidence in the class prediction of the i -th test example, and μ and σ denote the mean and the standard deviation of base-learner's confidences over all test examples. Therefore, the weight of a test example is between zero and two. The weight of all training examples is set to two. In experiments with locally weighted regression, confidence estimates c_i were set to the sizes of confidence intervals. Model and regression trees do not provide similar confidence estimates. For this reason, confidence estimates of all test examples were set to one.

4 Empirical Evaluation

4.1 Experimental Details

Here we evaluate accuracy benefits of Q^2 learning with various numerical base-learners. Given a set of training exam-

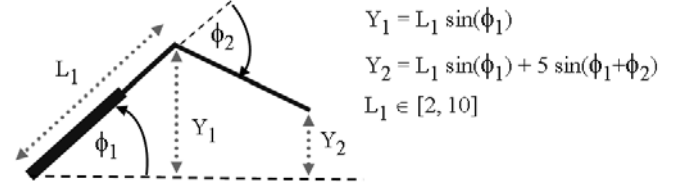


Figure 3: A planar two-link, two-joint robot arm. The first link is extendible with length L_1 ranging from 2 to 10.

ples, a base-learner is used to predict class values of test examples. The same training examples are used for QUIN to induce a qualitative tree. The qualitative tree, the training and the test examples with base-learner's class predictions are then used by Qfilter to give predictions that are consistent with the qualitative tree. This procedure is then repeated (for example, ten times with ten-fold cross-validation) with different training and test examples.

In the experiments we compare *root relative squared errors* (*RREs* for short) of the base-learner and the Q^2 learner. Here, the *RRE* is the root mean squared error normalized by the root mean squared error of average class value. The base-learners in our study are regression and model trees [Breiman *et al.*, 1984; Quinlan, 1993], and locally weighted regression [Atkeson *et al.*, 1997]. The first two base-learners were chosen because they are well-established numerical learners that provide a symbolical model. Locally weighted regression (LWR for short) does not provide a model explaining a studied domain, but often gives more accurate predictions than model or regression trees. With such base-learners, the explanation benefits of Q^2 learning are even more obvious. In experiments we used our implementations of these base-learners. Our regression and model trees use cost-complexity pruning [Breiman *et al.*, 1984] and smoothing. LWR uses Gaussian weighting function with local optimization to set the kernel size at each prediction point. We also give *RREs* of M5 model trees [Quinlan, 1993] and its Weka implementation called M5Prime [Witten and Wang, 1997] to show that we are not comparing Q^2 to base-learners that perform poorly. With all learners, default values of their parameters were used.

We analyze the reasons for the Q^2 accuracy improvements using bias-variance decomposition [Geman *et al.*, 1992] and draw some interesting conclusions.

4.2 Robot Arm Domain

Here we describe experiments in the domain of a planar two-link, two-joint robot arm depicted in Figure 3. The angle in the shoulder joint is denoted by Φ_1 and the angle in the elbow joint is denoted by Φ_2 . Angle Φ_1 is between zero and π , while Φ_2 is between $-\pi/2$ and $\pi/2$. The first link, i.e. the link from the shoulder to the elbow joint, is extendible with length L_1 ranging from 2 to 10. The second link has fixed length 5. Y -coordinates of the first and the second link ends are denoted by Y_1 and Y_2 respectively. We experimented with four learning problems that differ in class variable (Y_1 or Y_2) and the attributes used for learning. These learning problems were defined to pose increasingly more difficult problems to Q^2 learning.

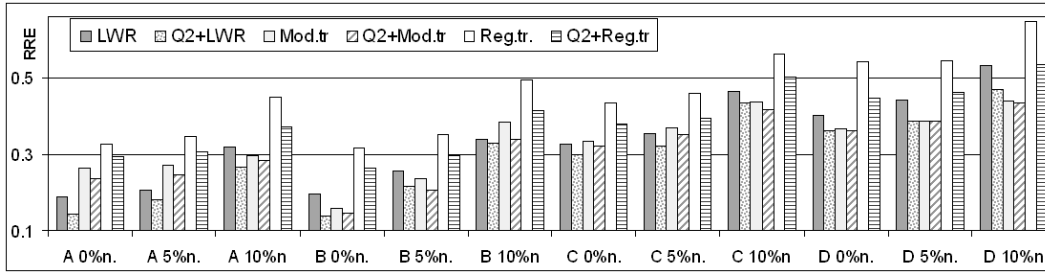


Figure 4: Comparing RREs of base-learners and Q^2 learners in learning problems A, B, C and D with 0%, 5% and 10% noise.

Table 1: RREs of LWR, model (Mt) and regression trees (Rt) and the corresponding Q^2 learners with no noise. The last column gives RREs of M5Prime.

	LWR	Q^2 LWR	Mt	Q^2 Mt	Rt	Q^2 Rt	M5Pr
A	0.187	0.145	0.264	0.238	0.327	0.293	0.512
B	0.196	0.139	0.159	0.146	0.318	0.263	0.462
C	0.327	0.300	0.336	0.324	0.434	0.378	0.489
D	0.403	0.362	0.366	0.363	0.542	0.448	0.553

The easiest learning problem, called problem A, is to predict the y -coordinate of the first link end given its length and angle, i.e. learning $Y_1 = f(L_1, \Phi_1)$. Other learning problems require predicting the y -coordinate of the second link end using different attributes. In learning problems B and C we helped the learners with a derived attribute $\Phi_{sum} = \Phi_1 + \Phi_2$, i.e. the deflection of the second link from the horizontal. Problem C is to learn $Y_2 = f(L_1, \Phi_1, \Phi_2, \Phi_{sum})$, while in problem B we also used Y_1 as an attribute. Problem D requires learning $Y_2 = f(L_1, \Phi_1, \Phi_2)$. These problems pose increasingly more difficult problems to Q^2 learning. The easiest is problem A, because a correct qualitative model is a simple qualitative tree, given also in Figure 1. Learning problem D is the most difficult for Q^2 , since a correct qualitative model cannot be expressed by a qualitative tree.

To compare accuracy of different base-learners and Q^2 learning we generated examples where angles Φ_1 and Φ_2 and link length L_1 were randomly generated from a uniform distribution. We experimented with different percentages of Gaussian noise in the class variable. Noise percentage $p\%$ means that the standard deviation of noise is $p \times d_c/100$, where d_c denotes the difference between maximal and minimal class value. We used 100 training examples and measured accuracy on separate test sets of 200 examples without noise. All results are averages on 20 randomly generated training and test sets.

Table 1 gives RREs of LWR, model and regression trees and Q^2 learning that uses these base-learners. For comparison RREs of M5Prime model trees are also given. Results with zero, 5%, and 10% noise are given in Figure 4. In all of the learning problems and with all noise levels, Q^2 improves the average RREs of all base-learners. These improvements in accuracy depend on the base-learner and the learning problem. Generally, the improvements are the greatest with regression trees and the smallest with model trees. It is notable

that in general our base-learners are much more accurate than M5Prime on these learning problems. Although we are here not interested in comparison of different base-learners, it is good to know that with Q^2 we are not improving the base-learners that perform poorly.

The significance of the Q^2 accuracy improvements for each learning problem was tested using the resampled paired t test. Four learning problems $\times$ three base-learners $\times$ three noise levels gives 36 comparisons of RRE of a base-learner and a corresponding Q^2 learner. At 5% significance level, the Q^2 learners are significantly better in 33 comparisons and about the same in only three comparisons. The comparisons where differences in RRE are not significant correspond to model trees on learning problem D with all three noise levels.

Bias-Variance Decomposition

To understand the reasons for the Q^2 accuracy improvements we used bias-variance decomposition [Geman *et al.*, 1992; Domingos, 2000], which has proved to be a very useful tool for understanding machine learning algorithms. Bias-variance decomposition in regression states that the expected squared error of a learner on test example x is the sum of the irreducible noise $N(x)$, the bias $B(x)$ and the variance $V(x)$. The bias $B(x)$ of the learner on an example x is the squared difference between the true value and the mean prediction on x over all possible training sets. Here defined bias $B(x)$ is in the literature called also squared bias, but we use the notation from [Domingos, 2000] and call it bias. The variance $V(x)$ is the expected value of the squared difference between the true value and the mean prediction on x . The bias measures the systematic error incurred by a learner, and the variance measures the error incurred by its fluctuations around the central tendency in response to different training sets. Irreducible noise is the error of the optimal (Bayes) model and is in general very difficult to estimate. However, we are here using an artificial learning problem and can therefore measure the bias and variance by directly simulating the definitions.

We used 100 training sets of size 100, generated as in the previous experiment and measured the average bias and average variance on a test set of 810 equidistant data points. We refer to these averages over different training sets and different test examples simply as bias and variance. Comparing the base-learners we noticed that regression trees have the highest variance. This is in accordance with their complexity – they usually had more than 20 leaves. Model trees are smaller and have the smallest variance when no noise, but with increas-

Table 2: Description of data sets and average 10-fold cross-validation root relative squared errors ($RREs$) of base-learners and the corresponding Q^2 learners. The last column gives $RREs$ of M5 where this is available, and M5Prime otherwise.

Data set	Cases	Attributes	LWR	Q^2 LWR	Mod.tr.	Q^2 Mod.tr.	Reg.tr.	Q^2 Reg.tr.	M5
AutoMpg	398	5/8	0.361	0.380	0.374	0.386	0.459	0.405	0.383
AutoPrice	159	16/16	0.380	0.346	0.523	0.364	0.435	0.381	0.402
Housing	506	12/13	0.373	0.363	0.422	0.358	0.532	0.454	0.431
MachineCpu	209	6/6	0.353	0.317	0.370	0.334	0.460	0.377	0.414
Servo	167	2/4	0.478	0.457	0.588	0.584	0.508	0.504	0.536
CraneSkill1	354	3/3	0.142	0.065	0.194	0.086	0.091	0.049	0.247
CraneSkill2	618	3/3	0.249	0.169	0.190	0.159	0.191	0.165	0.215
AntiSway	200	4/4	0.414	0.157	0.319	0.250	0.193	0.165	0.412

ing noise the variance increases most notably. Comparing Q^2 learners to the corresponding base-learners we noticed that Q^2 always notably reduces the bias of the base-learners. This happens in all four learning problems, with all three base-learners and different noise levels. For example, in problem D with no noise, Q^2 reduces bias and variance of LWR from 3.85 and 2.57 to 2.13 and 1.99, respectively. Q^2 always notably decreases the variance of regression trees, but sometimes increases the variance of LWR and model trees. The base-learners, used here, have a bias towards linear models. It seems that Q^2 , with less restrictive monotonicity constraints, reduces their bias, but does not considerably increase the variance.

Q^2 learning combines hypotheses of two different learning algorithms, i.e. a qualitative learner and a numerical learner. In this respect it is similar to ensembles of classifiers and in particular approaches that combine classifiers constructed by different learning algorithms, as for example stacking and its variations. Such methods improve accuracy mainly by reducing the error due to the bias of a learner. This is the consequence of combining hypotheses that make uncorrelated errors [Dietterich, 1998]. We believe that such uncorrelated errors lead to bias reductions also in the case of Q^2 learning. It should be noted that, although Q^2 predictions are consistent with an induced qualitative model, they combine both the base-learner's predictions, and the qualitative model.

4.3 UCI and Dynamic Domains

To explore the potentials of learning similar constraints with a wider range of learning problems we describe experiments with eight data sets. The first five are the smallest regression data sets from the UCI repository [Blake and Merz, 1998] with the majority of continuous attributes. A reason for choosing these data sets is also that Quinlan [1993] gives results of M5 and several other regression methods on these data sets, which enables a better comparison of Q^2 to other methods. These data sets are *AutoMpg*, *AutoPrice*, *Housing*, *MachineCpu* and *Servo*.

The other three data sets are from dynamic domains where QUIN has typically been applied so far [Šuc, 2003; Šuc and Bratko, 2002]. It should be noted that in these domains the primary objective was to explain the underlying control skill and to use the induced qualitative models to control a dynamic system. Until now, it was not possible to measure their numerical accuracy or compare it to other learning methods.

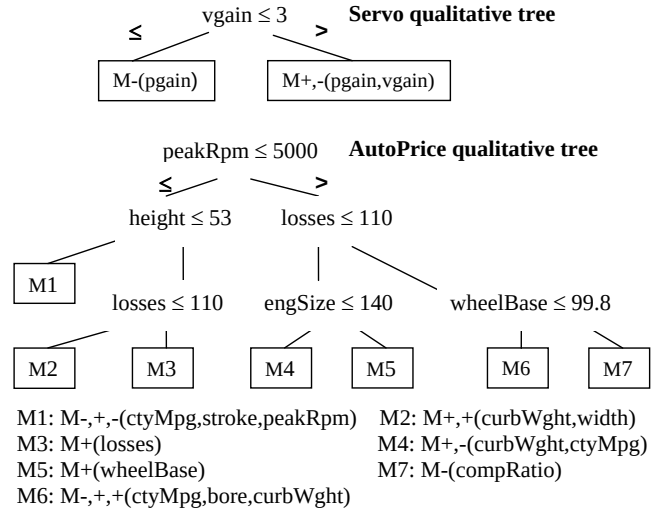


Figure 5: Qualitative trees induced from data sets *Servo* and *AutoPrice*. MQCs in leaves of each qualitative tree give monotonic constraints on the class variable.

Data sets *CraneSkill1* and *CraneSkill2* are the logged data of two experienced human operators controlling a crane simulator. Such control traces are typically used to reconstruct the underlying operator's control skill. The learning task is to predict the velocity of a crane trolley given the position of the trolley, rope angle and its velocity. Data set *AntiSway* was used in reverse-engineering an industrial gantry crane controller. This so-called *anti-sway crane* is used in metallurgical companies to reduce the swing of the load and increase the productivity of transportation of slabs. The learning task is to learn the control force applied to the trolley, given the desired and the current trolley velocity and the position and velocity of the load relative to the trolley.

In all experiments, qualitative trees were considerably simpler than other induced models. Figure 5 gives examples of qualitative trees induced in data sets *Servo* and *AutoPrice*. These qualitative trees are considerably simpler than model trees. In *Servo*, M5Prime induces a model tree with eleven leaves, with all four attributes appearing in all eleven linear models. In *AutoPrice*, M5Prime induces a model tree with ten leaves, with at least ten attributes in each linear model.

Ten-fold cross-validation results are given Table 2. For

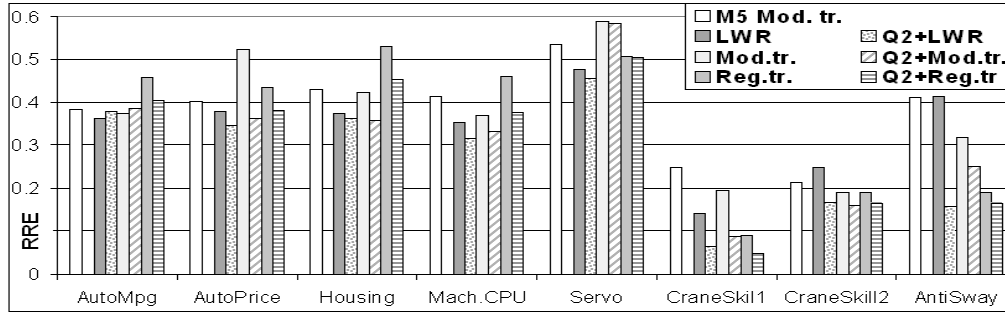


Figure 6: Comparison of RRE s of base-learners and corresponding Q^2 learners. Disjoint columns give RRE s of M5 model trees where available, and M5Prime otherwise.

each data set we give, respectively, the numbers of cases, numbers of continuous and all attributes, RRE s of base-learners and RRE s of corresponding Q^2 learners. The last column gives published RRE s of M5 model trees [Quinlan, 1993] on UCI data sets, and RRE s of M5Prime for others. These results are presented also in Figure 6.

A general observation is that Q^2 improves RRE s of all three base-learners in seven out of eight data sets. Q^2 is worse just in *AutoMpg* with LWR and model trees. Smaller root relative squared errors in the last three data sets do not imply that these learning problems are easier, but are just the consequence of the normalization of RRE s with large class variances.

The significance of the Q^2 accuracy improvements was tested using the 10-fold cross-validated paired t test. Eight data sets $\times$ three base-learners gives 24 comparisons of RRE of a base-learner and the corresponding Q^2 learner. At 5% significance level, the Q^2 learners are significantly better in 18 comparisons and about the same in six comparisons. The differences are not significant in *AutoMpg* with all three base-learners and with one base-learner in *AutoPrice*, *Housing* and *CraneSkill2*. Q^2 is never significantly worse.

5 Discussion and Conclusions

The goal of this paper is to explore the accuracy and explanation benefits of Q^2 learning that were observed also in previous work [Šuc *et al.*, 2004]. We extend the empirical evaluation to a number of domains and analyze the results. The proposed Q^2 learning scheme, makes it possible to combine learning of qualitative constraints with an arbitrary numerical learner. It uses algorithm Qfilter, which is particularly interesting since the base-learner predictions are optimally changed (in the sense of squared error) to be consistent with an induced qualitative tree. Therefore, the accuracy improvements of Q^2 with respect to a base-learner are only due to the induced qualitative trees.

Q^2 learning as a hierarchical learning scheme

Q^2 learning, as presented in this paper, can be seen as a hierarchical learning scheme, where a learner at a higher level induces a hypothesis h_n that guides the learner at a lower level. At higher levels of the scheme more abstract concepts (or more general constraints) are learned. In this way, h_n pro-

vides inductive bias for learning of hypothesis h_{n-1} . In the case of Q^2 learning, h_1 is a qualitative tree induced by QUIN and h_0 are numerical predictions found by Qfilter. Similar hierarchical learning schemes were proposed either to improve the generalization of a single learning task, for example Stacked Generalization [Wolpert, 1992] and Cascade Generalization [Gama and Brazdil, 2000], or to facilitate the learning of several tasks in a hierarchy [Stone and Veloso, 2000]. Q^2 learning has two distinctive properties. First, the hypotheses in the hierarchy are consistent and described at different levels of abstraction. This is important for the explanation of the phenomena in the modelled domain. Second, the hypotheses are learned in the general-to-specific order, which can reduce the search space, and can consequently improve also the generalization properties.

Explanation and Accuracy Benefits of Q^2 Learning

Q^2 predictions are consistent with a qualitative model that provides an explanation at a higher level of abstraction. In this respect Q^2 learning is different than other methods for combining classifiers. Qualitative consistency enables Q^2 to improve numerical accuracy by combining hypotheses induced by different learners, but retain a simple explanation.

Benefits of qualitative learning in terms of providing simple and understandable explanations in various domains, including *CraneSkill1*, *CraneSkill2* and *AntiSway* are discussed in [Šuc, 2003; Šuc and Bratko, 2002]. For example, qualitative trees induced in *CraneSkill1* and *CraneSkill2* have only a few leaves and were, because of their simplicity, preferred over the previous approaches for skill reconstruction, which typically learned regression or model trees with more than twenty leaves. Although the induced qualitative trees are simple, they reveal some surprising and nontrivial aspect of the human skill. Simple and understandable models were induced also in other domains studied in this paper. For example, in robot arm domain (problems A, B and C) QUIN usually induced qualitative trees that are very close to the correct qualitative models even with high percentage of noise. Simple qualitative trees were induced also from UCI data sets (see Figure 5).

In the presented experiments, Q^2 typically improved accuracy of the three base-learners. We compared root relative squared errors, but similar improvements were observed also with mean absolute errors. We experimented also with

k -nearest-neighbor algorithm as a base-learner. It generally performed worse than the other three base-learners and the accuracy improvements of Q^2 learning were even more obvious.

Bias-variance decomposition in the robot arm domain shows that the accuracy improvements stem mainly from correcting for errors caused by the bias of a base-learner. Experiments in the previous section suggest that combining “monotonic regularities” inductive bias with inductive bias of other learners is beneficial in a wide range of domains, also in domains when this might be less expected. As noted in Section 4.2, Q^2 learning is similar to ensembles of classifiers, in particular approaches that combine classifiers constructed by different learning algorithms. Although, the accuracy improvements of Q^2 are not comparable to those achieved by ensembles of classifiers, Q^2 has advantages in terms of explanation.

Limitations and Future Work

Limitations of Q^2 learning presented in this paper are mainly tied to the current implementation of QUIN. Because of its complexity it is difficult to apply it to very large data sets. Incorporating sampling techniques with qualitative learning might be beneficial. Another idea for future work is to assess and use the quality of the induced MQCs in leaves, for example on a separate set of examples. The error of a Q^2 learner could be used to prune a qualitative tree. This might be useful also in data sets such as *AutoMpg*, where Q^2 otherwise has problems. By considering only leaves where MQCs significantly improves a base-learner, a qualitative tree could be transformed into a set of qualitative association rules.

An interesting direction for future work is to explore the possibilities of the proposed hierarchical learning scheme with several layers of constraints, describing hypotheses at different levels of abstraction. Experimental results with Q^2 learning suggest that such scheme can be used to combine classifiers and improve accuracy, and, at the same time provide an understandable explanation of the phenomena in a modelled domain.

Acknowledgements

The work reported in this paper was supported by National ICT Australia and the Slovenian Ministry of Education, Science and Sport. National ICT Australia is founded by the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.

References

- [Atkeson *et al.*, 1997] C.G. Atkeson, A.W. Moore, and S.A. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
- [Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [Breiman *et al.*, 1984] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, California, 1984.
- [Dietterich, 1998] T.G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
- [Domingos, 2000] Pedro Domingos. A unified bias-variance decomposition and its applications. In *Proc. 17th International Conf. on Machine Learning*, pages 231–238. Morgan Kaufmann, San Francisco, CA, 2000.
- [Gama and Brazdil, 2000] J. Gama and P. Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.
- [Geman *et al.*, 1992] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- [Kuipers, 1994] B. Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, Massachusetts, 1994.
- [Quinlan, 1993] J.R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the 12th International Conf. on Machine Learning*, pages 236–243, San Mateo, CA, 1993. Morgan Kaufmann.
- [Stone and Veloso, 2000] P. Stone and M. Veloso. Layered learning. In *Machine Learning: ECML 2000 (Proceedings of the 11th European Conference on Machine Learning)*, pages 369–381. Springer Verlag, 2000.
- [Todorovski and Džeroski, 2003] L. Todorovski and S. Džeroski. Combining classifiers with meta decision trees. *Machine Learning*, 50(3):223–249, 2003.
- [Šuc and Bratko, 2002] D. Šuc and I. Bratko. Qualitative reverse engineering. In *Proceedings of the 19th International Conf. on Machine Learning*, pages 610–617. Morgan Kaufmann, 2002.
- [Šuc and Bratko, 2003] D. Šuc and I. Bratko. Improving numerical accuracy with qualitative constraints. In *Proceedings of the 14th European Conference on Machine Learning*, pages 385–396. Springer, 2003.
- [Šuc *et al.*, 2004] D. Šuc, D. Vladušić, and I. Bratko. Qualitatively faithful quantitative prediction. *Artificial Intelligence*, 158(2):190–219, 2004.
- [Šuc, 2003] D. Šuc. *Machine Reconstruction of Human Control Strategies*, volume 99 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, The Netherlands, 2003.
- [Witten and Wang, 1997] I.H. Witten and Y. Wang. Induction of model trees for predicting continuous classes. In *Proc. Poster Papers Europ. Conf. Machine Learning*, 1997.
- [Wolpert, 1992] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

On the Interaction between Inverse Features and Path-functional Dependencies in Description Logics *

David Toman and Grant Weddell

School of Computer Science, University of Waterloo
{david, gweddell}@uwaterloo.ca

Abstract

We investigate how inverse features can be added to a boolean complete description logic with path-functional dependencies in ways that avoid undecidability of the associated logical implication problem. In particular, we present two conditions that ensure the problem remains EXPTIME-complete. The first is syntactic in nature and limits the form that dependencies may have in argument terminologies. The second is a coherence condition on terminologies that is sufficiently weak to allow the transfer of relational and emerging object-oriented normalization techniques.

1 Introduction

For many applications, there is considerable incentive to enhance the modeling utility of feature based description logics (DLs) with an ability to capture richer varieties of uniqueness constraints such as keys and functional dependencies [DeHaan *et al.*, 2003; Khizder *et al.*, 2000; Stanchev and Weddell, 2003; Toman and Weddell, 2004b]. Unfortunately, in combination with feature or role inversion, the associated logical implication problem quickly becomes undecidable [Calvanese *et al.*, 2001]. We investigate conditions under which inverse features *can* be added to a boolean complete DL with *path-functional dependencies* (PFDs) [Weddell, 1989] without any consequent impact on the complexity of the associated logical implication problem.

Two conditions are presented that ensure this problem remains EXPTIME-complete. The first is syntactic in nature and imposes a prefix condition on PFDs that occur in argument terminologies. The condition complements and extends the results in [Calvanese *et al.*, 2001] which considered the problem of adding keys and functional dependencies to a DL with predicates in place of roles. The second is a coherence condition on terminologies that allows unrestricted use of PFDs, and is sufficiently weak to allow the formal specification of arbitrary relational or object-oriented schema, including those that fail to satisfy normalization conditions.

*The support of Nortel Ltd. and of the Canadian governments through NSERC and CITO is gratefully acknowledged.

This latter observation is important since it enables an incremental development of terminologies that encode schema. One can begin, for example, with a “relational” terminology that fails to satisfy the conditions of Boyce-Codd Normal Form. (The approach used in [Calvanese *et al.*, 2001] is not generally capable of handling such anomalous cases.) Standard normalization algorithms and methodology can then employ reasoning services based on our results. Thus, our DL together with coherence is better equipped to enable the transfer of results in normalization and emerging object design theory for relational and object-oriented data models [Biskup *et al.*, 1996; Biskup and Polle, 2000].

1.1 Related Work

Our coherence condition derives from a similar condition proposed in [Biskup and Polle, 2003] to enable the development of a sound and complete axiomatization for an object-oriented data model, which essentially adds inclusion dependencies to a data model defined earlier in [Weddell, 1989]. The DL we consider in this paper is a further generalization; thus, our EXPTIME-completeness result *settles an open problem on the decidability of the implication problem for their model*.

In [Calvanese *et al.*, 2001], the authors consider a DL with (relational) functional dependencies together with a general form of keys called *identification constraints*. They show that their dialect is undecidable in the general case, but becomes decidable when unary functional dependencies are disallowed. We show undecidability in a simpler setting, in particular without the use of number restrictions. Our prefix condition on PFDs complements and extends their decidability result to more general PFDs, and our coherency condition serves as an alternative method for regaining decidability.

A form of key dependency with left-hand-side feature paths is considered for a DL coupled with various concrete domains [Lutz *et al.*, 2003; Lutz and Milicic, 2004]. The authors explore how the complexity of satisfaction is influenced by the selection of a concrete domain together with various syntactic restrictions on the key dependencies themselves. In contrast, we consider a DL that admits more general kinds of key constraints (and functional dependencies) for which identifying values can be defined on arbitrary domains.

The remainder of the paper is organized as follows. We begin by introducing the DL dialect $\mathcal{DLFAD}$ that will be the focus of the remainder of the paper. The dialect is *feature based* and therefore more functional in style as opposed to the more common *role based* derivatives of $\mathcal{ALC}$. As a consequence, it becomes straightforward to incorporate dependencies into the logic for capturing PFDs. In Section 3, we show that the combination of inverse features and arbitrary PFDs in $\mathcal{DLFAD}$ leads to the undecidability of its associated logical implication problem. Our main results then follow in Section 4 in which we consider two ways to recover decidability based on a *prefix restriction* condition for PFDs in argument terminologies and on a coherency condition for terminologies. Our summary comments follow in Section 5.

2 Preliminaries

Definition 1 (Description Logic $\mathcal{DLFAD}$) Let F and C be sets of feature names and primitive concept names, respectively. A path expression is defined by the grammar “ $\text{Pf} ::= f, \text{Pf} \mid \text{Id}$ ” for $f \in F$. We define derived concept descriptions by a second grammar on the left-hand-side of Figure 1. A concept description obtained by using the final production of this grammar is called a path-functional dependency (PFD).

An inclusion dependency $\mathcal{C}$ is an expression of the form $D \sqsubseteq E$. A terminology $\mathcal{T}$ consists of a finite set of inclusion dependencies.

SYNTAX	SEMANTICS: “ $(\cdot)^{\mathcal{I}}$ ”
$D ::= C$	$(C)^{\mathcal{I}} \subseteq \Delta$
$D_1 \sqcap D_2$	$(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$
$\neg D$	$\Delta \setminus (D)^{\mathcal{I}}$
$\forall f.D$	$\{x : (f)^{\mathcal{I}}(x) \in (D)^{\mathcal{I}}\}$
$\exists f^{-1}.D$	$\{(f)^{\mathcal{I}}(x) : x \in (D)^{\mathcal{I}}\}$
$E ::= D$	
$D : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}$	$\{x : \forall y \in (D)^{\mathcal{I}}. \bigwedge_{i=1}^k (\text{Pf}_i)^{\mathcal{I}}(x) = (\text{Pf}_i)^{\mathcal{I}}(y) \Rightarrow (\text{Pf})^{\mathcal{I}}(x) = (\text{Pf})^{\mathcal{I}}(y)\}$

Figure 1: SYNTAX AND SEMANTICS OF $\mathcal{DLFAD}$.

The semantics of expressions is defined with respect to a structure $(\Delta, \cdot^{\mathcal{I}})$, where Δ is a domain of “objects” and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concepts C to be subsets of Δ and primitive features f to be total functions $(f)^{\mathcal{I}} : \Delta \rightarrow \Delta$. The interpretation is extended to path expressions, $(\text{Id})^{\mathcal{I}} = \lambda x.x$, $(f, \text{Pf})^{\mathcal{I}} = (\text{Pf})^{\mathcal{I}} \circ (f)^{\mathcal{I}}$ and derived concept descriptions D and E as defined on the right-hand-side of Figure 1.

An interpretation satisfies an inclusion dependency $D \sqsubseteq E$ if $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$. The logical implication problem asks if $\mathcal{T} \models D \sqsubseteq E$ holds; that is, if $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$ for all interpretations that satisfy all constraints in $\mathcal{T}$.

For the remainder of the paper, we use the following abbreviated notation: $\forall \text{Pf}.D$ is shorthand for $\forall f_1.\forall f_2.\dots.\forall f_k.D$, and $\exists \text{Pf}^{-1}.D$ for $\exists f_k^{-1}.\dots.\exists f_2^{-1}.\exists f_1^{-1}.D$ for $\text{Pf} = f_1.f_2.\dots.f_k.\text{Id}$. We also identify single feature names f with path descriptions $f.\text{Id}$ and allow concatenation of path descriptions, $\text{Pf Pf}'$ to denote their composition $\text{Pf} \circ \text{Pf}'$.

In addition, we classify constraints by the description on their right-hand side as PFDs, when the right-hand side is of the form $D : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}$, and as *simple constraints* otherwise.

3 Undecidability of $\mathcal{DLFAD}$ Implication

We show a reduction of the unrestricted tiling problem to the $\mathcal{DLFAD}$ implication problem using a construction similar to that presented in [Calvanese et al., 2001]. An unrestricted tiling problem U is a triple (T, H, V) where T is a finite set of tile types and $H, V \subseteq T \times T$ two binary relations. A solution to T is a mapping $t : \mathbb{N} \times \mathbb{N} \rightarrow T$ such that $(t(i, j), t(i + 1, j)) \in H$ and $(t(i, j), t(i, j + 1)) \in V$ for all $i \in \mathbb{N}$. This problem is Π_0^0 -complete [Berger, 1966; van Emde Boas, 1997].

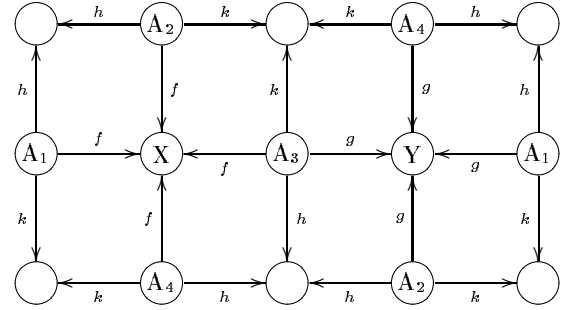


Figure 2: DEFINING A GRID.

The first step in the reduction is to establish an *integer grid*. This can be achieved, for example, as follows.

1. Introduce four disjoint concepts, A_1, A_2, A_3 , and A_4 , denoting cell edges.

$$A_i \sqcap A_j \sqsubseteq \perp \quad \text{for } 0 < i < j \leq 4,$$

2. Grid cells are mapped to concepts X and Y that have four incoming f and g features, respectively.

$$X \sqsubseteq \bigcap_{0 < i \leq 4} \exists f^{-1}.A_i \quad Y \sqsubseteq \bigcap_{0 < i \leq 4} \exists g^{-1}.A_i$$

3. To ensure that squares are formed, add the following.

$$\begin{aligned} A_1 &\sqsubseteq A_2 : f \rightarrow h \sqcap A_2 : h \rightarrow f \sqcap A_2 : g \rightarrow k \sqcap A_2 : k \rightarrow g \\ A_2 &\sqsubseteq A_3 : f \rightarrow k \sqcap A_3 : k \rightarrow f \sqcap A_3 : g \rightarrow h \sqcap A_3 : h \rightarrow g \\ A_3 &\sqsubseteq A_4 : f \rightarrow h \sqcap A_4 : h \rightarrow f \sqcap A_4 : g \rightarrow k \sqcap A_4 : k \rightarrow g \\ A_4 &\sqsubseteq A_1 : k \rightarrow h \sqcap A_1 : h \rightarrow k \sqcap A_1 : g \rightarrow f \sqcap A_1 : f \rightarrow g \end{aligned}$$

4. And force squares to extend to the right and up by including the following.

$$A_1 \sqsubseteq \forall f.X, \quad A_3 \sqsubseteq \forall g.Y, \quad A_2 \sqsubseteq \forall g.Y, \quad A_4 \sqsubseteq \forall f.X$$

The accumulated effect of these inclusion dependencies on an interpretation is illustrated in Figure 2.

The adjacency rules for the instance U of the tiling problem can now be captured, e.g., as follows:

$$\begin{aligned} A_1 \sqcap \forall g.T_i &\sqsubseteq \forall f. \bigsqcup_{(t_i, t_j) \in H} T_j, & A_2 \sqcap \forall f.T_i &\sqsubseteq \forall g. \bigsqcup_{(t_i, t_j) \in V} T_j, \\ A_3 \sqcap \forall f.T_i &\sqsubseteq \forall g. \bigsqcup_{(t_i, t_j) \in H} T_j, & A_4 \sqcap \forall g.T_i &\sqsubseteq \forall f. \bigsqcup_{(t_i, t_j) \in V} T_j \end{aligned}$$

where T_i corresponds to a tile type $t_i \in T$; we assume $T_i \sqcap T_j \sqsubseteq \perp$ for all $i < j$. The above constraints form a terminology $\mathcal{T}_U$ associated with an unrestricted tiling problem U .

Theorem 2 *A tiling problem U admits a solution iff $\mathcal{T}_U \not\models X \sqcap (\bigsqcup_{t_i \in T} T_i) \sqsubseteq \perp$.*

Thus, the $\mathcal{DLFAD}$ implication problem is undecidable for unrestricted terminologies.

4 On Recovering Decidability

In this section, we present two ways to recover decidability for the $\mathcal{DLFAD}$ logical implication problem based on a syntactic prefix condition for PFDs occurring in argument terminologies and on a coherency condition for terminologies. Decidability in both cases is shown by exhibiting a reduction of logical implication problems in $\mathcal{DLFAD}$ to decidability problems of monadic sentences in the Ackermann prefix class.

Definition 3 (Monadic Ackerman Formulae) *Let P_i be monadic predicate symbols and x, y_i, z_i variables. A monadic first-order formula in the Ackermann class is a formula of the form $\exists z_1 \dots \exists z_k \forall x \exists y_1 \dots \exists y_l. \varphi$ where φ is a quantifier-free formula over the symbols P_i .*

Every formula with the Ackermann prefix can be converted to Skolem normal form: by replacing variables z_i by Skolem constants and y_i by unary Skolem functions not appearing in the original formula. This, together with standard boolean equivalences, yields a finite set of universally-quantified clauses containing at most one variable (x).

It is known that an Ackerman sentence has a model if and only if it has a Herbrand model; this allows us to use syntactic techniques for model construction. To establish the complexity bounds we use the following result for the satisfiability of Ackermann formulae:

Proposition 4 (Fürier, 1981) *The complexity of the satisfiability problem for Ackerman formulae is EXPTIME-complete.*

4.1 Prefix-restricted PFDs

Recall that the first of our conditions is syntactic and applies to the argument PFDs occurring in a terminology. This condition complements and extends the results in [Calvanese *et al.*, 2001]. However, because of “accidental common prefixes”,

it is not sufficient to follow the approach in [Calvanese *et al.*, 2001] of simply requiring unary PFDs to resemble keys. Non-unary PFDs can also cause trouble, as the following example illustrates.

Example 5 Consider the k -ary PFD

$$A_1 \sqsubseteq A_2 : f.a_1, \dots, f.a_k \rightarrow h.$$

This PFD has a logical consequence $A_1 \sqsubseteq A_2 : f \rightarrow h$ and thus we can construct tiling similar to the one presented in the previous section.

The problem with the above k -ary PFD is that all the k paths in the precondition have the same prefix, f . To avoid this problem, it suffices to impose the following prefix condition.

Definition 6 (Prefix-restricted PFDs) *Let*

$$D : \text{Pf Pf}_1, \dots, \text{Pf Pf}_k \rightarrow \text{Pf}'$$

be an arbitrary PFD where Pf is the maximal common prefix of the path expressions $\{\text{Pf Pf}_1, \dots, \text{Pf Pf}_k\}$. The PFD is prefix-restricted if either Pf' is a prefix of Pf or Pf is a prefix of Pf' .

The above restriction allows us to recover an *almost* tree-model property for the logic. By using this restriction, we are able to construct special interpretations that satisfies all constraints in a given terminology but falsify a PFD constraint whenever an interpretation exists at all—essentially, the interpretation has the shape of two trees rooted by the two elements of the domain that provide a counterexample to the given PFD. To capture the effect of equalities implied by PFDs, the two trees are allowed to *share* nodes that are in the range of the same path function applied to the respective roots. In addition, we are able to bound the indegree branching in such an interpretation.

Definition 7 (Rank of Implication Problem) *Let $\mathcal{T}$ be a $\mathcal{DLFAD}$ terminology, $\mathcal{C}$ a constraint. We define $\text{Rank}(\mathcal{T}, \mathcal{C})$ to be the number of occurrences of the $\exists f^{-1}.D$ concept constructors in $\mathcal{T}$ and $\mathcal{C}$.*

The $\text{Rank}(\mathcal{T}, \mathcal{C})$ limits the maximal number of different predecessors needed to satisfy all constraints in the terminology. The above observations provide the necessary tools for describing a single Herbrand interpretation with a fixed branching outdegree using monadic sentences that simulate the (special) interpretation showing that $\mathcal{T} \not\models \mathcal{C}$.

In the Herbrand interpretation, each term represents two elements, a left element and a right element of the original interpretation, unless these two elements are made equal by the effect of a PFD. We use the following unary predicates and function symbols:

Unary function symbols f (representing a feature f) and $f_1, \dots, f_{\text{Rank}(\mathcal{T}, \mathcal{C})}$ (representing the possible inverses of f) for each feature $f \in F$. These function symbols are used, together with the constant 0 denoting the two roots

in the original interpretation, to form terms. We overload the notation for path descriptions and use $\text{Pf}(0)$ to denote terms as well.

Predicates $N^L(t)$ and $N^R(t)$ true for t representing elements that exist in the left and right parts of the original interpretation; these emulate partiality of the inverses.

Predicates $P_D^L(t)$ and $P_D^R(t)$ that are true for t that represent elements belonging to the description D in the respective parts of the original interpretation.

Predicates $E^{\text{Pf}}(t)$ true for t whenever the two elements denoted by t agree on Pf in the original interpretation.

To ensure a finite number of assertions, we assume in the following that concept descriptions and features f are subconcepts of concepts or are features appearing in $\mathcal{T}$ and $\mathcal{C}$. We call the following collection of assertions $\Pi_{\mathcal{T}}$ (assertions with a superscript X stand for a pair of assertions, one with X substituted by L and one by R):

1. Totality of features: each object must have one outgoing feature f .

$$\begin{aligned} \forall x. N^X(x) \rightarrow N^X(f(x)) \text{ for } x \neq f_i(y), \\ \forall x. N^X(f_i(x)) \rightarrow N^X(x) \end{aligned}$$

2. Functionality of features: Each element has at most one outgoing feature f .

$$\forall x. \neg(N^X(x) \wedge N^X(f(f_i(x))))$$

3. Rules of equality: equalities propagate through function application, equal nodes have the same predecessors, must exist, and must belong to the same descriptions.

$$\begin{aligned} \forall x. (N^L(f(x)) \wedge N^R(f(x))) \rightarrow (E^{Id}(x) \rightarrow E^{Id}(f(x))) \\ \forall x. (N^L(f_i(x)) \wedge N^R(f_i(x))) \rightarrow (E^{Id}(f_i(x)) \leftrightarrow E^{Id}(x)) \\ \forall x. E^{Id}(x) \rightarrow (N^L(x) \wedge N^R(x)) \\ \forall x. E^{Id}(x) \rightarrow (P_D^L(x) \leftrightarrow P_D^R(x)) \end{aligned}$$

4. Concept formation—boolean constructors: enforce the excluded-middle law and the correct behavior of conjunction.

$$\begin{aligned} \forall x. N^X(x) \rightarrow (P_D^X(x) \vee P_{\neg D}^X(x)) \\ \forall x. N^X(x) \rightarrow \neg(P_D^X(x) \wedge P_{\neg D}^X(x)) \\ \forall x. N^X(x) \rightarrow (P_{D_1 \cap D_2}^X(x) \leftrightarrow (P_{D_1}^X(x) \wedge P_{D_2}^X(x))) \end{aligned}$$

5. Concept formation—value restrictions: assert value restrictions for pairs of neighboring nodes.

$$\begin{aligned} \forall x. (N^X(x) \wedge N^X(f(x))) \rightarrow (P_{\forall f, D}^X(x) \leftrightarrow P_D^X(f(x))) \\ \forall x. (N^X(f_i(x)) \wedge N^X(x)) \rightarrow (P_{\forall f, D}^X(f_i(x)) \leftrightarrow P_D^X(x)) \end{aligned}$$

6. Concept formation—existential restrictions: satisfy existential restrictions for inverses. Note that in the case of the left and right sides of the interpretations agreeing, the appropriate predecessor can be on either side of the interpretation (first two assertions).

$$\begin{aligned} \forall x. E^{Id}(f(x)) \rightarrow (P_{\exists f, \neg D}^X(f(x)) \leftrightarrow \\ (N^L(x) \wedge P_D^L(x)) \vee (N^R(x) \wedge P_D^R(x)) \vee \\ \bigvee N^X(f_i(f(x)) \wedge P_D^X(f_i(f(x)))))) \\ \forall x. N^X(f(x)) \wedge \neg E^{Id}(f(x)) \rightarrow (P_{\exists f, \neg D}^X(f(x)) \leftrightarrow \end{aligned}$$

$$\begin{aligned} (N^X(x) \wedge P_D^X(x)) \vee \\ \bigvee N^X(f_i(f(x)) \wedge P_D^X(f_i(f(x)))))) \\ \forall x. N^X(x) \wedge x \neq f(y) \rightarrow (P_{\exists f, \neg D}^X(x) \leftrightarrow \\ \bigvee N^X(f_i(x) \wedge P_D^X(f_i(x)))) \end{aligned}$$

7. Satisfaction of simple constraints in the terminology (GCDs): enforce simple subsumption constraints present in $\mathcal{T}$.

$$\forall x. P_{D_1}^X(x) \rightarrow P_{D_2}^X(x) \text{ for } D_1 \sqsubseteq D_2 \in \mathcal{T}$$

8. Satisfaction of prefix-restricted PFDs by inverses: disallow violations of prefix-restricted PFDs due to existence of multiple inverse features agreeing on a node.

$$\begin{aligned} \forall x. \neg(N^L(x) \wedge N^R(x) \wedge \neg E^{Id}(x) \wedge E^{Id}(f(x))) \wedge \\ P_{\exists \text{Pf Pf}'^{-1}, D_1}^L(x) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_2}^R(x)) \\ \forall x. \neg(N^L(x) \wedge N^R(x) \wedge \neg E^{Id}(x) \wedge E^{Id}(f(x))) \wedge \\ P_{\exists \text{Pf Pf}'^{-1}, D_2}^L(x) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_1}^R(x)) \\ \forall x. \neg(N^X(x) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_1}^X(x) \wedge \\ N^X(f_i(f(x))) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_2}^X(f_i(f(x)))) \\ \forall x. \neg(N^X(x) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_2}^X(x) \wedge \\ N^X(f_i(f(x))) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_1}^X(f_i(f(x)))) \\ \forall x. \neg(N^X(f_i(x)) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_1}^X(f_i(x)) \wedge \\ N^X(f_j(x)) \wedge P_{\exists \text{Pf Pf}'^{-1}, D_2}^X(f_j(x))) \end{aligned}$$

for all $D_1 \sqsubseteq D_2 : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf} \in \mathcal{T}$ such that $\text{Pf Pf}' f$ is the common prefix of $\text{Pf}_1, \dots, \text{Pf}_k$.

9. Path agreements: extend the simple equality to assertions of two nodes agreeing on a path; this is necessary to avoid exponential blowup in the length of the path descriptions in the PFDs.

$$\begin{aligned} \forall x. (N^L(x) \wedge N^R(x) \wedge x \neq f_i(y)) \rightarrow \\ (E^{f, \text{Pf}}(x) \leftrightarrow E^{\text{Pf}}(f(x))) \\ \forall x. (N^L(f_i(x)) \wedge N^R(f_i(x))) \rightarrow (E^{f, \text{Pf}}(f_i(x)) \leftrightarrow E^{\text{Pf}}(x)) \end{aligned}$$

where Pf ranges over all prefixes of path descriptions in $\mathcal{T}$'s PFDs.

10. Prefix-restricted PFDs: enforce PFDs between the left and right parts of the interpretation.

$$\begin{aligned} \forall x. P_{\exists \text{Pf}'^{-1}, D_1}^L(x) \wedge P_{\exists \text{Pf}'^{-1}, D_2}^R(x) \wedge \\ E^{\text{Pf}_1}(x) \wedge \dots \wedge E^{\text{Pf}_k}(x) \rightarrow E^{\text{Pf}}(x) \\ \forall x. P_{\exists \text{Pf}'^{-1}, D_1}^R(x) \wedge P_{\exists \text{Pf}'^{-1}, D_2}^L(x) \wedge \\ E^{\text{Pf}_1}(x) \wedge \dots \wedge E^{\text{Pf}_k}(x) \rightarrow E^{\text{Pf}}(x) \end{aligned}$$

for all $D_1 \sqsubseteq D_2 : \text{Pf}' \text{Pf}_1, \dots, \text{Pf}' \text{Pf}_k \rightarrow \text{Pf}' \text{Pf} \in \mathcal{T}$.

The above assertions simulate the constraints implied by $\mathcal{T}$. To capture the violation of the constraint $\mathcal{C}$ we set $\Pi_{\mathcal{C}} = \{N^L(0), P_{D_1}^L(0), N^R(0), \neg P_{D_2}^R(0), E^{Id}(0)\}$ for $\mathcal{C}$ ordinary and $\Pi_{\mathcal{C}} = \{N^L(0), N^R(0), P_{D_1}^L(0), P_{D_2}^R(0), E^{\text{Pf}_1}(0), \dots, E^{\text{Pf}_k}(0), \neg E^{\text{Pf}}(0)\}$ for $\mathcal{C}$ a PFD.

Theorem 8 Let $\mathcal{T}$ be a $\mathcal{DLFAD}$ terminology and $\mathcal{C}$ a constraint. Then $\mathcal{T} \models \mathcal{C}$ if and only if $\Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$ is not satisfiable.

Proof (sketch): Consider a model $\mathcal{M}$ of $\Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$. We construct an interpretation $\mathcal{I} = (\Delta, (\cdot)^{\mathcal{I}})$ as follows: Let $t^0(0)$, $t^L(0)$, and $t^R(0)$ be distinct values for each term $t(0)$.

$$\begin{aligned}
\Delta = & \{t^0(0) \mid \mathcal{M} \models E^{Id}(t(0))\} \cup \\
& \{t^L(0) \mid \mathcal{M} \models N^L(t(0)) \wedge \neg E^{Id}(t(0))\} \cup \\
& \{t^R(0) \mid \mathcal{M} \models N^R(t(0)) \wedge \neg E^{Id}(t(0))\} \\
(f)^{\mathcal{I}} = & \{(t^X(0), f(t^X(0))) \mid t^X(0), f(t^X(0)) \in \Delta\} \cup \\
& \{(f_i(t^X(0)), t^X(0)) \mid t^X(0), f_i(t^X(0)) \in \Delta\} \cup \\
& \{(t^R(0), f(t^0(0))) \mid t^R(0), f(t^0(0)) \in \Delta\} \cup \\
& \{(t^L(0), f(t^0(0))) \mid t^L(0), f(t^0(0)) \in \Delta\} \\
\text{for } X \in & \{0, L, R\}, \\
(D)^{\mathcal{I}} = & \{t^0(0) \mid \mathcal{M} \models N^L(t(0)) \wedge P_D^L(t(0))\} \cup \\
& \{t^L(0) \mid \mathcal{M} \models N^L(t(0)) \wedge P_D^L(t(0))\} \cup \\
& \{t^R(0) \mid \mathcal{M} \models N^R(t(0)) \wedge P_D^R(t(0))\}
\end{aligned}$$

It is easy to verify that this interpretation satisfies all constraints in $\mathcal{T}$ and violates $\mathcal{C}$. Thus $\mathcal{T} \not\models \mathcal{C}$.

Conversely, given an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models \mathcal{C}$, we construct a model of $\Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$ as follows.

For $\mathcal{C} = D_1 \sqsubseteq D_2 : \mathbf{Pf}_1, \dots, \mathbf{Pf}_k \rightarrow \mathbf{Pf}$ a PFD, there must be elements $o_1, o_2 \in \Delta$ such that $o_1 \in (D_1)^{\mathcal{I}}$ and $o_2 \in (D_1)^{\mathcal{I}}$ such that $(\mathbf{Pf}_i)^{\mathcal{I}}(o_1) = (\mathbf{Pf}_i)^{\mathcal{I}}(o_2)$ for $0 < i \leq k$ but $(\mathbf{Pf})^{\mathcal{I}}(o_1) \neq (\mathbf{Pf})^{\mathcal{I}}(o_2)$.

We first define a part of the interpretation that consists of successors of o_1 and o_2 :

$$\begin{aligned}
& \{N^L(\mathbf{Pf}(0)), P_D^L(\mathbf{Pf}(0)) \mid (\mathbf{Pf})^{\mathcal{I}}(o_1) \in (D)^{\mathcal{I}}\} \cup \\
& \{N^R(\mathbf{Pf}(0)), P_D^R(\mathbf{Pf}(0)) \mid (\mathbf{Pf})^{\mathcal{I}}(o_2) \in (D)^{\mathcal{I}}\} \cup \\
& \{E^{\mathbf{Pf}'}(\mathbf{Pf}(0)) \mid (\mathbf{Pf} \mathbf{Pf}')^{\mathcal{I}}(o_1) = (\mathbf{Pf} \mathbf{Pf}')^{\mathcal{I}}(o_2)\} \subseteq \mathcal{M}
\end{aligned}$$

This part of $\mathcal{M}$ violates $\mathcal{C}$ and satisfies all constraints in $\mathcal{T}$ with the exception of the $\exists f^{-1}.D$. We extend $\mathcal{M}$ to satisfy the existential restrictions as follows:

for $\mathbf{Pf}(0)$ such that $E^{Id}(\mathbf{Pf}(0)) \notin \mathcal{M}$ and $N^L(\mathbf{Pf}(0))$. We chose $l \leq \text{Rank}(\mathcal{T}, \mathcal{C})$ predecessors of $(\mathbf{Pf})^{\mathcal{I}}(o_1)$ that satisfy all implied existential restrictions. and identify these with the terms $f_1(\mathbf{Pf}(0)), \dots, f_l(\mathbf{Pf}(0))$ and set

$$N^L(f_i(\mathbf{Pf}(0))), P_{D(f_i(\mathbf{Pf}(0)))}^L, P_{\exists f^{-1}.D_i}^L(\mathbf{Pf}(0)) \in \mathcal{M}$$

for all the chosen D_i predecessors of $(\mathbf{Pf})^{\mathcal{I}}(o_1)$ for $0 < i \leq l$. For all the terms $f_i(\mathbf{Pf}(0))$ we extract a tree interpretation from $\mathcal{I}$. Similarly, we extend $\mathcal{M}$ on the right side.

for $\mathbf{Pf}(0)$ such that $E^{Id}(\mathbf{Pf}(0)) \in \mathcal{M}$ we follow the same steps as above for the left part only.

Note that since $\mathcal{I}$ satisfies $\mathcal{T}$ all (translations of) ordinary constraints are satisfied and no pair of nodes on the left/right sides can violate a prefix-restricted PFD thus satisfying constraints generated from PFDs (8-10 above).

For $\mathcal{C} = D_1 \sqsubseteq D_2$ a simple constraint, there must be an element $o \in \Delta$ such that $o \in (D_1)^{\mathcal{I}}$ and $o \notin (D_2)^{\mathcal{I}}$. We define $\mathcal{M}$ using the same process as above noting that the two roots o_1 and o_2 are already equal in $\mathcal{I}$. $\square$

As a side-effect of the above construction, we can now transform an arbitrary interpretation that satisfies $\mathcal{T}$ and violates $\mathcal{C}$ to an almost-tree interpretation informally referred to above in order to motivate the monadic assertions in $\Pi_{\mathcal{T}}$ and $\Pi_{\mathcal{C}}$.

The translation, in itself polynomial, therefore provides an EXPTIME decision procedure by appealing to Proposition 4. Completeness follows from EXPTIME-hardness of the implication problem for the $\{D_1 \sqcap D_2, \forall f.D\}$ fragment [Toman and Weddell, 2001; 2004b].

Corollary 9 *The implication problem for $\mathcal{DLFAD}$ with prefix-restricted PFDs is EXPTIME-complete.*

Using similar techniques, the $\exists f^{-1}.D$ concept constructor can be generalized to more general *number restrictions* requiring lower and upper bounds (coded in binary) on the number of f predecessors while still maintaining the complexity bound.

4.2 Coherent Terminologies

The second of our conditions for recovering decidability is to impose a coherency condition on terminologies themselves. The main advantage of this approach is that we thereby regain the ability for unrestricted use of PFDs in terminologies. The disadvantage is roughly that there is a “single use” restriction on using feature inversions in terminologies.

Definition 10 (Coherent Terminology) *A terminology $\mathcal{T}$ is coherent if*

$$\mathcal{T} \models (\exists f^{-1}.D) \sqcap (\exists f^{-1}.E) \sqsubseteq \exists f^{-1}.(D \sqcap E)$$

for all descriptions D, E that appear as subconcepts of concepts that appear in $\mathcal{T}$, or their negations.

Note that we can *syntactically guarantee* that $\mathcal{T}$ is coherent by adding the $(\exists f^{-1}.D) \sqcap (\exists f^{-1}.E) \sqsubseteq \exists f^{-1}.(D \sqcap E)$ assertions to $\mathcal{T}$ for all descriptions D, E that appear in $\mathcal{T}$. This restriction allows us to construct interpretations of non-PFD descriptions that have the following property:

Definition 11 *An interpretation $(\Delta, (\cdot)^{\mathcal{I}})$ is coherent if, for any $f \in F$, description D and $x, y \in \Delta$, $y \in (D)^{\mathcal{I}}$ if $x \in (D)^{\mathcal{I}}$ and $(f)^{\mathcal{I}}(x) = (f)^{\mathcal{I}}(y)$.*

Lemma 12 *Let $\mathcal{T}$ be a coherent terminology, $\mathcal{C}$ a simple constraint, and $\mathcal{I}$ an interpretation such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models \mathcal{C}$. Then there is a coherent interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathcal{T}$ and $\mathcal{I}' \not\models \mathcal{C}$.*

Proof (sketch): Consider distinct $x, y \in \Delta_{\mathcal{I}}$ such that (i) $x \in (D_1)^{\mathcal{I}}$, (ii) $y \in (D_2)^{\mathcal{I}}$, and (iii) $(f)^{\mathcal{I}}(x) = (f)^{\mathcal{I}}(y)$. Then, since $\mathcal{T}$ is coherent, $x \in (D_1 \sqcap D_2)^{\mathcal{I}}$. For, $x \in (D_1 \sqcap \neg D_2)^{\mathcal{I}}$ leads to $(f_i)^{\mathcal{I}}(x) \in (\exists f_i^{-1}.(D_1 \sqcap \neg D_2) \sqcap \exists f_i^{-1}.D_2)^{\mathcal{I}}$, a contradiction. Thus, as models of $\mathcal{DLFAD}$ have the tree model property, we can remove the farther of x or y and all its descendants, where the distance is measured from the node falsifying $\mathcal{C}$ in $\mathcal{I}$. The resulting interpretation still satisfies $\mathcal{T}$ and falsifies $\mathcal{C}$. Repeating this process yields a coherent interpretation. $\square$

By restricting logical implication problems for $\mathcal{DLFAD}$ to cases in which terminologies are coherent, it becomes possible to apply reductions to satisfiability problems for Ackerman formulae.

Theorem 13 Let $\mathcal{T}$ be a coherent $\mathcal{DLFAD}$ terminology. Then the implication problem $\mathcal{T} \models \mathcal{C}$ is decidable and EXPTIME-complete.

Proof (sketch): We use a reduction similar to the one for prefix-restricted terminologies. However, as the terminology is coherent, in the left and right parts of the interpretation objects can never have more than one incoming feature with the same name and thus a single *inverse* f_1 for $f \in F$ is always sufficient to model this situation using the monadic formulas. Also, the left and right sides of the interpretation vacuously satisfy all PFDs (and thus the assertions in item 8 can be dropped). The interaction between PFDs and inverses is now completely captured by the auxiliary E^{Pf} predicates. For details see [Toman and Weddell, 2004a]. $\square$

Note that we cannot consider coherent interpretations only as then *all* PFDs would be vacuously satisfied—no two nodes could possibly agree on a common path. This would make all PFDs trivial *coherent* consequences (i.e., when only coherent interpretations are considered). Our coherency restriction on terminologies is weaker: it only postulates that we can avoid multiple f predecessors if we wish to do so.

5 Summary and Future Work

We have presented a pair of conditions, one syntactic and one semantic, under which it becomes possible to combine feature inversion with path-functional dependencies in a boolean complete description logic while still ensuring that the associated logical implication problem is EXPTIME-complete; the problem is shown to be undecidable otherwise. The second of these conditions resolves an open issue on decidability of an analogous implication problem in [Biskup and Polle, 2003].

A natural extension of the description logic presented here allows *regular languages* (L) to replace path expressions, yielding the $\forall L.D$, $\exists L.D$, $\exists L^{-1}.D$, and $D : L \rightarrow L'$ constructors, and developing a decision procedure using the approach in [Toman and Weddell, 2004b]. One of the main applications of such an extension we envision is describing data structures for purposes of query optimization, extending [Liu *et al.*, 2002] to inductive data types.

Another direction of research considers weaker restrictions on $\mathcal{DLFAD}$ terminologies that still guarantee decidability, e.g., relaxing our *coherence* condition with respect to the unary PFDs actually present in a terminology.

References

- [Berger, 1966] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.
- [Biskup and Polle, 2000] Joachim Biskup and Torsten Polle. Decomposition of Database Classes under Path Functional Dependencies and Onto Constraints. In *Foundations of Information and Knowledge Systems*, pages 31–49, 2000.
- [Biskup and Polle, 2003] Joachim Biskup and Torsten Polle. Adding inclusion dependencies to an object-oriented data model with uniqueness constraints. *Acta Informatica*, 39:391–449, 2003.
- [Biskup *et al.*, 1996] Joachim Biskup, Ralf Menzel, Torsten Polle, and Yehoshua Sagiv. Decomposition of Relationships through Pivoting. In *Conceptual Modeling - ER'96*, pages 28–41, 1996.
- [Calvanese *et al.*, 2001] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 155–160, 2001.
- [DeHaan *et al.*, 2003] David DeHaan, David Toman, and Grant E. Weddell. Rewriting Aggregate Queries using Description Logics. In *Description Logics 2003*, pages 103–112. CEUR-WS vol.81, 2003.
- [Fürier, 1981] Martin Fürier. Alternation and the Ackermann Case of the Decision Problem. *L'Enseignement Math.*, 27:137–162, 1981.
- [Khizder *et al.*, 2000] Vitaliy L. Khizder, David Toman, and Grant E. Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, pages 1017–1032, 2000.
- [Liu *et al.*, 2002] Huizhu Liu, David Toman, and Grant E. Weddell. Fine Grained Information Integration with Description Logic. In *Description Logics 2002*, pages 1–12. CEUR-WS vol.53, 2002.
- [Lutz and Milicic, 2004] C. Lutz and M. Milicic. Description Logics with Concrete Domains and Functional Dependencies. In *Proc. European Conference on Artificial Intelligence (ECAI)*, pages 378–382, 2004.
- [Lutz *et al.*, 2003] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, Nominals, and Concrete Domains. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 349–354, 2003.
- [Stanchev and Weddell, 2003] Lubomir Stanchev and Grant E. Weddell. Index Selection for Embedded Control Applications using Description Logics. In *Description Logics 2003*, pages 9–18. CEUR-WS vol.81, 2003.
- [Toman and Weddell, 2001] David Toman and Grant E. Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Description Logics 2001*, pages 76–85. CEUR-WS vol.49, 2001.
- [Toman and Weddell, 2004a] David Toman and Grant E. Weddell. Attribute Inversion in Description Logics with Path Functional Dependencies. In *Description Logics 2004*, pages 178–187. CEUR-WS vol.104, 2004.
- [Toman and Weddell, 2004b] David Toman and Grant E. Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 2004. doi:10.1016/j.tcs.2004.10.036.
- [van Emde Boas, 1997] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture notes in pure and applied mathematics*, pages 331–363. Marcel Dekker Inc., 1997.
- [Weddell, 1989] Grant Weddell. A Theory of Functional Dependencies for Object Oriented Data Models. In *International Conference on Deductive and Object-Oriented Databases*, pages 165–184, 1989.

Ordering Heuristics for Description Logic Reasoning

Dmitry Tsarkov and Ian Horrocks

School of Computer Science

University of Manchester

Manchester, UK

traskov|horrocks@cs.man.ac.uk

Abstract

We present a new architecture for Description Logic implementations, a range of new optimisation techniques and an empirical analysis of their effectiveness.

1 Introduction

Description Logics (DLs) are a family of logic based knowledge representation formalisms. Although they have a range of applications (e.g., configuration [McGuinness & Wright, 1998], and reasoning with database schemas and queries [Calvanese *et al.*, 1998b; 1998a]), they are perhaps best known as the basis for widely used ontology languages such as OIL, DAML+OIL and OWL [Horrocks *et al.*, 2003]. As well as DLs providing the formal underpinnings for these languages (i.e., a declarative semantics), DL systems are also used to provide computational services for ontology tools and applications [Knublauch *et al.*, 2004; Rector, 2003].

Most modern DL systems are based on tableaux algorithms. Such algorithms were first introduced by Schmidt-Schauß and Smolka [Schmidt-Schauß & Smolka, 1991], and subsequently extended to deal with ever more expressive logics [Baader *et al.*, 2003]. Many systems now implement the *SHIQ* DL, a tableaux algorithm for which was first presented in [Horrocks *et al.*, 1999]; this logic is very expressive, and corresponds closely to the OWL ontology language. In spite of the high worst case complexity of the satisfiability/subsumption problem for this logic (ExpTime-complete), highly optimised implementations have been shown to work well in many realistic (ontology) applications [Horrocks, 1998].

Optimisation is crucial to the viability of tableaux based systems: in experiments using both artificial test data and application ontologies, (relatively) unoptimised systems performed very badly, often being (at least) several orders of magnitude slower than optimised systems; in many cases, hours of processing time (in some cases even hundreds of hours) proved insufficient for unoptimised systems to solve problems that took only a few milliseconds for an optimised system [Massacci, 1999; Horrocks & Patel-Schneider, 1998]. Modern systems typically employ a wide range of optimisations, including (at least) those described in [Baader *et al.*, 1994; Horrocks & Patel-Schneider, 1999].

Tableaux algorithms try to construct a graph (usually a tree) representation of a model of a concept, the structure of which is determined by syntactic decomposition of the concept. Most implementations employ a space saving optimisa-

tion known as the *trace technique* that uses a top-down construction requiring (for PSpace logics) only polynomial space in order to delineate a tree structure that may be exponential in size (with respect to the size of the input concept). For the ExpTime logics implemented in modern systems, however, guaranteeing polynomial space usage is no longer an option. Moreover, for logics that support inverse roles (such as *SHIQ*), a strictly top down approach is no longer possible as constraints may be propagated both “up” and “down” the edges in the tree.

We describe an alternative architecture for tableaux implementations that uses a (set of) queue(s) instead of (an adaptation of) the standard top-down approach. This architecture, which we have implemented in our new FaCT++ system, has a number of advantages when compared to the top-down approach. Firstly, it is applicable to a much wider range of logics, including the expressive logics implemented in modern systems, because it makes no assumptions about the structure of the graph (in particular, whether tree shaped or not), or the order in which the graph will be constructed. Secondly, it allows for the use of more powerful heuristics that try to improve typical case performance by varying the global order in which different syntactic structures are decomposed; in a top-down construction, such heuristics can only operate on a local region of the graph—typically a single vertex.

2 Preliminaries

We present here a brief introduction to DL (in particular *SHIQ*) syntax, semantics and reasoning; for further details the reader is referred to [Baader *et al.*, 2003].

2.1 Description Logics

Syntax Let $\mathbf{R}$ be a set of *role names* with both transitive and normal role names $\mathbf{R}_+ \cup \mathbf{R}_P = \mathbf{R}$, where $\mathbf{R}_+ \cap \mathbf{R}_P = \emptyset$. The set of *SHIQ-roles* (or *roles* for short) is $\mathbf{R} \cup \{\mathbf{R}^- \mid \mathbf{R} \in \mathbf{R}\}$. Let N_C be a set of *concept names*. The set of *SHIQ-concepts* (or *concepts* for short) is the smallest set such that every concept name $C \in N_C$ is a concept, and if C and D are concepts, R is a role, S is a *simple* role¹ and $n \in \mathbb{N}$, then $(C \sqcap D)$, $(C \sqcup D)$, $(\neg C)$, $(\forall R.C)$, $(\exists R.C)$, $(\leq nR.C)$ and $(\geq nR.C)$ are also concepts; the last four are called, respectively, value, exists, atmost and atleast restrictions.

For R and S (possibly inverse) roles, $R \sqsubseteq S$ is called a *role inclusion axiom*, and a finite set of role inclusion axioms is called a *role hierarchy*. For C and D (possibly complex)

¹A simple role is one that is neither transitive nor has any transitive subroles. Restricting number restrictions to simple roles is required for decidability [Horrocks *et al.*, 1999].

concepts, $C \sqsubseteq D$ is called a *general concept inclusion* (GCI), and a finite set of GCIs is called a *TBox*.

Semantics An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for $P \in \mathbf{R}$ and $R \in \mathbf{R}_+$, $\langle x, y \rangle \in P^{\mathcal{I}}$ iff $\langle y, x \rangle \in P^{-\mathcal{I}}$, and if $\langle x, y \rangle \in R^{\mathcal{I}}$ and $\langle y, z \rangle \in R^{\mathcal{I}}$ then $\langle x, z \rangle \in R^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$\begin{aligned} (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, & (C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\ \neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\ (\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, C) \neq \emptyset\}, \\ (\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(x, \neg C) = \emptyset\}, \\ (\leq n R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#R^{\mathcal{I}}(x, C) \leq n\}, \text{ and} \\ (\geq n R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#R^{\mathcal{I}}(x, C) \geq n\}, \end{aligned}$$

where $\#M$ is the cardinality of a set M and $R^{\mathcal{I}}(x, C)$ is defined as $\{y \mid \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for each $R \sqsubseteq S \in \mathcal{R}$, and it satisfies a TBox $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each $C \sqsubseteq D \in \mathcal{T}$; such an interpretation is called a *model* of $\mathcal{R}$ and $\mathcal{T}$.

A concept C is *satisfiable* w.r.t. a role hierarchy $\mathcal{R}$ and a TBox $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a *model* of C w.r.t. $\mathcal{R}$ and $\mathcal{T}$. As usual for expressive DLs, subsumption can be reduced to satisfiability, and reasoning w.r.t. a TBox and role hierarchy can be reduced to reasoning w.r.t. a role hierarchy only [Horricks *et al.*, 1999].

2.2 Tableaux Algorithms

The basic idea behind a tableau algorithm is to take an input concept C and role hierarchy $\mathcal{R}$, and to try to prove the satisfiability of C w.r.t. $\mathcal{R}$ by constructing a model $\mathcal{I}$ of C w.r.t. $\mathcal{R}$. This is done by syntactically decomposing C so as to derive constraints on the structure of such a model. For example, any model of C must, by definition, contain some individual x such that x is an element of $C^{\mathcal{I}}$, and if C is of the form $\exists R.D$, then the model must also contain an individual y such that $\langle x, y \rangle \in R^{\mathcal{I}}$ and y is an element of $D^{\mathcal{I}}$; if D is non-atomic, then continuing with the decomposition of D would lead to additional constraints. The construction fails if the constraints include a *clash* (an obvious contradiction), e.g., if some individual z must be an element of both C and $\neg C$ for some concept C . Algorithms are normally designed so that they are guaranteed to terminate, and guaranteed to construct a model if one exists; such an algorithm is clearly a decision procedure for concept satisfiability.

In practice, algorithms often work on a tree shaped graph that has a close correspondence to a model; this may be because, e.g., models could be non-finite (although obviously finitely representable), or non-trees (although usually tree-like). Typically this will be a labelled graph (usually a tree or collection of trees) where nodes represent individuals in the model, and are labelled with a set of concepts of which they are instances, and edges represent role relationships between pairs of individuals, and are labelled with a set of role names.

The decomposition and construction is usually carried out by applying so called tableaux expansion rules to the concepts in node labels, with one rule being defined for each of the syntactic constructs in the language (with the exception of

negation, which is pushed inwards using re-writings such as de Morgan's laws, until it applies only to atomic concepts). For example, the expansion rule for conjunction causes C and D to be added to any node label already containing $C \sqcap D$ (in order to guarantee termination, side conditions prevent rules from being applied if they do not change either the graph or its labelling).

There are two forms of non-determinism in the expansion procedure. In the first place, many rules may be simultaneously applicable, and some order of rule applications must be chosen. From a correctness perspective, this choice is usually irrelevant² (because, if there is a model, then it will be found by any expansion ordering), but as we will see later, the order of expansion can have a big effect on efficiency. In the second place, some rules expand the graph non-deterministically; e.g., the expansion rule for disjunction causes *either* C or D to be added to any node label already containing $C \sqcup D$. From a correctness perspective, this choice is relevant (because one choice may lead to the successful construction of a model, while another one does not), and is usually dealt with by backtracking search. Although such search must (in the worst case) consider *all* possible expansions, the order in which they are considered can still have a big effect on efficiency.

Two kinds of rule will be of particular interest in the following discussion: *non-deterministic* rules, such as the $\sqcup$ -rule mentioned above, and *generating* rules, such as the $\exists$ -rule, that add new nodes to the graph. Applying these rules is likely to be more “costly”, as they either increase the size of the graph or increase the size of the search space, and they are typically applied with lower priority than other rules.

3 FaCT++ System Architecture

As discussed above, many implementations use a top-down expansion based on the trace technique. The idea of the top-down expansion is to apply the $\exists$ -rule with the lowest priority (i.e., only apply this rule when no other rule is applicable); the added refinement of the trace technique is to discard fully expanded sub-trees, so that only a single “trace” (i.e., a branch of the tree) is kept in memory at any one time.

This technique has the advantage of being very simple and easy to implement—a procedure that exhaustively expands a node label can be applied to the current node and then, recursively, to each of its successors. It does, however, have some serious drawbacks. In the first place, for logics with inverse roles, the top-down method simply breaks down as it relies on the fact that rules only ever add concepts to the label of the node to which they are applied or to the label of one of its successor nodes. The result is that, once the rules have been exhaustively applied to a given node label, no further expansion of that label will be possible. In the presence of inverse roles, expansion rules may also add concepts to the labels of predecessor nodes, which could then require further expansion. Moreover, discarding fully expanded sub-trees may no longer be possible, as the expansion of a concept added to the label of a predecessor may cause concepts to be added to the label of a sibling node that had previously been fully expanded.

In the second place, the top down method forces non-deterministic rules to be applied with a higher priority than generating rules. As the size of the search space caused by non-deterministic rule expansions is, in practice, by

²Although the correctness of some algorithms requires a priority ordering for different rules.

far the most serious problem for tableaux based systems [Horrocks, 1997], it may be advantageous to apply non-deterministic rules with the lowest priority [Giunchiglia & Sebastiani, 1996]. In fact, top-down implementations typically apply non-deterministic rules with a priority that is lower than that of all of the other rules *except* the generating rules [Horrocks & Patel-Schneider, 1999].

ToDo List Architecture The FaCT++ system was designed with the intention of implementing DLs that include inverse roles, and of investigating new optimisation techniques, including new ordering heuristics. Currently, FaCT++ implements *SHIF*, a slightly less expressive variant of *SHIQ* where the values in atleast and atmost restrictions can only be zero or one.³

Instead of the top-down approach, FaCT++ uses a *ToDo list* to control the application of the expansion rules. The basic idea behind this approach is that rules may become applicable whenever a concept is added to a node label. When this happens, a note of the node/concept pair is added to the *ToDo* list. The *ToDo* list sorts all entries according to some order, and gives access to the “first” element in the list.

A given tableaux algorithm takes an entry from the *ToDo* list and processes it according to the expansion rule(s) relevant to the entry (if any). During the expansion process, new concepts may be added to node labels, and hence entries may be added to the *ToDo* list. The process continues until either a clash occurs or the *ToDo* list becomes empty.

In FaCT++ we implement the *ToDo* list architecture as a set of queues (FIFO buffers). It is possible to set a priority for each rule type (e.g., $\sqcap$ and $\exists$), and a separate queue is implemented for each unique priority. Whenever the expansion algorithm asks for a new entry, it is taken from the non-empty queue with the highest priority, and the algorithm terminates when all the queues are empty. This means that if the $\exists$ -rule has a low priority (say 0), and all other rules have the same priority (say 1), then the expansion will be (modulo inverse roles) top-down and breadth first; if stacks (LIFO buffers) were used instead of queues with the same priorities, then the expansion would simulate the standard top-down method.

4 Heuristics

When implementing reasoning algorithms, heuristics can be used to try to find a “good” order in which to apply inference rules (we will call these *rule-ordering* heuristics) and, for non-deterministic rules, the order in which to explore the different expansion choices offered by rule applications (we will call these *expansion-ordering* heuristics). The aim is to choose an order that leads rapidly to the discovery of a model (in case the input is satisfiable) or to a proof that no model exists (in case the input is unsatisfiable). The usual technique is to compute a weighting for each available option, and to choose the option with the highest (or lowest) weight. Much of the “art” in devising useful heuristics is in finding a suitable compromise between the cost of computing the weightings and their accuracy in predicting good orderings.

Such heuristics can be very effective in improving the performance of propositional satisfiability (SAT) reasoners [Freeman, 1995], but finding useful heuristics for description and modal logics has proved to be more difficult. Choosing a good heuristic, or at least not choosing a bad one, is very important: an inappropriate heuristic may not simply fail to improve performance, it may seriously degrade it. Even more

problematical is, given a range of possible heuristics, choosing the best one to use for a given (type of) problem.

So far, the heuristics tried with DL reasoners have mainly been adaptations of those already developed for SAT reasoners, such as the well known MOMS heuristic [Freeman, 1995] and Jeroslow and Wang’s weighted occurrences heuristic [Jeroslow & Wang, 1990]. These proved to be largely ineffective, and even to degrade performance due to an adverse interaction with backjumping [Baader *et al.*, 2003]. An alternative heuristic, first presented in [Horrocks, 1997], tries to maximise the effect of dependency directed backtracking (backjumping) by preferentially choosing expansions that introduce concept with “old” dependencies. Even this heuristic, however, has relatively little effect on performance with realistic problems, e.g., problems encountered when reasoning with application ontologies.

We conjecture that the standard top-down architecture has contributed to the difficulty in finding useful heuristics as it rules out many possible choices of rule-ordering; in particular, the top-down technique may require generating rules to be applied with a low priority, and so lead to non-deterministic rules being applied before deterministic generating rules. In contrast, the *ToDo* list architecture gives a much wider range of possible rule orderings, and so has allowed us to investigate a range of new rule-ordering heuristics, in particular heuristics that give non-deterministic rules the lowest priority.

Another factor that has contributed to the weakness of SAT derived heuristics is that they treat concepts as though they were atoms. This is obviously appropriate in the case of propositional satisfiability, but not in the case of concept satisfiability where sub-concepts may have a complex structure. We have also investigated expansion-ordering heuristics that take into account this structure, in particular a concept’s size, maximum quantifier depth, and frequency of usage in the knowledge base.

Implementation in FaCT++ The FaCT++ reasoner uses the standard backtracking search technique to explore the different possible expansions offered by non-deterministic rules (such as the $\sqcup$ -rule). Before applying a non-deterministic rule, the current state is saved, and when backtracking, the state is restored before re-applying the same rule (with a different expansion choice). When inverse roles are supported, it is possible for a sequence of deterministic rule applications to propagate changes throughout the graph, and it may, therefore, be necessary to save and restore the whole graph structure (in addition to other data structures such as the *ToDo* list). FaCT++ tries to minimise the potentially high cost of these operations by lazily saving the graph, (i.e., saving parts of the graph only as necessitated by the expansion), but the cost of saving the state still makes it expensive to apply a non-deterministic rule, even if the state is never restored during backtracking.

As discussed in Section 3, FaCT++ uses a *ToDo* list architecture with separate queues for each priority level. Different rule-ordering heuristics can, therefore, be tried simply by varying the priorities assigned to different rule types. Low priorities are typically given to generating and non-deterministic rules, but the *ToDo* list architecture allows different priority ordering of these rule types; in contrast, the top-down architecture forces a lower priority to be given to generating rules.

FaCT++ also includes a range of different expansion-ordering heuristics that can be used to choose the order in which to explore the different expansion choices offered by the non-deterministic $\sqcup$ -rule. This ordering can be on the ba-

³ *SHIF* corresponds to the OWL-Lite ontology language [Horrocks *et al.*, 2003].

sis of the size, maximum quantifier depth, or frequency of usage of each of the concepts in the disjunction, and the order can be either ascending (smallest size, minimum depth and lowest frequency first) or descending. In order to avoid the cost of repeatedly computing such values, FaCT++ gathers all the relevant statistics for each concept as the knowledge base is loaded, and caches them for later use.

5 Empirical Analysis

In order to evaluate the usefulness of the heuristics implemented in FaCT++, we have carried out an empirical analysis using both real-life ontologies and artificial tests from the DL'98 test suite [Horrocks & Patel-Schneider, 1998].

Ontologies can vary widely in terms of size and complexity (e.g., structure of concepts, and types of axiom used). We used three ontologies with different characteristics in order to see how the heuristics would perform in each case:

WineFood A sample ontology that makes up part of the OWL test suit⁴ [Carroll & De Roo, 2004]; it is small, but has a complex structure and includes 150 GCIs.

DOLCE A foundational (top-level) ontology, developed in the WonderWeb project [Gangemi *et al.*, 2002]; it is of medium size and medium complexity.

GALEN The anatomical part of the well-known medical terminology ontology [Rogers *et al.*, 2001]; it is large (4,000 concepts) and has a relatively simple structure, but includes over 400 GCIs.

FaCT++ separates the classification process into satisfiability testing (SAT) and subsumption testing (SUB) phases; the results from the SAT phase are cached and used to speed up subsequent tests via a standard “model-merging” optimisation [Horrocks & Patel-Schneider, 1999]. FaCT++ allows different heuristics to be used in the two phases of the process; this is because the tests have different characteristics: in the SAT phase, nearly all of the tests are satisfiable (ontologies typically do not give names to unsatisfiable concepts), while in the SUB phase, up to one in four of the tests are unsatisfiable. We measured the time (in CPU seconds) taken by FaCT++ to complete each phase.

In addition to the ontologies, we used artificially generated test data from the DL'98 test suite. Artificial tests are in some sense corner cases for a DL reasoner designed primarily for ontology reasoning, and these tests are mainly intended to investigate the effect of hard problems with very artificial structures on the behaviour of our heuristics. For this purpose we selected from the test suite several of the tests that proved to be hard for FaCT++.

Each of these tests consists of a set of 21 satisfiability testing problems of similar structure, but (supposedly exponentially) increasing difficulty; the idea of the test is to determine the number of the largest problem that can be solved within a fixed amount of processing time (100 seconds of CPU time in our case). The names of the tests are of the form “test_p” or “test_n”, where “test” refers to the kind of problem (e.g., the “ph” tests are derived from encodings of pigeon hole sorting problems), and “p/n” refers to whether the problems in the test set are satisfiable (n) or unsatisfiable (p). For these tests we have reported the number of the largest problem solved in less than 100 seconds (21 means that all the problems were solved), along with the time (in CPU seconds) taken for the hardest problem that was successfully solved.

⁴This ontology therefore has a much weaker claim to being “real-life”.

For all the tests, FaCT++ v.0.99.2 was used on Pentium 4 2.2 GHz machine with 512Mb of memory, running Linux. Times were averaged over 3 test runs.

5.1 Rule-ordering Heuristics

In these tests we tried a range of different rule-ordering strategies. Each “strategy” is shown as a sequence of letters specifying the priorities (highest first) of the different rule types, where “O” refers to the $\sqcup$ -rule, “E” to the $\exists$ -rule, and “a” to any other rule type. E.g., “aO” describes the strategy where the $\sqcup$ -rule has the lowest priority, and all other rules have an equal higher priority.

Ontology tests The results of using different rule-ordering strategies with the various ontologies are shown in Table 1. All ontologies were tested with the best disjunction-ordering heuristic, as determined in separate tests (see below).

KB	DOLCE		WineFood		GALEN	
	SAT	SUB	SAT	SUB	SAT	SUB
a	0.74	0.74	0.22	2.44	99.44	1678.11
aO	0.64	0.68	0.14	1.64	29.80	569.64
aEO	0.58	0.57	0.15	1.67	9.88	173.79
aE	0.60	0.58	0.27	2.87	13.35	205.32
aOE	0.61	0.59	0.27	2.93	13.22	201.40

Table 1: Ontology tests with different rule-orderings

The first thing to note is that rule-orderings have relatively little effect on the DOLCE and WineFood ontologies; in contrast, the performance of the best and worst strategies differs by a factor of almost 10 in the GALEN tests. Even in the GALEN case, however, the difference between the “-O” strategies (i.e., those that assign the lowest priority to the $\sqcup$ -rule) and “-E” strategies (i.e., those that assign the lowest priority to the $\exists$ -rule) is relatively small. In most cases the best result is given by the “aEO” strategy, i.e., by assigning the lowest priority to the $\sqcup$ -rule and the next lowest priority to the $\exists$ -rule, and even when “aEO” is not the best strategy, the difference between it and the best strategy is very small. Moreover, the difference between the “aEO” and “aOE” strategies is small in most cases, and never more than a factor of 2.

DL98 tests The results of using different rule-ordering strategies with the DL98 tests are shown in Table 2. The first thing to note from these results is that rule-ordering heuristics can have a much more significant effect than in the ontology tests: in some cases the performance of the best and worst strategies differs by a factor of more than 100. In most tests, the “-E” strategies give the best results, with the difference between “-O” and “-E” strategies being much more marked than in the case of the ontology tests. In the case of the d4_n test, however, performance is dramatically improved (by a factor of 20) when an “-O” strategy is used.

test	br_n		br_p		d4_n		ph_n		ph_p	
	last	time	last	time	last	time	last	time	last	time
a	8	16.7	9	20.5	20	94.8	11	99.0	7	15.5
aO	11	38.2	11	38.1	21	0.8	10	10.8	7	32.1
aEO	11	38.8	11	39.0	21	0.8	10	10.9	7	32.9
aE	11	17.1	12	18.3	21	15.7	11	97.4	7	15.2
aOE	11	19.3	12	21.1	21	16.1	11	99.5	7	15.9

Table 2: DL-98 tests with different rule-ordering strategies

5.2 Expansion-ordering Heuristics

In these tests we tried a range of different expansion-ordering heuristics. Each heuristic is denoted by two letters, the first of

which indicates whether the ordering is based on concept size (“S”), maximum depth (“D”) or frequency of usage (“F”), and the second of which indicates ascending (“a”) or descending (“d”) order. In each group of tests we used the best rule-ordering heuristic as determined by the tests in Section 5.1.

Ontology tests For the ontology tests, we tried different orderings for the SAT and SUB phases of classification. The results are presented in Tables 3, 4 and 5; the first figure in each column is the time taken by the SAT phase using the given ordering, and the remaining figures are the subsequent times taken using different SUB phase orderings.

For DOLCE (Table 3), the difference between the best and worst orderings was a factor of about 4, and many possible orderings were near optimal. For WineFood (Table 4), the difference between the best and worst orderings was a factor of about 2, and using Sd for SAT tests and Dd for SUB tests gave the best result, although several other orderings gave similar results. For GALEN (Table 5), the difference between the best and worst orderings was so large that we were only the orderings given allowed tests to be completed in a reasonable time. The best result was given by using Da for both phases.

SAT	Sa	Da	Fa	Sd	Dd	Fd
SUB	1.29	1.28	1.24	0.61	0.6	0.6
Sa	2.53	2.52	2.52	2.46	2.45	2.41
Da	2.53	2.53	2.53	2.44	2.44	2.41
Fa	0.91	0.91	0.89	0.97	0.98	0.88
Sd	0.61	0.60	0.60	0.59	0.59	0.59
Dd	0.60	0.60	0.60	0.60	0.59	0.60
Fd	1.33	1.34	1.33	1.30	1.34	1.33

Table 3: DOLCE test with different expansion-orderings

SAT	Sa	Da	Fa	Sd	Dd	Fd
SUB	0.26	0.29	0.19	0.13	0.13	0.20
Sa	3.15	3.57	3.27	3.21	3.21	3.68
Da	3.54	3.57	3.44	3.20	3.40	3.47
Fa	3.67	3.57	2.32	2.12	2.41	2.35
Sd	1.77	1.80	1.71	1.80	1.80	1.83
Dd	1.69	1.77	1.87	1.66	1.78	1.78
Fd	2.30	2.26	2.75	3.14	3.54	2.76

Table 4: WineFood test with different expansion-orderings

SAT	Sa	Da
SUB	18.76	9.88
Sa	276.90	276.16
Da	185.79	172.89
Fd	1049.74	943.06

Table 5: GALEN test with different expansion-orderings

DL98 tests Table 6 presents the results for the DL98 tests. Each column shows the times taken using different expansion orderings to solve the hardest problem that was solvable within the stipulated time limit using any ordering.

In almost every test, the difference between the best and worst strategies is large: a factor of more than 300 in the d4_n test. Moreover, strategies that are good in one test can be very bad in another (the Sd and Dd strategies are the best ones in the branch tests (br_n and br_p), but (by far) the worst in the d4_n test), and this is not strongly dependent on the satisfiability result (in the br tests, all strategies perform similarly in both satisfiable and unsatisfiable cases). The Fd strategy is, however, either optimal or near optimal in all cases.

order	br_n	br_p	d4_n	ph_n	ph_p
	test 11	test 12	test 21	test 10	test 7
Sa	22.6	24.8	0.9	8.1	29.5
Da	22.6	24.8	0.9	>300	24.5
Fa	>300	>300	32.0	22.9	20.2
Sd	17.0	18.3	>300	38.7	24.7
Dd	17.1	18.3	>300	19.7	19.3
Fd	22.2	25.1	0.8	6.2	15.3

Table 6: DL98 tests with different Or strategies

5.3 Analysis

The different rule-ordering heuristics we tried had relatively little effect on the performance of the reasoner when classifying the DOLCE and WineFood ontologies. With the GALEN ontology, any strategy that gave a lower priority to the $\exists$ - and $\sqcup$ -rules worked reasonably well, and the aEO strategy was optimal or near-optimal in all cases. The crucial factor with GALEN is giving low priority to the $\exists$ -rule. This is due to the fact that GALEN is large, contains many GCI and also contains existential cycles in concept inclusion axioms (e.g., $C \sqsubseteq \exists R.D$ and $D \sqsubseteq \exists R^-.C$); as a result, the graph can grow very large, and this increases both the size of the search space (because GCI related non-determinism may apply on a per-node basis) and the cost of saving and restoring the state during backtracking search. Giving a low priority to the $\exists$ -rule minimises the size of the graph and hence can reduce both the size of the search space and the cost of saving and restoring. This effect is less noticeable with the other ontologies because their smaller size and/or lower number of GCIs greatly reduces the maximum size of graphs and/or search space. In view of these results, FaCT++’s default rule-ordering strategy has been set to aEO.⁵

The picture is quite different in the case of the DL’98 tests. Here, different strategies can make a large difference, and no one strategy is universally near optimal. This is to be expected, given that some of the tests include very little non-determinism, but are designed to force the construction of very large models (and hence graphs), while others are highly non-deterministic, but have only very small models. Given that these extreme cases are not representative of typical real-life ontologies, the test results may not be directly relevant to a system designed to deal with such ontologies. It is interesting, however, to see how *badly* the heuristics can behave in such cases: in fact the standard aEO strategy is near optimal in two of the tests, and is never worse than the optimal strategy by a factor of more than 2.

The expansion-ordering heuristics had a much bigger effect on ontology reasoning performance (than the rule-ordering heuristics). In the case of DOLCE and WineFood, almost any strategy that uses Sd or Dd in the SUB phase is near optimal. For GALEN, however, using Da in both phases gives by far the best results. This is again due to the characteristic structure of this ontology, and the fact that preferentially choosing concepts with low modal depth tends to reduce the size of the graph. Unfortunately, no one strategy is universally good (Da/Da is best for GALEN but worst for DOLCE and WineFood); currently, Sd/Dd is the default setting, as the majority of real life ontologies resemble DOLCE and WineFood more than GALEN), but this can of course be changed

⁵Top-down architectures necessarily give lowest priority to the $\exists$ -rule, and generally give low priority to $\sqcup$ -rule, which is why they work relatively well with ontologies.

by the user if it is known that the ontology to be reasoned with will have a GALEN-like structure.

For the DL'98 tests, the picture is again quite confused: the Sd strategy (the default in the SAT phase) is optimal in some tests, but bad in others—disastrously so in the case of the d4_n test. As in the ontology case, the only “solution” offered at present is to allow users to tune these settings according to the problem type or empirical results.

6 Discussion and Future Work

We have described the ToDo list architecture used in the FaCT++ system along with a range of heuristics that can be used for rule and expansion ordering. We have also presented an empirical analysis of these heuristics and shown how these have led us to select the default setting currently used by FaCT++.

These default settings reflect the current predominance of relatively small and simply structured ontologies. This may not, however, be a realistic picture of the kinds of ontology that we can expect in the future: many existing ontologies (including, e.g., WineFood) pre-date the development of OWL, and have been translated from less expressive formalisms. With more widespread use of OWL, and the increasing availability of sophisticated ontology development tools, it may be reasonable to expect the emergence of larger and more complex ontologies. As we have seen in Section 5.1, heuristics can be very effective in helping us to deal efficiently with such ontologies, but choosing a suitable heuristic becomes of critical importance.

In our existing implementation, changing heuristics requires the user to set the appropriate parameters when using the reasoner. This is clearly undesirable at best, and unrealistic for non-expert users. We are, therefore, working on techniques that will allow us to guess the most appropriate heuristics for a given ontology. The idea is to make an initial guess based on an analysis of the syntactic structure of the ontology (it should be quite easy to distinguish GALEN-like ontologies from DOLCE and WineFood-like ontologies simply by examining the statistics that have already been gathered for use in expansion-ordering heuristics), with subsequent adjustments being made based on the behaviour of the algorithm (e.g., the size of graphs being constructed).

Another limitation of the existing implementation is that a single strategy is used for all the tests performed in the classification process. In practice, the characteristics of different tests (e.g., w.r.t. concept size and/or satisfiability) may vary considerable, and it may make sense to dynamically switch heuristics depending on the kind of test being performed. This again depends on having an effective (and cheap) method for analysing the likely characteristics of a given test, and syntactic and behavioural analyses will also be investigated in this context.

References

- [Baader *et al.*, 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.
- [Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook*. CUP, 2003.
- [Calvanese *et al.*, 1998a] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, 1998.
- [Calvanese *et al.*, 1998b] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*, pages 2–13, 1998.
- [Carroll & De Roo, 2004] J. Carroll and J. De Roo. OWL web ontology language test cases. W3C Recommendation, 2004.
- [Freeman, 1995] J. W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.
- [Gangemi *et al.*, 2002] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In *Proc. of EKAW 2002*, 2002.
- [Giunchiglia & Sebastiani, 1996] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for *ALC*. In *Proc. of KR'96*, pages 304–314, 1996.
- [Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180, 1999.
- [Horrocks *et al.*, 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
- [Horrocks, 1997] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
- [Horrocks, 1998] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR'98*, pages 636–647, 1998.
- [Horrocks & Patel-Schneider, 1998] I. Horrocks and P. F. Patel-Schneider. DL systems comparison. In *Proc. of DL'98*, pages 55–57, 1998.
- [Horrocks & Patel-Schneider, 1999] I. Horrocks and P. F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.
- [Jeroslow & Wang, 1990] R. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Ann. of Mathematics and Artificial Intelligence*, 1:167–187, 1990.
- [Knublauch *et al.*, 2004] H. Knublauch, R. Ferguson, N. Noy, and M. Musen. The protégé OWL plugin: An open development environment for semantic web applications. In *Proc. of ISWC 2004*, 2004.
- [Massacci, 1999] F. Massacci. TANCS non classical system comparison. In *Proc. of TABLEAUX'99*, 1999.
- [McGuinness & Wright, 1998] D. L. McGuinness and J. R. Wright. An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, pages 69–77, 1998.
- [Rector, 2003] A. Rector. Description logics in medical informatics. In *The Description Logic Handbook*, pages 306–346. CUP, 2003.
- [Rogers *et al.*, 2001] J. E. Rogers, A. Roberts, W. D. Solomon, E. van der Haring, C. J. Wroe, P. E. Zanstra, and A. L. Rector. GALEN ten years on: Tasks and supporting tools. In *Proc. of MEDINFO2001*, pages 256–260, 2001.
- [Schmidt-Schauß & Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

Going Far, Logically

Fusun Yaman and Dana Nau and V.S. Subrahmanian

Department of Computer Science,
University of Maryland Institute for Advanced Computer Studies (UMIACS),
and Institute for Systems Research (ISR)
University of Maryland
College Park, MD, 20742
{fusun, nau, vs}@cs.umd.edu

Abstract

There are numerous applications where we need to ensure that multiple moving objects are sufficiently far apart. Furthermore, in many moving object domains, there is positional indeterminacy — we are not 100% sure exactly when a given moving object will be at a given location. [Yaman *et al.*, 2004] provided a logic of motion but did not provide algorithms to ensure that moving objects are kept sufficiently far apart. In this paper, we extend their logic to include a “far” predicate. We develop the **CheckFar** algorithm that checks if any given two objects will always be sufficiently far apart at during a time interval. We have run a set of experiments showing that our **CheckFar** algorithm scales very well.

1 Introduction

All of us fly in airplanes. We know that flight plans are almost never 100% accurate. We all have a vested interest in ensuring that planes we fly in are sufficiently far from other planes that are flying in the sky at the same time. The goal of this paper is to develop a hybrid *logical and constraint based framework* to ensure that moving objects with *positional uncertainty* (where will the object be at a given time) remain sufficiently far apart.

Our work builds on [Yaman *et al.*, 2004] who developed the concept of a “go-theory”. A go-theory is a finite set of “go-atoms.” A go-atom can express statements such as “*Plane p1 leaves location L1 some time between 10 and 12 and arrives at location L2 at some time between 30 and 40 and during the flight the speed of the plane is between 5 and 6*”. They provide algorithms to check if a given plane is within a given region at a given time point. They also introduce ground atoms of the form $\text{near}(o_1, o_2, d, t_1, t_2)$ — intuitively, this means that at all times during the interval t_1, t_2 , o_1 and o_2 are guaranteed to be within d units of each other.

In this paper, we introduce a predicate symbol called $\text{far}(o_1, o_2, d, t_1, t_2)$ — intuitively, this means that at all times during the interval t_1, t_2 , o_1 and o_2 are guaranteed to be at least d units apart.

One may think that $\text{near}()$ and $\text{far}()$ are complements of each other. Unfortunately, this is not true. Yaman *et al.* [Ya-

man *et al.*, 2004] define interpretations and a notion of satisfaction. For $\text{near}(o_1, o_2, d, t_1, t_2)$ atom to be a logical consequence of a go-theory G , for every interpretation $\mathcal{I}$ that satisfies G , it must be the case that for all times $t_1 \leq t \leq t_2$, the distance between the locations of o_1 and o_2 at time t is less than or equal to d . Thus, $\neg \text{near}((o_1, o_2, d, t_1, t_2))$ is a logical consequence of G if for every interpretation $\mathcal{I}$ which satisfies G there exists a time $t_1 \leq t \leq t_2$, the distance between the locations of o_1 and o_2 at time t (according to the interpretation $\mathcal{I}$) is greater than d . In contrast, for $\text{far}(o_1, o_2, d, t_1, t_2)$ to be entailed by S , it must be the case that for every interpretation $\mathcal{I}$ which satisfies G and for every time point $t_1 \leq t \leq t_2$, the distance between the locations of o_1 and o_2 at time t is greater than d . Thus, entailment of $\text{far}()$ atoms is not the same as entailment of either $\text{near}()$ atoms or $\neg \text{near}()$ literals.

In this paper, we define the semantics of $\text{far}()$ and develop an algorithm called **CheckFar** to check entailment of ground $\text{far}()$ atoms by a go-theory G . We have conducted extensive experiments on the computational feasibility of **CheckFar** — our experiments show in a compelling way that in real world situations¹, **CheckFar** will work very well indeed. In our experiments we answered far queries in less than 0.6 seconds for go theories up to 1000 atoms per object.

2 Background On go-Theories

We now provide a quick overview of go-theories from [Yaman *et al.*, 2004]. We assume the existence of several sets of constant symbols: $\mathbf{R}$ is the set of all real numbers, $\mathbf{O}$ is the set of names of objects, $\mathbf{P} = \mathbf{R} \times \mathbf{R}$ is the set of all points in two-dimensional cartesian space. We assume the existence of three disjoint sets of variable symbols, $V_{\mathbf{R}}$, $V_{\mathbf{O}}$, and $V_{\mathbf{P}}$, ranging over $\mathbf{R}$, $\mathbf{O}$ and $\mathbf{P}$, respectively. A *real* term t is any member of $\mathbf{R} \cup V_{\mathbf{R}}$. Object terms and point terms are defined similarly. Ground terms are defined in the usual way. We now define atoms as follows.

- If o_1, o_2 are object terms, and d, t_1, t_2 are positive real terms, then $\text{near}(o_1, o_2, d, t_1, t_2)$ is an *atom*. Intuitively, this atom says that o_1, o_2 are within distance d of each other during the time interval $[t_1, t_2]$.
- If o is an object term, P_1, P_2 are point terms, and t_1, t_2 are positive real terms, then $\text{in}(o, P_1, P_2, t_1, t_2)$ is an

¹We built an application to manage separation between ships in port shipping lanes.

atom. Intuitively, this atom says that object o is in the rectangle whose lower left (resp. upper right) corner is P_1 (resp. P_2) at some point in the time interval $[t_1, t_2]$.

- If o is an object term, P_1, P_2 are point terms, and $t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+$ are positive real terms, then $\text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ is an atom called a *go atom*. Intuitively, this atom says that object o leaves point P_1 at some time in $[t_1^-, t_1^+]$ and arrives at point P_2 during $[t_2^-, t_2^+]$, traveling in a straight line with a minimum speed v^- and maximum speed v^+ .

Ground atoms are defined in the usual way. A *go theory* is a finite set of *ground go-atoms*. Note that *go-theories* do not contain *near()* or *in()* atoms.

Notation. If $g = \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$, then,

$$\begin{aligned} \text{obj}(g) &= o, & v^-(g) &= v^-, & v^+(g) &= v^+, \\ \text{loc}_1(g) &= P_1, & t_1^-(g) &= t_1^-, & t_1^+(g) &= t_1^+, \\ \text{loc}_2(g) &= P_2, & t_2^-(g) &= t_2^-, & t_2^+(g) &= t_2^+. \end{aligned}$$

If G is a go-theory and o is an object id, the *restriction of G to o* , denoted G^o is the set $\{g \in G \mid \text{obj}(g) = o\}$.

An *interpretation* is a continuous function $\mathcal{I} : \mathbf{O} \times \mathbf{R}^+ \rightarrow \mathbf{P}$. Intuitively, $\mathcal{I}(o, t)$ is o 's location at time t .

Definition 1 Let $g = \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ be a ground atom and $\mathcal{I}$ be an interpretation. $\mathcal{I}$ *satisfies* g w.r.t. a time interval $T = [t_1, t_2]$ iff:

- $t_1 \in [t_1^-, t_1^+]$ and $\mathcal{I}(o, t_1) = P_1$
- $t_2 \in [t_2^-, t_2^+]$ and $\mathcal{I}(o, t_2) = P_2$
- $\forall t \in [t_1, t_2]$, $\mathcal{I}(o, t)$ is on the line segment $[P_1, P_2]$
- $\forall t, t' \in [t_1, t_2]$, $t < t'$ implies $\text{dist}(\mathcal{I}(o, t), P_1) < \text{dist}(\mathcal{I}(o, t'), P_1)$, where dist is the function that computes the Euclidean distance between two points.
- For all but finitely many times in $[t_1, t_2]$, $v = d(|\mathcal{I}(o, t)|)/dt$ is defined and $v^-(g) \leq v \leq v^+(g)$.

This intuitively says that $\mathcal{I} \models g$ w.r.t. a time interval $T = [t_1, t_2]$ iff o starts moving at t_1 and stops moving at t_2 and furthermore, during $[t_1, t_2]$, the object moves away from P_1 towards P_2 without either stopping or turning back or wandering away from the straight line connecting P_1 and P_2 . We are now ready to define the full concept of satisfaction.

$\mathcal{I}$ *satisfies* a ground literal (denoted $\mathcal{I} \models A$) in these cases:

1. $\mathcal{I} \models \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ iff there exists an interval $[t_1, t_2]$ such that $\mathcal{I} \models A$ w.r.t. $[t_1, t_2]$.
2. $\mathcal{I} \models \text{near}(o_1, o_2, d, t_1, t_2)$ iff $\text{dist}(\mathcal{I}(o_1, t), \mathcal{I}(o_2, t)) \leq d$ for all $t_1 \leq t \leq t_2$.
3. $\mathcal{I} \models \text{in}(o, P_1, P_2, t_1, t_2)$ iff there are numbers $t \in [t_1, t_2]$, $x \in [P_1^x, P_2^x]$ and $y \in [P_1^y, P_2^y]$ such that $\mathcal{I}(o, t) = (x, y)$.
4. $\mathcal{I} \models \neg A$ iff $\mathcal{I}$ does not satisfy A .

The above definition can be extended in the obvious way to handle quantification — in this paper, we will only consider the ground case.

$\mathcal{I}$ *satisfies* (or *is a model of*) a set of ground atoms MT iff $\mathcal{I}$ satisfies every $A \in \text{MT}$. MT is *consistent* iff there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \text{MT}$. L is a *logical consequence*

of MT , denoted $\text{MT} \models L$, iff every model of MT is also a model of L .

3 far() atoms

We are now ready to extend the logic in [Yaman *et al.*, 2004] to include *far()* atoms. If o_1, o_2 are objects, and d, t_1, t_2 are real terms, then $\text{far}(o_1, o_2, d, t_1, t_2)$ is a *far()* atom.

Definition 2 (satisfaction of far() atoms) Suppose f is a ground *far* atom and $\mathcal{I}$ is an interpretation. $\mathcal{I} \models f$ iff for all $t_1 \leq t \leq t_2$, $\text{dist}(\mathcal{I}(o_1, t), \mathcal{I}(o_2, t)) > d$.

We say a go theory G *entails* f iff whenever $\mathcal{I} \models G$, it is also the case that $\mathcal{I} \models f$. The *far()-entailment problem* is that of checking whether a given go-theory entails a *ground far()*-atom. The entailment of a negated *near* literal does not imply entailment of the associated *far* atom. Similarly entailment of a negated *far* literal does not imply entailment of the associated *near* atom. However, the following results hold:

Lemma 1 Suppose G is a go theory and suppose $\text{far}(o_1, o_2, d, t_1, t_2)$ and $\text{near}(o_1, o_2, d, t_1, t_2)$ are ground. If $G \models \text{far}(o_1, o_2, d, t_1, t_2)$, then $G \models \neg \text{near}(o_1, o_2, d, t_1, t_2)$. If $G \models \text{near}(o_1, o_2, d, t_1, t_2)$, then $G \models \neg \text{far}(o_1, o_2, d, t_1, t_2)$.

The following result tells us that checking for entailment of a ground *far()*-atom is co-NP complete.

Theorem 1 Let G be a go theory and $f = \text{far}(o, o', d, t_1, t_2)$ be a ground atom. Checking if $G \models f$ is coNP-complete.

We will omit the proofs due to space constraints.

4 Far Algorithm

Finding an algorithm to solve the *far()*-entailment problem is a complex task. Our solution involves the following steps. First we partition a go-theory into clusters G^o of go-atoms about the same object o . Next, for any given go-atom g , we define the *temporal certainty interval* which specifies a time interval when the object is guaranteed to be on the line segment $[\text{loc}_1(g), \text{loc}_2(g)]$. Third, we define the *positional certainty interval* that finds the smallest subsegment of the above line in which the vehicle is guaranteed to be at time t . Both temporal and positional certainty intervals are defined w.r.t. a given ordering of go-atoms. Our fourth step is to explain how to solve the *far()*-entailment problem when a go-theory has only two go-atoms (one for each object) and $t_1 = t_2$ in the *far()* atom whose entailment we are trying to check. Our fifth step generalizes this to the case when $t_1 \leq t_2$. Finally, we show how to remove the assumption that G contains only two go-atoms.

Due to space constraints, we will proceed under the assumption that G is non-collinear as defined below. Extending our algorithm to remove this assumption is straightforward². Our implementations do not require this assumption.

²[Yaman *et al.*, 2004] provides a technique to merge multiple collinear go-atoms into “movements” that are non-collinear. The same technique can be used here.

Definition 3 A go theory G is non-collinear iff there are no go-atoms $g, g' \in G$ such that $obj(g) = obj(g')$ and the following two conditions both hold:

- The intersection of line segments $[loc_1(g), loc_2(g)]$ and $[loc_1(g'), loc_2(g')]$ is a line segment $[P, Q]$ such that P is visited before Q in both g and g' and
- $t_1^-(g) \leq t_2^+(g')$ and $t_1^-(g') \leq t_2^+(g)$.

4.1 Temporal and Positional Certainty Intervals

In this section, we define the notions of temporal and positional uncertainty.

Definition 4 ($\mathcal{C}(G, o, \sqsubseteq)$) Let G be a go theory and o be an object id. Let $\sqsubseteq$ be any total ordering on G^o . The set $\mathcal{C}(G, o, \sqsubseteq)$ of linear constraints associated with $G, o, \sqsubseteq$ is defined as follows:

- $\forall g = go(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+) \in G^o$
 - $t_1^- \leq S_g \leq t_1^+$ and $t_2^- \leq E_g \leq t_2^+$
 - $v^- \times (E_g - S_g) \leq \text{dist}(P_1, P_2) \leq v^+ \times (E_g - S_g)$
- for every $g, g' \in G^o$ such that $g \sqsubseteq g'$: $E_g \leq S_{g'}$.

S_g and E_g are variables associated with an atom g which intuitively denote the start time and end time of the movement described by g . When G^o has only one atom g , we will use $\mathcal{C}(g)$ as a short hand notation.

Example 1 Let $g = go(o, (40, 10), (70, 50), 12, 13, 21, 21, 4, 10)$ and $g' = go(o, (70, 50), (30, 80), 20, 21, 30, 31, 4, 10)$. Suppose $G = \{g, g'\}$ and $g \sqsubseteq g'$. Then $\mathcal{C}(G, o, \sqsubseteq)$ contains the following constraints:

- $12 \leq S_g \leq 13, 21 \leq E_g \leq 21$,
- $4 \times (E_g - S_g) \leq 50 \leq 10 \times (E_g - S_g)$.
- $20 \leq S_{g'} \leq 21, 30 \leq E_{g'} \leq 31$,
- $4 \times (E_{g'} - S_{g'}) \leq 50 \leq 10 \times (E_{g'} - S_{g'})$,
- $E_g \leq S_{g'}$.

We now use linear programming methods to define the temporal certainty interval of a go atom.

Definition 5 ($TCI(G, o, \sqsubseteq, g)$) Let G be a go theory, o be an object, and $\sqsubseteq$ be a total order on G^o such that $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. The **temporal certainty interval**, $TCI(G, o, \sqsubseteq, g)$, of an atom $g \in G^o$ w.r.t. $\sqsubseteq$ is the time interval $[T^-, T^+]$ where

- T^- is the result of solving the linear program: “**maximize** S_g **subject to** $\mathcal{C}(G, o, \sqsubseteq)$ ”,
- T^+ is the result of solving the linear program: “**minimize** E_g **subject to** $\mathcal{C}(G, o, \sqsubseteq)$ ”.

$TCI(G, o, \sqsubseteq, g)$ is undefined if $T^- > T^+$.

When G^o has only one atom g we will $TCI(g)$ as a short hand notation. Intuitively, $TCI(G, o, \sqsubseteq, g)$ is the interval when we know for sure that object o is within the line specified in go-atom g .

Example 2 Let g and g' be the two go atoms in Example 1. $TCI(g)$ is $[13, 21]$. If $G = \{g, g'\}$ and $g \sqsubseteq g'$, then $TCI(G, o, \sqsubseteq, g') = [21, 30]$.

Conversely, given a time point t and a ground go-atom g , we wish to know the potential segment on the line connecting $loc_1(g)$ and $loc_2(g)$ where the object associated with g could possibly be. To find this, we need another linear program.

Definition 6 ($PCI(G, o, \sqsubseteq, g, t)$) Let G be a go theory, o be an object, and $\sqsubseteq$ be a total order on atoms of G^o such that $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. Let g be an atom in G^o such that $TCI(G, o, \sqsubseteq, g)$ is defined and t be any time point in $TCI(G, o, \sqsubseteq, g)$. The **positional certainty interval**, $PCI(G, o, \sqsubseteq, g, t)$, of atom g at time t w.r.t. ordering $\sqsubseteq$ is defined as follows:

- Let $C(g, t)$ be the set of constraints:

$$\begin{aligned} -\frac{d_{g,t}}{v^+(g)} &\leq t - S_g \leq \frac{d_{g,t}}{v^-(g)} \\ -\frac{D-d_{g,t}}{v^+(g)} &\leq E_g - t \leq \frac{D-d_{g,t}}{v^-(g)} \end{aligned}$$

where $D = \text{dist}(loc_1(g), loc_2(g))$. Note that this is a linear program when g is fixed because many terms above (such as $D, v^-(g), v^+(g)$ are constants).

- Let $d_{g,t}^{min}$ be the result of solving the linear program

minimize $d_{g,t}$ **subject to** $\mathcal{C}(G, o, \sqsubseteq) \cup C(g, t)$.

- Let $d_{g,t}^{max}$ be obtained by maximizing the objective function in the above linear program (instead of minimizing).
- Let $P^-(G, o, \sqsubseteq, g, t)$ be the point exactly $d_{g,t}^{min}$ units away from $loc_1(g)$ on the line connecting $loc_1(g), loc_2(g)$ and likewise, let $P^+(G, o, \sqsubseteq, g, t)$ be the point exactly $d_{g,t}^{max}$ units away from $loc_1(g)$ on the same line.

The **positional certainty interval** $PCI(G, o, \sqsubseteq, g, t)$ is the line segment from $P^-(G, o, \sqsubseteq, g, t)$ to $P^+(G, o, \sqsubseteq, g, t)$.

When G^o has only one atom g we use the shorter notation $PCI(g, t)$.

Lemma 2 Let g be a go atom such that $TCI(g)$ is defined, $\mathcal{I}$ be an interpretation, and suppose $t \in TCI(g)$. If $\mathcal{I} \models g$, then $\mathcal{I}(o, t)$ is on the line segment $PCI(g, t)$.

4.2 Binary go-theories about a single time point

A binary go-theory is one which contains two go-atoms $G = \{g, g'\}$ where $obj(g) = o, obj(g') = o'$. Consider a ground $\text{far}()$ query $\text{far}(o, o', t_1, t_2, d)$ where $t_1 = t_2$. We need to check if the distance between o and o' is guaranteed to be greater than d at time t . The following lemma presents necessary and sufficient conditions for entailment of such queries under the above assumptions.

Lemma 3 Let $G = \{g, g'\}$ be a go-theory such that $obj(g) = o$ and $obj(g') = o'$ and let $f = \text{far}(o, o', d, t, t)$ be a ground atom. $G \models f$ iff

- $t \in TCI(g)$ and $t \in TCI(g')$ and
- The minimum distance between line segments $PCI(g, t)$ and $PCI(g', t)$ is greater than d .

Example 3 Let $g = go(o, (40, 10), (70, 50), 12, 13, 21, 21, 4, 10)$ and $g' = go(o', (55, 20), (45, 80), 17, 18, 32, 33, 2, 6)$ be two atoms. Let $G = \{g, g'\}$ and $f = \text{far}(o, o', 5, 19, 19)$. $TCI(g) = [13, 21]$ and $TCI(g') = [18, 32]$ both include the

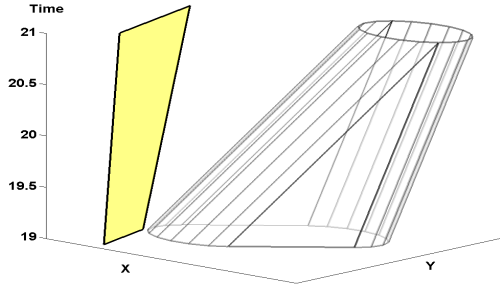


Figure 1: The polygon on the left is the space envelope of g' during $[19, 21]$ and the volume on the right is 5-neighbourhood of g during time interval $[19, 21]$. g and g' are as defined in Examples 4, 5 and 6.

time point 19. At time 19, o is somewhere on the line segment $PCI(g, 19) = [(58, 34), (65.2, 43.6)]$ and object o' is on the line segment $PCI(g', 19) = [(54.67, 21.97), (53.08, 31.84)]$. The minimum distance between these two lines is 33.73 which is greater than 5 so $G \models f$.

Consider the atom $\text{near}(o, o', d, t, t)$ instead of $\text{far}(o, o', d, t, t)$ then the second bullet of Lemma 3 becomes: “The **maximum** distance between line segments $PCI(g, t)$ and $PCI(g', t)$ is **less than or equal to** d ”. This maximum distance is achieved at the end points of $PCI(g, t)$ and $PCI(g', t)$, e.g. $P^-(g, t)$ and $P^+(g', t)$. However the minimum distance between $PCI(g, t)$ and $PCI(g', t)$ is not necessarily at the end points hence its computation is more complex.

4.3 Binary go-theories with temporal intervals

The complexity of computing far queries gets magnified even more when we consider the case $t_1 \leq t_2$. For the near atom this is easy because it is enough to check the distance at the end points of $PCI(g, t)$ and as shown in [Yaman et al., 2004] end points of $PCI(g, t)$ are piecewise linear functions over a time interval. This is not enough for the far atom. For this reason answering far queries over time intervals requires a different approach than the one in [Yaman et al., 2004].

We first define the *space envelope* of a go-atom. Intuitively, the space envelope of a go-atom g is the set of all (x, y, t) -triples such that there exists a model $\mathcal{I}$ of g in which $\mathcal{I}(\text{obj}(g), t) = (x, y)$. In other words, it defines where and when it is possible for object o to be.

Definition 7 Let g be a ground go-atom such that $TCI(g)$ is defined and let $T = [t_1, t_2]$ be any time interval such that $T \subseteq TCI(g)$. The **space envelope**, $SE(g, T)$ of g during interval T is $\{(x, y, t) \mid t \in T \text{ and } (x, y) \in PCI(g, t)\}$.

Theorem 2 $SE(g, T)$ is a convex set.

Example 4 Let $g' = \text{go}(o', (55, 20), (45, 80), 17, 18, 32, 33, 2, 6)$ be a go atom. The space envelope of g' over time interval $[19, 21]$, is shown in Figure 1 on the left side. It is easy to see that $SE(g', [19, 21])$ is convex.

We now define the set of points that are closer than a given distance d to any possible location of an object at a given time.

Definition 8 (d -neighbourhood) Let g be a ground go-atom such that $TCI(g)$ is defined. Let d be a real number and let $T = [t_1, t_2]$ be a time interval such that $T \subseteq TCI(g)$. The **d -neighbourhood** of g during T , denoted $Nbr(g, T, d) = \{(x, y, t) \mid t_1 \leq t \leq t_2 \text{ and } (x, y) \in \text{NearPts}(g, t, d)\}$ where $\text{NearPts}(g, t, d) = \{(x, y) \mid \exists (x', y') \in PCI(g, t) \text{ and } \text{dist}((x, y), (x', y')) \leq d\}$.

Intuitively $\text{NearPts}(g, t, d)$ is the set of all points p such that all points on the line segment $PCI(g, t)$ which are d units or less in distance from p at time t . Similarly $Nbr(g, T, d)$ is the set of all points (x, y, t) such that it is possible for $\text{obj}(g)$ to be within d units of (x, y) at some time t in interval T .

Theorem 3 (i) $Nbr(g, T, d)$ is a convex set. (ii) If $d = 0$, then $Nbr(g, T, d) = SE(g, T)$.

Example 5 Let $g = \text{go}(o, (40, 10), (70, 50), 12, 13, 19, 21, 4, 10)$ be a go atom. The 5-neighborhood of g over time interval $[19, 21]$ is shown on the far right hand side of Figure 1. It is easy to see that it is a convex set.

The following theorem states necessary and sufficient conditions under which $\{g, g'\} \models f$.

Theorem 4 Let $f = \text{far}(o, o', d, t_1, t_2)$ and $G = \{g, g'\}$ be a go theory where $\text{obj}(g) = o$ and $\text{obj}(g') = o'$. $G \models f$ iff

- $[t_1, t_2] \subseteq TCI(g)$ and $[t_1, t_2] \subseteq TCI(g')$
- $Nbr(g, [t_1, t_2], d) \cap SE(g', [t_1, t_2]) = \emptyset$

Thus, an algorithm to solve the far()-entailment problem only needs to check both these conditions. [Mantyla, 1988] provides polynomial algorithms to check for the intersection of two convex sets — these can be used directly to check the second condition above.

Example 6 Let $g = \text{go}(o, (40, 10), (70, 50), 12, 13, 21, 21, 4, 10)$ and $g' = \text{go}(o', (55, 20), (45, 80), 17, 18, 32, 33, 2, 6)$ be two go atoms. Let $G = \{g, g'\}$ and $f = \text{far}(o, o', 5, 19, 21)$. Then $TCI(g) = [13, 21]$ and $TCI(g') = [18, 32]$. Both include the time interval $[19, 21]$. Figure 1 shows $Nbr(g, [19, 21], 5)$ and $SE(g', [19, 21])$. It is apparent from the figure that the two do not intersect: hence $G \models f$.

4.4 Arbitrary Go-theories with temporal intervals

We now remove the restriction that G is a binary go-theory. Doing so introduces several complications. For any single o , there may be many possible total orderings associated with G^o . Furthermore, for some theories we can predict the possible locations of an object not only in the certainty interval of an atom, but also during a time interval that spans over several atoms. The following lemma gives the necessary and sufficient conditions when this can be done.

Lemma 4 Let G be a go-theory, o be an object, and $\sqsubseteq$ be a total order on G^o such that $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. Suppose $g_1 \sqsubseteq g_2 \dots \sqsubseteq g_n$ are the atoms of G^o . $TCI(G, o, \sqsubseteq, g_i) \cup TCI(G, o, \sqsubseteq, g_{i+1}) \dots \cup TCI(G, o, \sqsubseteq, g_j)$ is a single time interval iff for every $1 \leq i \leq k < j \leq n$ with $TCI(G, o, \sqsubseteq, g_k) = [T_k^-, T_k^+]$ the following are true

- $T_k^+ = T_{k+1}^-$
- $\text{loc}_2(g_k) = \text{loc}_1(g_{k+1})$

Definition 9 Let G be a go-theory, o be an object, and $\sqsubseteq$ be a total order on G^o such that $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. Suppose $S \subseteq G^o$. S is **temporally relevant** to time interval $[t_1, t_2]$ is iff

- The atoms in S satisfy the conditions in lemma 4 and
- There are atoms g, g' in S such that $t_1 \in TCI(G, o, \sqsubseteq, g)$ and $t_2 \in TCI(G, o, \sqsubseteq, g')$

Example 7 Let $g_1 = \text{go}(o, (40, 10), (70, 50), 12, 13, 21, 21, 4, 10)$ and $g_2 = \text{go}(o, (70, 50), (30, 80), 20, 21, 30, 31, 4, 10)$ be go atoms. If $G = \{g_1, g_2\}$ and $g_1 \sqsubseteq g_2$ then $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. Moreover $TCI(G, o, \sqsubseteq, g_1) = [13, 21]$, $TCI(G, o, \sqsubseteq, g_2) = [21, 30]$ and $TCI(G, o, \sqsubseteq, g_1) \cup TCI(G, o, \sqsubseteq, g_2)$ is a single interval. Suppose $T = [19, 30]$ then $\{g_1, g_2\}$ is temporally relevant to T .

We now generalize definitions 7 and 8 to accommodate non-binary go-theories.

Definition 10 ($SE(G, o, \sqsubseteq, T)$) Let G be a go-theory, o be an object, and $\sqsubseteq$ be a total order on G^o such that $\mathcal{C}(G, o, \sqsubseteq)$ has a solution. If $T = [t_1, t_2]$ is a time interval, then $SE(G, o, \sqsubseteq, T)$ is the set of all points (x, y, t) such that

- $t \in T$ and $t \in TCI(G, o, \sqsubseteq, g)$ for some $g \in G^o$,
- (x, y) is on $PCI(G, o, \sqsubseteq, g, t)$.

$SE(G, o, \sqsubseteq, T)$ is not defined if there is a time point $t \in T$ such that for all $g \in G^o$, $t \notin TCI(G, \sqsubseteq, g)$.

Note that $SE(G, o, \sqsubseteq, T)$ is not necessarily convex when T spans over multiple go atoms. We can generalize Nbr to $Nbr(G, o, \sqsubseteq, T, d)$ in a similar manner — these are omitted due to space constraints. $Nbr(G, \sqsubseteq, T, d)$ also may not be convex. We now state a theorem describing the conditions under which a go theory G entails a far() atom.

Theorem 5 Suppose G is a go-theory and o, o' are objects. The ground atom $f = \text{far}(o, o', d, t_1, t_2)$ is a logical consequence of G iff for every pair of total orders $\sqsubseteq$ and $\sqsubseteq'$ on G^o and $G^{o'}$ such that $\mathcal{C}(G, o, \sqsubseteq)$ and $\mathcal{C}(G, o', \sqsubseteq')$ are solvable:

- $\exists S \subseteq G^o$ such that S is temporally relevant to $[t_1, t_2]$
- $\exists S' \subseteq G^{o'}$ such that S' is temporally relevant to $[t_1, t_2]$
- $Nbr(G, o, \sqsubseteq, [t_1, t_2], d) \cap SE(G, o', \sqsubseteq', [t_1, t_2]) = \emptyset$.

Example 8 Let g_1 and g_2 be go atoms in Example 7 and $g' = \text{go}(o', (55, 20), (45, 80), 17, 18, 32, 33, 2, 6)$. Let $G = \{g_1, g_2, g'\}$ and $f = \text{far}(o, o', 5, 19, 30)$. If $g_1 \sqsubseteq g_2$ and $\sqsubseteq' = \emptyset$ then Figure 2 shows $Nbr(G, o, \sqsubseteq, [19, 30], 5)$ and $SE(G, o', \sqsubseteq', [19, 30])$. The figure also shows that the sets intersect - hence $G \not\models f$.

As $Nbr(G, o, \sqsubseteq, T, d)$ and $SE(G, o', \sqsubseteq', T)$ are not always convex, computing their intersection is tricky. We partition T into subintervals $T_1, T_2, \dots, T_n$ (we call this a **convex partition of T**) such that for all $i < n$, the end point of T_i is the end point of the temporal certainty interval of some $g \in G^o \cup G^{o'}$. It is easy to verify that $n \leq |G^o| + |G^{o'}|$. For each T_i , $Nbr(G, o, \sqsubseteq, T_i, d)$ and $SE(G, o', \sqsubseteq', T_i)$ are convex and we leverage this in the CheckFar algorithm below.

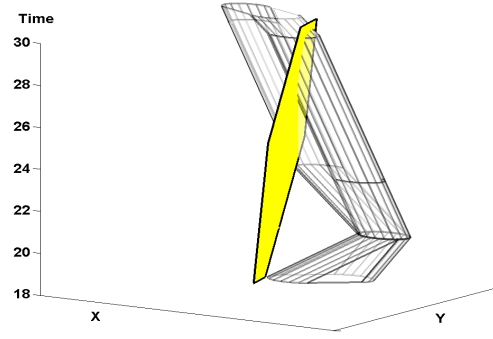


Figure 2: The space envelope of o' with respect to G and 5-neighbourhood of o w.r.t. G and $\sqsubseteq$ during time interval $[19, 30]$. G and $\sqsubseteq$ are as defined in Example 8

Algorithm CheckFar($G, \sqsubseteq, \sqsubseteq', f$)

```

Suppose  $f = \text{far}(o, o', d, t_1, t_2)$ ;
if  $\mathcal{C}(G, o, \sqsubseteq)$  or  $\mathcal{C}(G, o', \sqsubseteq')$  has no solution then return true
Let  $S_o \subseteq G^o$  such that  $S_o$  is temporally relevant to  $[t_1, t_2]$ 
Let  $S_{o'} \subseteq G^{o'}$  such that  $S_{o'}$  is temporally relevant to  $[t_1, t_2]$ 
if no such  $S_o$  or  $S_{o'}$  exists then return false
Let  $T_1, T_2, \dots, T_n$  be the convex partition of  $[t_1, t_2]$ 
for each  $i < n$  do
  if  $Nbr(G, o, \sqsubseteq, T_i, d) \cap SE(G, o', \sqsubseteq', T_i) \neq \emptyset$ 
    then return false
end for
return true

```

Theorem 6 Suppose G is a go-theory and $f = \text{far}(o, o', d, t_1, t_2)$ is a ground atom. Then: f is entailed by G iff for every total order $\sqsubseteq$ and $\sqsubseteq'$ on G^o and $G^{o'}$, the algorithm CheckFar($G, \sqsubseteq, \sqsubseteq', f$) returns “true”.

5 Implementation

We have implemented the CheckFar algorithm in Matlab. Our test results show that CheckFar($G, \sqsubseteq, \sqsubseteq', f$) runs in linear time. Note that a single run of CheckFar does not check if $G \models f$. In fact CheckFar($G, \sqsubseteq, \sqsubseteq', f$) should be executed for every possible $\sqsubseteq, \sqsubseteq'$ pair that produce a solvable set of constraints. Although in theory this number is exponential which leads to coNP-hardness, in practice we expect this number to be bounded by a manageable constant k . In general individual movements can have some uncertainty however the order in which they are going to be accomplished is usually well known. Figure 3 shows the running time of 3 far queries as we vary the number of atoms per object when $k = 256$. Each data point is an average of 300 runs. For these experiments we created go theories G with at most 16 total orderings and all atoms are in a rectangle of size 300 by 400. We queried those theories using three query templates $Q1 = \text{far}(o, o', 1, t_1, t_2)$, $Q2 = \text{far}(o, o', 50, t_1, t_2)$ and $Q3 = \text{far}(o, o', 10000, t_1, t_2)$ where $[t_1, t_2]$ is a random time interval with length 10. The reader will notice that as the distance increases, the time to compute the query decreases. In the case of Q3 which is trivially false it takes almost no time to return the answer. Queries Q1 and Q2 take at most 0.6 seconds for go theories with 1000 atoms per object.

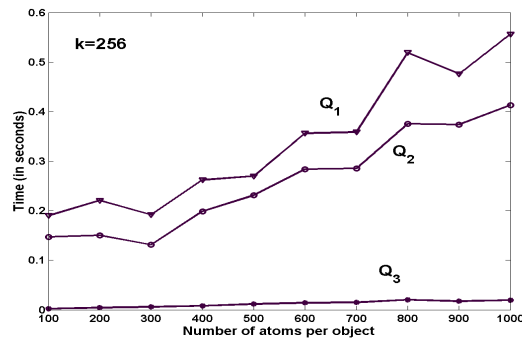


Figure 3: Time to compute three far queries Q1, Q2 and Q3 when $k = 256$.

6 Related Work

To our knowledge, there is no work on a logical foundation for checking separation between moving objects in the presence of temporal and spatial uncertainty. However, almost all models of moving entities like cars, airplanes, ships, humans and animals are subject to uncertainty about where they are at a given point in time.

[Gabelaia *et al.*, 2003; Merz *et al.*, 2003; Wolter and Zakharyashev, ; Cohn *et al.*, 2003] study spatio-temporal logics where time is a discrete sequence rather than being continuous. [Muller,] describes a formal theory for reasoning about motion in a qualitative framework that supports complex motion classes. The work is purely symbolic. [Shanahan, 1995] discusses the frame problem, when constructing a logic-based calculus for reasoning about the movement of objects in a real-valued co-ordinate system. [Rajagopalan and Kuipers, 1994] focuses on relative position and orientation of objects with existing methods for qualitative reasoning in a Newtonian framework. The focus of these works is qualitative - in contrast our work is heavily continuous and rooted in a mix of geometry and logic rather in just logic alone. `far()` queries are not studied in any of these papers.

There is also some peripherally related work in computer graphics [Lin and Gottschalk, 1998] and in “moving object databases” [Erwig *et al.*, 1999] – they provide no model theory or algorithms for `far()` queries nor allow any uncertainty.

7 Conclusions

The ability to query a large number of objects moving uncertainly in time and space so that separation constraints between objects are maintained is critical to many applications (e.g. air traffic control, shipping port lane management). In this paper, we have extended the “logic of motion” proposed by Yaman *et al.* [Yaman *et al.*, 2004] via a `far()` predicate. We develop a model theory for `far()`-entailment. We show that the `far()`-entailment problem is coNP-complete. We develop an efficient algorithm called **CheckFar** that effectively checks if a given total ordering of atoms in a go-theory preserve separation constraints. To check whether two objects are guaranteed to be at least d units of distance away from each other during an interval of time, we must apply **CheckFar** to different possible orderings of the atoms in the go-

theory. In real-world applications such as air traffic control and a shipping port lane management we have built, we expect such orderings to be relatively small in number even if the number of objects is very large. The reason for this is that usually the flight plan for each airplane consists of a sequence of steps which only yields 1 ordering of the atoms per airplane (whereas our experiments have used 16). Our experiments show that **CheckFar** performs very well in practice.

Acknowledgements

This work was supported in part by ARO grant DAAD190310202, ARL grants DAAD190320026 and DAAL0197K0135, the ARL CTAs on Telecommunications and Advanced Decision Architectures, NSF grants IIS0329851 and IIS0412812 and 0205489 and UC Berkeley contract number SA451832441 (subcontract from DARPA’s REAL program).

References

- [Cohn *et al.*, 2003] A. G. Cohn, D. Magee, A. Galata, D. Hogg, and S. Hazarika. Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In *Spatial Cognition III*. 2003.
- [Erwig *et al.*, 1999] M. Erwig, R. H. Güting, M. Schneider, and M. Vazirgiannis. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296, 1999.
- [Gabelaia *et al.*, 2003] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyashev. On the computational complexity of spatio-temporal logics. In *FLAIRS Conference*, pages 460–464, 2003.
- [Lin and Gottschalk, 1998] M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, 1998.
- [Mantyla, 1988] M. Mantyla. *Introduction to Solid Modeling*. W. H. Freeman and Co., 1988.
- [Merz *et al.*, 2003] S. Merz, J. Zappe, and M. Wirsing. A spatio-temporal logic for the specification and refinement of mobile systems. In *Fund. Appr. to Software Engr.*, pages 87–101, 2003.
- [Muller,] P. Muller. A qualitative theory of motion based on spatio-temporal primitives. In *KR98*, pages 131–141.
- [Rajagopalan and Kuipers, 1994] R. Rajagopalan and B. Kuipers. Qualitative spatial reasoning about objects in motion: Application to physics problem solving. In *Conf. on AI for Applications*, pages 238–245, 1994.
- [Shanahan, 1995] M. Shanahan. Default reasoning about spatial occupancy. *Artificial Intelligence*, 74(1):147–163, 1995.
- [Wolter and Zakharyashev,] F. Wolter and M. Zakharyashev. Spatio-temporal representation and reasoning based on RCC-8. In *KR2000*, pages 3–14.
- [Yaman *et al.*, 2004] F. Yaman, D. Nau, and V. S. Subrahmanian. A logic of motion. In *KR2004*, pages 85–94, 2004.

A Motion Closed World Assumption

Fusun Yaman and Dana Nau and V.S. Subrahmanian

Department of Computer Science,
University of Maryland Institute for Advanced Computer Studies (UMIACS),
and Institute for Systems Research (ISR)
University of Maryland
College Park, MD, 20742
{fusun, nau, vs}@cs.umd.edu

Abstract

Yaman et. al. [Yaman *et al.*, 2004] introduce “go theories” to reason about moving objects. In this paper, we show that this logic often does not allow us to infer that an object is *not* present at a given place or region, even though common sense would dictate that this is a reasonable inference to make. We define a class of models of go-theories called *coherent* models. We use this concept to define a *motion closed world assumption* (MCWA) and develop a notion of MCWA-entailment. We show that checking if a go-theory has a coherent model is NP-complete. An atom checks if a given object is present in a given region sometime in a given time interval. We provide sound and complete algorithms to check if a ground in literal (positive or negative in atom) can be inferred from a go-theory using the MCWA. In our experiments our algorithms answer such queries in less than 1 second when there are up to 1,000 go-atoms per object.

1 Introduction

Reasoning about moving objects is becoming increasingly important. Air traffic controllers in both the US and Europe are facing a dramatically increasing workload as the number of flights increases. Cell phone companies are increasingly interested in knowing where cell phones on their network are located — this is useful for hand-off policies between cell phone towers. Vehicle security systems such as LOJACK and ONSTAR are increasingly being used to determine where vehicles are and where they are not.

[Yaman *et al.*, 2004] proposed the concept of a “go theory” which can be used to make statements of the form “Object o is expected to leave location P_1 at some time point in the interval $[t_1^-, t_1^+]$ and reach location P_2 at some time point in the interval $[t_2^-, t_2^+]$ traveling at a velocity between v_1 and v_2 . Go theories can be used, for example, to make statements such as “Plane p22 is expected to take off from Paris at some time between 10 and 12 and land at Boston at some time between 18 and 23 traveling at a speed between 10 to 20.” Go theories are sets of such statements. Figure 1 shows the spatial layout of one such go-theory (the go theory is written in text at the top). [Yaman *et al.*, 2004] provides a model theory for go theories,

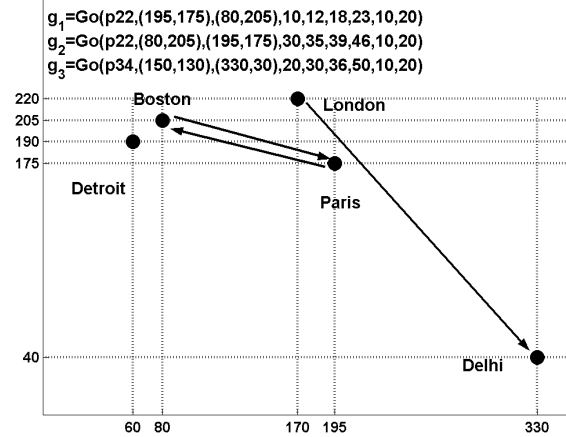


Figure 1: Planes example

together with algorithms to check entailment of certain kinds of atoms: “in” atoms that check if a given moving object is within a given region at a given time, and “near” atoms which are used to check if two objects are within a given distance of each other at some time.

Though the semantics given by [Yaman *et al.*, 2004] is adequate for positive atoms, it is inadequate for negative atoms. For example, suppose we consider the go-theory containing just the two statements above about plane p22. In this case, we would like to infer that plane p22 is not in Detroit at time 30, even though it may be theoretically possible for the plane to make it to Detroit. The goal of this paper is to ensure that intelligent negative inferences of this kind can be made from go-theories.

The contributions and organization of this paper are as follows: in Section 2, we recapitulate the syntax of go-theories from [Yaman *et al.*, 2004]. In section 3, we introduce the concept of a coherent model of a go-theory, and describe the concept of coherent entailment. We also introduce the *Motion Closed World Assumption* (MCWA for short) and show how the MCWA can be used to reason about negative information. Also in Section 3, we show that the problem of checking if a go-theory has a coherent model is NP-complete. In Section 4, we provide algorithms to evaluate “in” literals w.r.t. the MCWA semantics. We are developing algorithms to pro-

cess other kinds of queries such as the near literals described in [Yaman *et al.*, 2004] — however, space reasons prevent us from presenting them. Section 5 describes a prototype implementation to answer positive and negative “in” queries — the implementation shows that our system is highly scalable. We compare our work with related work in Section 6.

2 Go-Theories: Syntax and Semantics

We first provide a quick overview of the main definitions of [Yaman *et al.*, 2004]. We assume the existence of several sets of constant symbols: $\mathbf{R}$ is the set of all real numbers, $\mathbf{O}$ is the set of names of objects, $\mathbf{P} = \mathbf{R} \times \mathbf{R}$ is the set of all points in two-dimensional cartesian space. We assume the existence of three disjoint sets of variable symbols, $V_{\mathbf{R}}$, $V_{\mathbf{O}}$, and $V_{\mathbf{P}}$, ranging over $\mathbf{R}$, $\mathbf{O}$ and $\mathbf{P}$, respectively. A *real term* t is any member of $\mathbf{R} \cup V_{\mathbf{R}}$. Object terms and point terms are defined similarly. Ground terms are defined in the usual way. We now define atoms as follows.

- If o_1, o_2 are object terms, and d, t_1, t_2 are positive real terms, then $\text{near}(o_1, o_2, d, t_1, t_2)$ is an *atom*. When these terms are ground, this atom says that o_1, o_2 are within distance d of each other during the time interval $[t_1, t_2]$.
- If o is an object term, P_1, P_2 are point terms, and t_1, t_2 are positive real terms, then $\text{in}(o, P_1, P_2, t_1, t_2)$ is an *atom*. When these terms are ground, this atom says that object o is in the rectangle whose lower left (resp. upper right) corner is P_1 (resp. P_2) at some point in the time interval $[t_1, t_2]$.
- If o is an object term, P_1, P_2 are point terms, and $t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+$ are positive real terms, then $\text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ is an atom called a *go atom*. When all these terms are ground, this atom says that object o leaves point P_1 at some time in $[t_1^-, t_1^+]$ and arrives at point P_2 during $[t_2^-, t_2^+]$, traveling in a straight line with a minimum speed v^- and maximum speed v^+ .

Ground atoms are defined in the usual way. A *go theory* is a finite set of ground go-atoms.

Notation. If $g = \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ then

$$\begin{aligned} \text{obj}(g) &= o, & v^-(g) &= v^-, & v^+(g) &= v^+, \\ \text{loc}_1(g) &= P_1, & t_1^-(g) &= t_1^-, & t_1^+(g) &= t_1^+, \\ \text{loc}_2(g) &= P_2, & t_2^-(g) &= t_2^-, & t_2^+(g) &= t_2^+. \end{aligned}$$

If A is an atom, then A and $\neg A$ are called *literals*. Due to space constraints, we only consider literals in this paper — [Yaman *et al.*, 2004] provide a richer syntax including conjunction and disjunction.

An *interpretation* is a continuous function $\mathcal{I} : \mathbf{O} \times \mathbf{R}^+ \rightarrow \mathbf{P}$. Intuitively, $\mathcal{I}(o, t)$ is o ’s location at time t . We first define satisfaction of a “go” atom w.r.t. a given time interval.

Definition 1 Let $g = \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ be an atom and $\mathcal{I}$ be an interpretation. $\mathcal{I}$ *satisfies* g over a time interval $T = [t_1, t_2]$ iff:

- $t_1 \in [t_1^-, t_1^+]$ and $\mathcal{I}(o, t_1) = P_1$
- $t_2 \in [t_2^-, t_2^+]$ and $\mathcal{I}(o, t_2) = P_2$

- $\forall t \in [t_1, t_2], \mathcal{I}(o, t)$ is on the line segment $[P_1, P_2]$
- $\forall t, t' \in [t_1, t_2], t < t'$ implies $\text{dist}(\mathcal{I}(o, t), P_1) < \text{dist}(\mathcal{I}(o, t'), P_1)$ where dist is the function that computes the Euclidean distance between two points.
- For all but finitely many times in $[t_1, t_2]$, $v = d(|\mathcal{I}(o, t)|)/dt$ is defined and $v^-(g) \leq v \leq v^+(g)$.

The above definition intuitively says that $\mathcal{I} \models g$ over a time interval $T = [t_1, t_2]$ iff o starts moving at t_1 , stops moving at t_2 and during this interval, the object moves away from P_1 towards P_2 without either stopping or turning back or wandering away from the straight line connecting P_1 and P_2 . We are now ready to define the concept of satisfaction of arbitrary literals.

Definition 2 $\mathcal{I}$ satisfies a ground literal (denoted $\mathcal{I} \models A$) in these cases:

1. $\mathcal{I} \models \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+)$ iff there exists an interval $[t_1, t_2]$ such that $\mathcal{I}$ satisfies A over $[t_1, t_2]$.
2. $\mathcal{I} \models \text{near}(o_1, o_2, d, t_1, t_2)$ iff $\text{dist}(\mathcal{I}(o_1, t), \mathcal{I}(o_2, t)) \leq d$ for all $t_1 \leq t \leq t_2$
3. $\mathcal{I} \models \text{in}(o, P_1, P_2, t_1, t_2)$ iff there are numbers $t \in [t_1, t_2]$, $x \in [P_1^x, P_2^x]$ and $y \in [P_1^y, P_2^y]$ such that $\mathcal{I}(o, t) = (x, y)$.
4. $\mathcal{I} \models \neg A$ iff $\mathcal{I}$ does not satisfy A .

$\mathcal{I}$ satisfies (or is a model of) a set of ground atoms MT iff $\mathcal{I}$ satisfies every $A \in \text{MT}$. MT is *consistent* iff there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \text{MT}$. L is a *logical consequence* of MT, denoted $\text{MT} \models L$, iff every model of MT is also a model of L .

Example 1 The Planes go-theory of Figure 1 is consistent as the interpretations $\mathcal{I}_1, \mathcal{I}_2$ below both satisfy it.

- $\mathcal{I}_1$: p22 leaves Paris at time 11, flies to Boston at a constant speed of 14.85 and arrives in Boston at 19. p22 waits in Boston until 32, then it departs for Paris with a constant speed of 13.2 arriving in Paris at 41. The other plane, p34 leaves London at time 25 and flies to Delhi at a constant speed of 18.52, arriving in Delhi at 38.
- $\mathcal{I}_2$: p22 leaves Paris at time 10, flies to Boston at constant speed of 14.85 and reaches Boston at 18. It waits in Boston until 19, when it takes off for Detroit where it arrives at time 21. It immediately departs and reaches Boston at time 29. At time 30, p22 leaves Boston and flies to Paris at a constant speed of 11.88, arriving in Paris at time 40. The other plane, p34 leaves London at time 25 and flies to Delhi at a constant speed of 18.52, arriving in Delhi at 38.

It is important to note that even though $\mathcal{I}_2$ satisfies the Planes go theory, it is an interpretation that allows plane p22 to wander around in ways that were not explicitly stated in the Planes go theory. In particular, it lets the plane wander to Detroit which was never mentioned in the go-theory. We would like to exclude such “wandering” interpretations as they prevent us from making the intuitive (nonmonotonic) inference that Plane p22 was never in Detroit.

Throughout the rest of the paper we are going to use the notation $G[o]$ to denote the set of all atoms about an object o in a go theory G .

Definition 3 Let G be a go theory, o be an object and $G[o] = \{g_1, g_2, \dots, g_n\}$. Then for every $g_i, g_j \in G[o]$ we define a partial order $\preceq$ such that $g_i \preceq g_j$ iff $t_2^+(g_i) \leq t_1^-(g_j)$. A total order $\sqsubseteq$ on $G[o]$ is compatible with $G[o]$ iff $\sqsubseteq$ is a topological sort of $\preceq$.

Definition 4 Suppose G is a go theory, o is an object and $\sqsubseteq$ is a total order compatible with $G[o]$. Then $\mathcal{L}(G[o], \sqsubseteq)$ is set of linear constraints such that

- for every $g = \text{go}(o, P_1, P_2, t_1^-, t_1^+, t_2^-, t_2^+, v^-, v^+) \in G[o]$, $\mathcal{L}(G[o], \sqsubseteq)$ contains:
 - $t_1^- \leq S_g \leq t_1^+$ and $t_2^- \leq E_g \leq t_2^+$,
 - $v^- \times (E_g - S_g) \leq \text{dist}(P_1, P_2) \leq v^+ \times (E_g - S_g)$,
- for every $g, g' \in G[o]$ such that $g \sqsubseteq g'$, $\mathcal{L}(G[o], \sqsubseteq)$ contains: $E_g \leq S_{g'}$.

Intuitively S_g, E_g are variables that represent the times o starts and stops moving.

Definition 5 A go theory G is non-collinear iff for each object o there are no $g, g' \in G[o]$ such that

- The intersection of line segments $[loc_1(g), loc_2(g)]$ and $[loc_1(g'), loc_2(g')]$ is a line segment $[P, Q]$,
- The direction of the movement in g and g' is same, i.e., $\exists k \in \mathbf{R}^+$ such that $\vec{v} = k \times \vec{u}$ where $\vec{v} = loc_2(g) - loc_1(g)$ and $\vec{u} = loc_2(g') - loc_1(g')$
- $t_1^-(g) \leq t_2^+(g')$ and $t_1^-(g') \leq t_2^+(g)$, i.e. temporally overlapping.

The following theorem establishes necessary and sufficient conditions for non-collinear go theories to be consistent.

Theorem 1 A non-collinear go theory G is consistent iff for every object o there is a total order $\sqsubseteq_o$ compatible with $G[o]$ such that $\mathcal{L}(G[o], \sqsubseteq_o)$ has a solution.

3 Coherence

In this section, we define the concept of a coherent interpretation. We start by defining precedence of time intervals.

Definition 6 (Precedence) Let $S = \{T_1, \dots, T_n\}$ be a set of time intervals, where $T_i = [t_{i1}, t_{i2}]$ for each i . T_i **immediately precedes** T_j in S if $t_{i2} \leq t_{j1}$ and for every $T_k \in S$, either $t_{k2} \leq t_{i2}$ or $t_{k1} \geq t_{j1}$.

Intuitively, $\mathcal{I}$ is a coherent interpretation of a go theory G if for each object o , there is a time interval T such that for every time point $t \in T$, $\mathcal{I}(o, t)$ either satisfies a go-atom in G or keeps the object at the destination of the last satisfied go-atom in G .

Definition 7 (Coherent Model and Theory) Let $\mathcal{I}$ be a model of the go theory G . Let $G[o] = \{g_1, g_2, \dots, g_n\}$ be the set of all go-atoms in G about object o . $\mathcal{I}$ is **coherent w.r.t. o and G** iff

- There are time intervals $T_1 = [t_{11}, t_{12}]$, $T_2 = [t_{21}, t_{22}]$, $\dots$, $T_n = [t_{n1}, t_{n2}]$ such that for each i , $\mathcal{I}$ satisfies g_i over T_i and
- For every pair of time intervals T_i, T_j such that T_i immediately precedes T_j in $\{T_1, T_2, \dots, T_n\}$ the following holds:

$$\forall t \in [t_{i2}, t_{j1}] \mathcal{I}(o, t) = loc_2(g_i), \text{ i.e. destination of } g_i.$$

$\mathcal{I}$ is a **coherent model of G** iff $\mathcal{I}$ is coherent w.r.t. o and G for all objects o .

G is a **coherent go-theory** iff G has a coherent model.

Example 2 Let G be the go theory in Figure 1. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the two interpretations in Example 1. $\mathcal{I}_1$ is coherent with respect to G and $p22$ because it satisfies g_1 over $[11, 19]$, g_2 over $[32, 40]$ and in between $[19, 32]$ plane $p22$ is in Boston. $\mathcal{I}_2$ is not coherent with respect to G and $p22$ because although it satisfies g_1 over $[10, 18]$, g_2 over $[30, 41]$ during $[18, 30]$, $p22$ does not stay in Boston which is the destination of g_1 .

The following lemma and definition are useful in checking whether a non-collinear go-theory has a coherent model or not.

Definition 8 Suppose G is a non-collinear go theory, o is an object and $\sqsubseteq$ is a total order compatible with $G[o]$. Let $g_1 \sqsubseteq g_2 \dots \sqsubseteq g_n$ be the atoms of $G[o]$. $\sqsubseteq$ is **spatially continuous** w.r.t. $G[o]$ iff for every i , $1 \leq i < n$, $loc_2(g_i) = loc_1(g_{i+1})$, i.e., g_i 's destination is g_{i+1} 's origin;

Lemma 1 Suppose G is a non-collinear go theory. G is coherent if for every object o there is a total order $\sqsubseteq_o$ compatible with $G[o]$ such that $\mathcal{L}(G[o], \sqsubseteq_o)$ has a solution and $\sqsubseteq_o$ is spatially continuous w.r.t. $G[o]$.

The following theorem shows that checking coherence of a go-theory is NP-complete.

Theorem 2 (i) Checking coherence of a non-collinear go theory is NP-complete. **(ii)** Checking coherence of a go theory is NP-complete.

The proof is omitted due to lack of space. We now define the concept of coherent entailment.

Definition 9 (MCWA entailment) Let L be a ground literal and G be a go theory. G **entails L via MCWA**, denoted $G \models^{mcwa} L$, iff every coherent model of G also satisfies L .

The MCWA is inspired by Minker's generalized closed world assumption [Minker, 1982] where a class of models is used to check if a given literal is true. We do the same here. The following example shows that the MCWA can handle examples such as the Planes example.

Example 3 Let G be the go theory in Figure 1. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the interpretations in Example 1. Suppose $a = \text{in}(p22, (75, 200), (85, 210), 23, 30)$. $G \models^{mcwa} a$ since in all coherent models of G , during $[23, 30]$ plane $p22$ is in Boston which is inside the rectangle of the atom a .

Suppose $b = \text{in}(p22, (55, 185), (80, 200), 23, 30)$. Then $G \models^{mcwa} \neg b$ since in all coherent models of G , during $[23, 30]$ plane $p22$ stays in Boston which is not in the rectangle of the atom b .

Also note that $G \not\models a$ and $G \not\models \neg b$ because according to the semantics in [Yaman et al., 2004] plane $p22$ can be anywhere during $[23, 30]$.

Theorem 3 Let L be a ground $\text{in}()$ literal and G be a go theory. Checking if $G \models^{mcwa} L$ is co-NP complete.

Since incoherent theories entail everything the following section describes algorithms for coherent go theories.

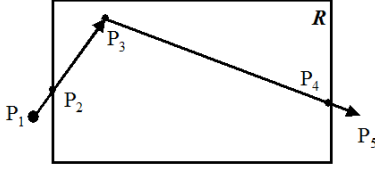


Figure 2: Spatial layout of two go atoms (related to object o) going from P_1 to P_3 and P_3 to P_5 and a rectangle R . In a coherent model, o stays in R between the points P_2 and P_4 .

4 MCWA-Entailment Algorithms

This section provides algorithms to check for MCWA-entailment of both positive and negative **ground** literals. Due to space limitations we assume all theories are non-collinear go theories. Extending our algorithm to remove this assumption is straightforward using the methods defined in [Yaman *et al.*, 2004] that combine collinear go atoms into “movements”—our implementation applies to all go theories.

We first introduce some notations. If g is a go atom, then $LS(g)$ is the line segment between the source and destination of g . Let G be a go-theory, o be an object and $\sqsubseteq$ be a total ordering compatible with $G[o]$. Let P be a point on $LS(g)$ where $g \in G$. Then $T^-(G[o], \sqsubseteq, g, P)$ and $T^+(G[o], \sqsubseteq, g, P)$ are the earliest and latest possible times for o to be at P , subject to $G, o, \sqsubseteq$ and g .¹

4.1 Coherent $\text{in}()$

In this section, we show how to check whether a **ground** atom of the form $a = \text{in}(o, q_1, q_2, t_1, t_2)$ is MCWA-entailed by a go-theory G . Let $\text{Rec}(a)$ denote the set of points P such that $q_1^x \leq P^x \leq q_2^x$ and $q_1^y \leq P^y \leq q_2^y$.

We first consider a non-collinear go theory $G = \{g_1, g_2\}$ about an object o , and an atom $a = \text{in}(o, q_1, q_2, t_1, t_2)$. Assume Figure 2 depicts $\text{Rec}(a)$ and the two line segments $[P_1, P_3], [P_3, P_5]$ representing movements defined by g_1 and g_2 . In any coherent model of G , g_1 will be satisfied before g_2 . Hence the object enters $\text{Rec}(a)$ at P_2 and leaves $\text{Rec}(a)$ at P_4 . If o always arrives at P_2 before t_2 and always leaves P_4 after t_1 subject to the constraints in G , then we can say that $G \models^{mcwa} a$.

For an arbitrary go theory, any object might enter and leave $\text{Rec}(a)$ multiple times. We need to identify these entrance and exit points as well as the atoms that contain them.

Definition 10 Let L be a sequence of line segments $\ell_1 = [P_{11}P_{12}]$, $\ell_2 = [P_{21}P_{22}]$, ..., $\ell_n = [P_{n1}P_{n2}]$ such that for

¹ $T^-(G[o], \sqsubseteq, g, P)$ is the solution to linear programming problem: **minimize** $X_{P,g}$ **subject to** $\mathcal{L}(G[o], \sqsubseteq) \cup L(P, g)$, where $L(P, g)$ contains the following linear constraints:

- $\frac{\text{dist}(\text{loc}_1(g), P)}{v^+(g)} \leq X_{P,g} - S_g \leq \frac{\text{dist}(\text{loc}_1(g), P)}{v^-(g)}$
- $\frac{\text{dist}(\text{loc}_2(g), P)}{v^+(g)} \leq E_g - X_{P,g} \leq \frac{\text{dist}(\text{loc}_2(g), P)}{v^-(g)}$

where $X_{P,g}$ is the variable that represents the time the object will arrive P while satisfying g . S_g and E_g are the variables associated with g in $\mathcal{L}(G[o], \sqsubseteq)$. $T^+(G[o], \sqsubseteq, g, P)$ can be computed in the same way, using maximization instead of minimization.

$1 \leq i < n$, $P_{i2} = P_{(i+1)1}$. Let R be a rectangular region. An entry-exit of L for R is (i, j) iff

- $\ell_i \cap R \neq \emptyset$ and $i > 1 \implies P_{i1} \notin R$
- $\ell_j \cap R \neq \emptyset$ and $j < n \implies P_{j2} \notin R$
- $\forall k \in [i, j) P_{k2} \in R$

The following lemma gives necessary conditions for $G \models^{mcwa} a$ when the atoms in G are satisfied in a specific order and the object enters and exits $\text{Rec}(a)$ multiple times.

Lemma 2 Let G be a coherent go theory, o be an object and $\sqsubseteq$ be a total order compatible with $G[o]$ such that $\mathcal{L}(G[o], \sqsubseteq)$ has a solution and $\sqsubseteq$ is spatially continuous w.r.t $G[o]$. Let $g_1 \sqsubseteq g_2 \cdots \sqsubseteq g_n$ be the atoms of $G[o]$. Let $a = \text{in}(o, q_1, q_2, t_1, t_2)$ be an atom. If $G \models^{mcwa} a$ then there is an entry-exit (i, j) of $LS(g_1) \dots LS(g_n)$ for $\text{Rec}(a)$ such that

$$T^+(G[o], \sqsubseteq, g_i, P_i) \leq t_2 \text{ and } t_1 \leq T^-(G[o], \sqsubseteq, g_j, Q_j).$$

where $[P_k, Q_k] = LS(g_k) \cap \text{Rec}(a)$,

The following algorithm uses this lemma to check for MCWA-entailment w.r.t. a specific total ordering.

Algorithm CheckCoherentIn($G, \sqsubseteq, a$)
 Suppose $a = \text{in}(o, q_1, q_2, t_1, t_2)$;
 Let $g_1 \sqsubseteq g_2 \cdots \sqsubseteq g_n$ be atoms of $G[o]$
if $\sqsubseteq$ is not spatially continuous w.r.t $G[o]$ **then return true**
if $\mathcal{L}(G[o], \sqsubseteq)$ has no solution **then return true**
for each entry-exit (i, j) of $LS(g_1) \dots LS(g_n)$ for $\text{Rec}(a)$
 Let $[P_i, Q_i] = LS(g_i) \cap \text{Rec}(a)$
 Let $[P_j, Q_j] = LS(g_j) \cap \text{Rec}(a)$
 if $T^+(G[o], \sqsubseteq, g_i, P_i) \leq t_2$ and $t_1 \leq T^-(G[o], \sqsubseteq, g_j, Q_j)$
 then return true
end for
return false

Theorem 4 Suppose G is a coherent go-theory and $a = \text{in}(o, q_1, q_2, t_1, t_2)$ is a ground atom. Then: a is entailed by G via MCWA iff for every total order $\sqsubseteq$ compatible with $G[o]$, the algorithm CheckCoherentIn($G, \sqsubseteq, a$) returns “true”.

4.2 Coherent $\neg \text{in}()$

We now address the problem of checking whether a literal of the form $\neg \text{in}(o, Q_1, Q_2, t_1, t_2)$ is MCWA-entailed by a go-theory G .

Consider a coherent go theory $G = \{g_1, g_2\}$ about an object o , and an in-atom $a = \text{in}(o, q_1, q_2, t_1, t_2)$. As before Figure 2 depicts $\text{Rec}(a)$ and two line segments $[P_1, P_3], [P_3, P_5]$ representing the movements defined by g_1 and g_2 . Note that in any coherent model of G , g_1 is satisfied before g_2 . Hence the object enters $\text{Rec}(a)$ at point P_2 and leaves $\text{Rec}(a)$ at point P_4 . $G \models^{mcwa} \neg a$ iff

- t_1 is greater than or equal to the start time of g_1 in any coherent model of G .
- t_2 is smaller than or equal to the end time of g_2 in any coherent model of G .
- Let T_1 be the earliest arrival time to P_2 and T_2 be the latest arrival time to P_4 in any coherent model of G then $T_1 > t_2$ or $T_2 < t_1$.

The following lemma gives necessary conditions for $G \models^{mcwa} \neg a$ to hold w.r.t. a specific total ordering $\sqsubseteq$ even if the object enters and exits $Rec(a)$ multiple times.

Lemma 3 Let G be a coherent go theory, o be an object and $\sqsubseteq$ be a total order compatible with $G[o]$ such that $\mathcal{L}(G[o], \sqsubseteq)$ has a solution and $\sqsubseteq$ is spatially continuous w.r.t. $G[o]$. Let $g_1 \sqsubseteq g_2 \cdots \sqsubseteq g_n$ be the atoms of $G[o]$. Let $a = \text{in}(o, q_1, q_2, t_1, t_2)$ be a ground atom. If $G \models^{mcwa} \neg a$ then the following hold

- $T^+(G[o], \sqsubseteq, g_1, loc_1(g_1)) \leq t_1$
- $T^-(G[o], \sqsubseteq, g_n, loc_2(g_n)) \geq t_2$
- $\forall$ entry-exit (i, j) of $LS(g_1) \dots LS(g_n)$ for $Rec(a)$, $T^-(G[o], \sqsubseteq, g_i, P_i) > t_2$ or $T^+(G[o], \sqsubseteq, g_j, Q_j) < t_1$ where $[P_k, Q_k] = LS(g_k) \cap Rec(a)$.

The following algorithm checks if $G \models^{mcwa} \neg \text{in}()$ w.r.t. a specific total ordering.

Algorithm CheckCoherentNotIn($G, \sqsubseteq, \neg a$)
 Suppose $a = \text{in}(o, p_1, p_2, t_1, t_2)$;
 Let $g_1 \sqsubseteq g_2 \cdots \sqsubseteq g_n$ the atoms of $G[o]$.
if $\sqsubseteq$ is not spatially continuous w.r.t $G[o]$ **then return true**
if $\mathcal{L}(G[o], \sqsubseteq)$ has no solution **then return true**
if $t_1 < T^+(G[o], \sqsubseteq, g_1, loc_1(g_1))$ **return false**
if $t_2 > T^-(G[o], \sqsubseteq, g_n, loc_2(g_n))$ **return false**
for each entry-exit (i, j) of $LS(g_1) \dots LS(g_n)$ for $Rec(a)$
 Let $[P_i, Q_i] = LS(g_i) \cap Rec(a)$
 Let $[P_j, Q_j] = LS(g_j) \cap Rec(a)$
 if $T^-(G[o], \sqsubseteq, g_i, P_i) \leq t_2$ and $t_1 \leq T^+(G[o], \sqsubseteq, g_j, Q_j)$
 then return false
end for
return true

Theorem 5 Suppose G is a coherent go-theory and $L = \neg \text{in}(o, q_1, q_2, t_1, t_2)$ is a ground literal. Then L is entailed by G via MCWA iff for every total order $\sqsubseteq$ compatible with $G[o]$, the algorithm CheckCoherentNotIn($G, \sqsubseteq, a$) returns “true”.

5 Implementation

Determining MCWA-entailment is co-NP complete because the number of orderings spatially continuous w.r.t. $G[o]$ can be exponential. However, in the real world, we expect a go-theory to allow only a small number of orderings compatible with $G[o]$. In other words, the respective order of movements an object is going to perform is mostly known. For example we might not know exactly when the plane $p22$ will land but we usually know where it is going to fly next. Thus, in practice there is a bound on the number of compatible total orderings per object.

For our experiments we generated random go theories with at most 256 spatially continuous orderings. This is not a hard-coded limit of our implementation. Generating random go-theories such that more than one spatially continuous ordering exists is a little bit tricky. Here is one method to generate a go theory $G = \{g_1, g_2, g_3, g_4, g_5\}$ with two spatially continuous orderings.

- Randomly pick points P_1, P_2, P_3 and P_4

- Set $loc_1(g_1) = P_1$ and $loc_2(g_1) = P_2$,
- Set $loc_1(g_2) = P_2, loc_2(g_2) = P_3$
and $loc_1(g_3) = P_3, loc_2(g_3) = P_2$,
- Set $loc_1(g_4) = P_2, loc_2(g_4) = P_4$
and $loc_1(g_5) = P_4, loc_2(g_5) = P_2$,
- Set temporal and speed intervals of every g_i so that g_1 is always first and the rest can be done in any order.

We have generalized the reasoning above to create random go theories with an arbitrary bound on the number of spatially continuous orderings.

We have implemented the two algorithms CheckCoherentIn and CheckCoherentNotIn in Matlab and conducted experiments on a mobile Athlon XP 1800 processor running under Windows XP and having 256MB of memory. Figure 3 shows the computation time of four types of queries for coherent go theories with at most 256 spatially continuous orderings and have the following properties: all points are selected randomly from the rectangle $[(0, 0), (1000, 1200)]$ and the speeds allowed for any object less than 100. The four query templates we used are:

- Q1:** $\text{in}(o, (500, 500), (550, 600), 0.5 * h, 0.75 * h)$
- Q2:** $\text{in}(o, (100, 150), (350, 400), h - 100, h - 10)$
- Q3:** $\neg Q1$
- Q4:** $\neg Q2$

where h is the latest end time for any atom related to o in the given theory. The data points in Figure 3 are an average of 300 runs.

Our implementation performs very well, executing most queries in less than 0.3 seconds even when there are as many as 1,000 go-atoms per object. In the query $Q1$ where CheckCoherentIn returns true in almost every compatible orderings the algorithm runs in linear time with respect to number of atoms per object and takes up to 0.9 seconds when there are 1,000 go-atoms per object. Consequently $Q3$, the complement of $Q1$, takes almost no time because CheckCoherentNotIn returns false for any compatible ordering.

6 Related work

The Closed World Assumption (CWA) proposed by [Reiter, 1977] holds that anything that cannot be entailed by a theory is false. Minker [Minker, 1982] extended the CWA to a Generalized CWA (GCWA) that accounts for disjunction. GCWA states that a formula is false if it is false in all minimal models of the theory. The go-theories proposed by [Yaman *et al.*, 2004] are disjunctive because the start and end times and object velocities are all known to be within a given range. The notion of a coherent model of a go-theory selects certain models (much like Minker selected minimal models in GCWA) and uses these to make closed world inferences.

[Intille, 1994; Intille *et al.*,] have used the CWA to track moving objects in football games using computer vision algorithms. They use CWA to adaptively select and weight image features used for correspondence. No motion reasoning of the type we perform in this paper is done.

Numerous spatio-temporal logics exist [Gabelaia *et al.*, 2003; Merz *et al.*, 2003; Wolter and Zakharyashev, 2000;

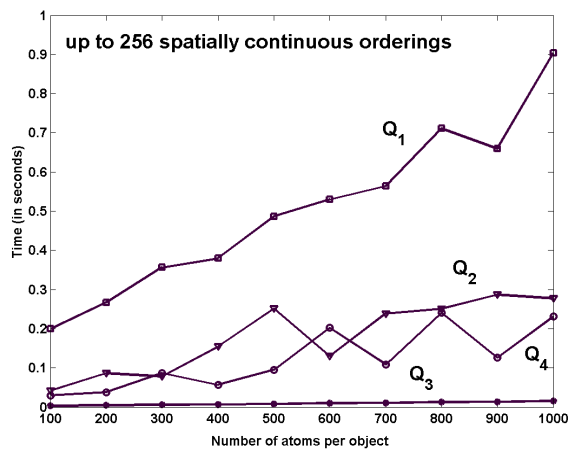


Figure 3: Time to answer queries Q_1 , Q_2 , Q_3 and Q_4 when total number of spatially continuous orderings is at most 256.

Cohn *et al.*, 2003] have proposed spatio-temporal logics. These logics extend temporal logics to handle space. By and large, these frameworks use discrete rather than the continuous representations we use. Moreover, these works focus on qualitative aspects of spatio-temporal reasoning, rather than deal quantitatively with the dynamics of motion. Our methods are rooted in a mix of geometry and logic, rather than in logic alone. Cohn [Anthony G. Cohn, 2001] provides an excellent survey of spatio-temporal logics but the survey sheds little light on reasoning above motion. A notable exception is the work of Muller [Muller, 1998a; 1998b] who describes a formal *qualitative* theory for reasoning about motion. The expressive power of the theory allows for the definition of complex motion classes. The work however is purely symbolic not quantitative.

7 Conclusions

Yaman *et al.* [Yaman *et al.*, 2004] introduce a logic based on “go theories” for reasoning about moving objects. In this paper, we show that this logic often does not allow us to infer that an object is *not* present at a given place or region, even though common sense would dictate that this is a reasonable inference to make. We define a class of models of go-theories called *coherent* models. We use this concept to define a *motion closed world assumption* (MCWA) and develop a notion of MCWA-entailment. We show that checking if a go-theory has a coherent model is NP-complete. An in atom checks if a given object is present in a given region during a given time interval. We provide sound and complete algorithms to check if an in literal (positive or negative in atom) can be inferred from a go-theory using the MCWA. In our experiments our algorithms executed queries in less than 1 second even when there are as many as 1,000 go-atoms per object.

Acknowledgements

This work was supported in part by ARO grant DAAD190310202, ARL grants DAAD190320026 and

DAAL0197K0135, the ARL CTAs on Telecommunications and Advanced Decision Architectures, NSF grants IIS0329851 and IIS0412812 and 0205489 and UC Berkeley contract number SA451832441 (subcontract from DARPA’s REAL program). The opinions expressed in this paper are those of authors and do not necessarily reflect the opinions of the funders.

References

- [Anthony G. Cohn, 2001] Shyamanta M. Hazarika Anthony G. Cohn. Qualitative spatial representation and reasoning: An overview. *Fundam. Inform.*, 46(1-2):1–29, 2001.
- [Cohn *et al.*, 2003] A. G. Cohn, D. Magee, A. Galata, D. Hogg, and S. Hazarika. Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In C. Freksa, C. Habel, and K.F. Wender, editors, *Spatial Cognition III*. 2003.
- [Gabelaia *et al.*, 2003] David Gabelaia, Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyashev. On the computational complexity of spatio-temporal logics. In *Proceedings of the 16th AAI International FLAIRS Conference*, pages 460–464, 2003.
- [Intille *et al.*,] S. Intille, J. Davis, and A. Bobick. Real-time closed-world tracking. In *IEEE CVPR*, pages 697–703, 1997.
- [Intille, 1994] S. Intille. Tracking using a local closed-world assumption: Tracking in the football domain, 1994. Master’s Thesis, M.I.T. Media Lab.
- [Merz *et al.*, 2003] Stephan Merz, Júlia Zappe, and Martin Wirsing. A spatio-temporal logic for the specification and refinement of mobile systems. In *Fundamental Approaches to Software Engineering*, volume 2621, pages 87–101. Springer-Verlag, 2003.
- [Minker, 1982] J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction*, volume 138 of *Lecture Notes in Computer Science*, pages 292–308. Springer Verlag, 1982.
- [Muller, 1998a] Philippe Muller. A qualitative theory of motion based on spatio-temporal primitives. In *Proceedings of KR’98*, pages 131–141, 1998.
- [Muller, 1998b] Philippe Muller. Space-time as a primitive for space and motion. In *FOIS’98*, pages 63–76, Amsterdam, 1998.
- [Reiter, 1977] Raymond Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 55–76, 1977.
- [Wolter and Zakharyashev, 2000] Frank Wolter and Michael Zakharyashev. Spatio-temporal representation and reasoning based on RCC-8. In *Proceedings of KR2000*, pages 3–14, 2000.
- [Yaman *et al.*, 2004] Fusun Yaman, Dana Nau, and V S Subrahmanian. A logic of motion. In *Proceedings of KR2004*, pages 85–94, 2004.

Solving Logic Program Conflict through Strong and Weak Forgetting

Yan Zhang

School of Computing & IT
University of Western Sydney
NSW 1797, Australia
E-mail: yan@cit.uws.edu.au

Norman Foo

School of Comp. Sci. & Eng.
University of New South Wales
NSW 2052, Australia

Kewen Wang

School of Computing & IT
Griffith University
QLD 4111, Australia
E-mail: k.wang@cit.gu.edu.au

Abstract

We consider how to forget a set of atoms in a logic program. Intuitively, when a set of atoms is forgotten from a logic program, all atoms in the set should be eliminated from this program in some way, and other atoms related to them in the program might also be affected. We define notions of strong and weak forgettings in logic programs to capture such intuition and reveal their close connections to the notion of forgetting in classical propositional theories. Based on these notions, we then propose a framework for conflict solving in logic programs, which is general enough to represent many important conflict solving problems. We also study some essential semantic and computational properties in relation to strong and weak forgettings and conflict solving in our framework.

1 Introduction

One promising approach in the research of reasoning about knowledge dynamics is to represent agents' knowledge bases as logic programs on which necessary updates are conducted as a way to model agents' knowledge evolution. A key issue in this study is to solve various conflicts and inconsistencies in logic programs, e.g. [Leite, 2003].

While different logic program update approaches have been developed recently, we observe that some typical conflict solving problems in logic programs have yet to be thoroughly investigated in the literature. Let us consider a scenario. John wants Sue to help him to complete his assignment. He knows that Sue will help him if she is not so busy. Tom is a good friend of John and wants John to let him copy John's assignment. Then John also learns that Sue hates Tom, and will not help him if he lets Tom copy his assignment, which will be completed under Sue's help. While John does not care whether Sue hates Tom or not, he has to consider Sue's condition to offer him help. What is John going to do? We formalize this scenario in a logic programming setting. John's knowledge base Π_J :

$$\begin{aligned} r_1 : & \text{Complete}(\text{John}, \text{Assignment}) \leftarrow \\ & \text{Help}(\text{Sue}, \text{John}), \\ r_2 : & \text{Help}(\text{Sue}, \text{John}) \leftarrow \text{notBusy}(\text{Sue}), \end{aligned}$$

$$\begin{aligned} r_3 : & \text{Goodfriend}(\text{John}, \text{Tom}) \leftarrow, \\ r_4 : & \text{Copy}(\text{Tom}, \text{Assignment}) \leftarrow \\ & \text{Goodfriend}(\text{John}, \text{Tom}), \\ & \text{Complete}(\text{John}, \text{Assignment}), \end{aligned}$$

and Sue's knowledge base Π_S :

$$\begin{aligned} r_5 : & \text{Hate}(\text{Sue}, \text{Tom}) \leftarrow, \\ r_6 : & \leftarrow \text{Help}(\text{Sue}, \text{John}), \text{Copy}(\text{Tom}, \text{Assignment}). \end{aligned}$$

In order to take Sue's knowledge base into account, suppose John updates his knowledge base Π_J in terms of Sue's Π_S . By applying proper logic program update approach, John may obtain a solution: $\Pi_J^{\text{final}} = \{r_1, r_2, r_3, r_5, r_6\}$ or its stable model, from which we know that *Sue* will help *John* to complete the assignment and *John* will not let *Tom* copy his assignment. Although the conflict between Π_J and Π_S has been solved by updating, the result is somehow not always satisfactory. For instance, while John wants Sue to help him, he may have no interest at all in integrating the information that Sue hates Tom into his new knowledge base.

As an alternative, John may just weaken his knowledge base by *forgetting* atom $\text{Copy}(\text{Tom}, \text{Assignment})$ from Π_J in order to accommodate Sue's constraint. Then John may have a new program $\Pi_J^{\text{final}} = \{r_1, r_2, r_3\}$ - John remains a maximal knowledge subset which is consistent with Sue's condition without being involved in Sue's personal feeling about Tom.

The formal notion of forgetting in propositional theories was initially considered by Lin and Reiter from a cognitive robotics perspective [Lin and Reiter, 1994] and has recently received a great attention in KR community. It has been shown that the theory of forgetting has important applications in solving knowledge base inconsistencies, belief update and merging, abductive reasoning, causal theories of actions, and reasoning about knowledge under various propositional (modal) logic frameworks, e.g. [Lang and Marquis, 2002; Lang *et al.*, 2003; Lin, 2001; Su *et al.*, 2004].

In this paper, we consider how to forget a set of atoms from a propositional normal logic program and how this idea can be used in general conflict solving under the context of logic programs. The rest of this paper is organized as follows. We present preliminary definitions and concepts in section 2. In section 3, we give formal definitions of strong and weak forgettings in logic programs. Based on these notions, in section 4, we propose a framework called logic program contexts for

general conflict solving in logic programs. In section 5, we investigate related semantic and computational properties. In section 6 we conclude the paper with some discussions.

2 Preliminaries

We consider finite propositional normal logic programs in which each rule has the form:

$$a \leftarrow b_1, \dots, b_m, \text{not} c_1, \dots, \text{not} c_n \quad (1)$$

where a is either a propositional atom or empty, and $b_1, \dots, b_m, c_1, \dots, c_n$ are propositional atoms. When a is empty, rule (1) is called a *constraint*. Given a rule r of the form (1), we denote $\text{head}(r) = \{a\}$, $\text{pos}(r) = \{b_1, \dots, b_m\}$ and $\text{neg}(r) = \{c_1, \dots, c_n\}$, and therefore, rule (1) may be represented as the form:

$$\text{head}(r) \leftarrow \text{pos}(r), \text{not } \text{neg}(r). \quad (2)$$

We also use $\text{Atom}(r)$ to denote the set of all atoms occurring in rule r . For a program Π , we define notions $\text{head}(\Pi) = \bigcup_{r \in \Pi} \text{head}(r)$, $\text{pos}(\Pi) = \bigcup_{r \in \Pi} \text{pos}(r)$, $\text{neg}(\Pi) = \bigcup_{r \in \Pi} \text{neg}(r)$, $\text{Atom}(\Pi)$ the set of all propositional atoms occurring in program Π . Given sets of atoms P and Q , we may use notion

$$r' : \text{head}(r) \leftarrow (\text{pos}(r) - P), \text{not } (\text{neg}(r) - Q)$$

to denote rule r' obtained from r by removing all atoms occurring in P and Q in the positive and negation as failure parts respectively.

The stable model of a program Π is defined as follows. Firstly, we consider Π to be a program in which each rule does not contain negation as failure sign *not*. A finite set S of propositional atoms is called a *stable model* of Π if S is the smallest set such that for each rule $a \leftarrow b_1, \dots, b_m$ from Π , if $b_1, \dots, b_m \in S$, then $a \in S$. Now let Π be an arbitrary normal logic program. For any set S of atoms, program Π^S is obtained from Π by deleting (1) each rule from Π that contains *not* c in the body and $c \in S$; and (2) all forms of *not* c in the bodies of the remaining rules. Then S is a stable model of Π if and only if S is a stable model of Π^S . A program may have one, more than one, or no stable models at all. A program is called *consistent* if it has a stable model. We say that an atom a is *entailed* from program Π , denoted as $\Pi \models a$ if a is in every stable model of Π .

Given two programs Π_1 and Π_2 . Π_1 and Π_2 are *equivalent* if Π_1 and Π_2 have the same stable models. Π_1 and Π_2 are called *strongly equivalent* if for every program Π , $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent [Lifschitz et al., 2001].

Observation: Let Π be a logic program. If each rule r in Π is of one of the following two forms: (1) $\text{head}(r) \neq \emptyset$ and $\text{head}(r) \subseteq \text{pos}(r)$, or (2) $\text{pos}(r) \cap \text{neg}(r) \neq \emptyset$, then Π is strongly equivalent to the empty set.

For a later reference, we call the two types of rules mentioned above *redundant rules*.

Let Π be a logic program. We use $[\Pi]^C$ to denote the conjunctive normal form obtained from Π by translating each rule of the form (1) in Π into the clause: $a \vee \neg b_1 \vee \dots \vee \neg b_m \vee c_1 \vee \dots \vee c_n$. For instance, if $\Pi = \{a \leftarrow \text{not} b, c \leftarrow a\}$, then we have $[\Pi]^C = (a \vee b) \wedge (c \vee \neg a)$. In general, we may write $[\Pi]^C = \{C_1, \dots, C_n\}$ where each C_i is a conjunct of $[\Pi]^C$.

If C_i is a clause, we call any sub formula of C_i a *subclause* of C_i .

Now we introduce the notion of forgetting in a classical propositional theory [Lin and Reiter, 1994; Lin, 2001]. Let T be propositional theory. We use $T(p/\text{true})$ (or $T(p/\text{false})$, resp.) to denote the theory obtained from T by substituting all occurrences of propositional atom p with *true* (or *false*, resp.). For instance, if $T = \{p \supset q, (q \wedge r) \supset s\}$, then $T(q/\text{true}) = \{r \supset s\}$ and $T(q/\text{false}) = \{\neg p\}$. Then we can define the notion of forgetting in terms of a propositional theory. For a given propositional theory T and a set of propositional atoms P , the result of *forgetting* P in T , denoted as $\text{Forget}(T, P)$, is defined inductively as follows:

$$\begin{aligned} \text{Forget}(T, \emptyset) &= T, \\ \text{Forget}(T, \{p\}) &= T(p/\text{true}) \vee T(p/\text{false}), \\ \text{Forget}(T, P \cup \{p\}) &= \text{Forget}(\text{Forget}(T, \{p\}), P). \end{aligned}$$

It is easy to see that the ordering in which atoms in P are considered does not affect the final result of forgetting P from T . Consider $T = \{p \supset q, (q \wedge r) \supset s\}$ again. From the above definition, we have $\text{Forget}(T, \{q\}) = \{(r \supset s) \vee \neg p\}$.

3 Strong and Weak Forgettings in Logic Programs

Let us consider how to forget a set of atoms from a logic program. Intuitively, we would expect that after forgetting a set of atoms, all occurrences of these atoms in the underlying program should be eliminated in some way, and moreover, other atoms having connections to them through rules in the program might also be affected. We observe that the notion of forgetting in propositional theories is not applicable to logic programs since there is no disjunctive operation for logic programs. Further, different ways of handling negation as failure in forgetting may also lead to different resulting programs. To formalize our idea of forgetting in logic programs, we first introduce a program transformation called *reduction*.

Definition 1 (Program reduction) Let Π be a program and p an atom. We define the reduction of Π with respect to p , denoted as $\text{Reduct}(\Pi, \{p\})$, to be a program obtained from Π by (1) for each rule r with $\text{head}(r) = p$ and each rule r' with $p \in \text{pos}(r')$, replacing r' with a new rule $r'' : \text{head}(r'') \leftarrow (\text{pos}(r) \cup \text{pos}(r') - \{p\}), \text{not}(\text{neg}(r) \cup \text{neg}(r'))$; (2) if there is such rule r' in Π and has been replaced by r'' in (1), then removing rule r from the remaining program. Let P be a set of atoms. Then the reduction of Π with respect to P is inductively defined as follows:

$$\begin{aligned} \text{Reduct}(\Pi, \emptyset) &= \Pi, \\ \text{Reduct}(\Pi, P \cup \{p\}) &= \text{Reduct}(\text{Reduct}(\Pi, \{p\}), P). \end{aligned}$$

Note that in our program reduction definition, step (1) is the same as logic program *unfolding* [Brass and Dix, 1999]. While unfolding is to eliminate positive body occurrences of an atom in a logic program, the reduction, on other hand, is further to remove those rules with heads of this atom.

Example 1 Let $\Pi_1 = \{a \leftarrow \text{not} b, a \leftarrow d, c \leftarrow a, \text{note}\}$, $\Pi_2 = \{a \leftarrow c, \text{not} b, c \leftarrow \text{not} d\}$, and $\Pi_3 = \{a \leftarrow b, b \leftarrow \text{not} d, c \leftarrow a, \text{note}\}$. Then $\text{Reduct}(\Pi_1, \{a\}) = \{c \leftarrow \text{not} b, \text{note}, c \leftarrow d, \text{note}\}$, $\text{Reduct}(\Pi_2, \{a\}) = \Pi_2$, and $\text{Reduct}(\Pi_3, \{a, b\}) = \{c \leftarrow \text{not} d, \text{note}\}$. $\square$

Definition 2 (Strong forgetting) Let Π be a logic program, and p an atom. We define a program to be the result of strongly forgetting p in Π , denoted as $SForgetLP(\Pi, \{p\})$, if it is obtained from the following transformation:

1. $\Pi' = Reduct(\Pi, \{p\})$;
2. $\Pi' = \Pi' - \{r \mid r \text{ is a redundant rule}\}$;
3. $\Pi' = \Pi' - \{r \mid head(r) = p\}$;
4. $\Pi' = \Pi' - \{r \mid p \in pos(r)\}$;
5. $\Pi' = \Pi' - \{r \mid p \in neg(r)\}$;
6. $SForgetLP(\Pi, \{p\}) = \Pi'$.

Let us take a closer look at Definition 2. Step 1 is just to perform reduction on Π with respect to atom p . This is to replace those *positive body occurrences* of p in rules with other rules having p as the head. Step 2 is to remove all redundant rules which may be introduced by the reduction of Π with respect to p . From Observation in section 2, we know that this does not change anything in the program. Steps 3 and 4 are to remove those rules which have p as the head or in the positive body. Note that after Steps 1 and 2, there does not exist any pair of rules r and r' such that $head(r) = \{p\}$ and $p \in pos(r')$. Then the intuitive meaning of Steps 3 and 4 is that after forgetting p , any atom's information in rules having p as their heads or positive bodies will be lost because they are all relevant to p , i.e. these atoms either serve as a support for p or p is in part of the supports for these atoms. On the other hand, Step 5 states that any rule containing p in its negation as failure part will be also removed. The consideration for this step is as follows. If we think $neg(r)$ is a part of support of $head(r)$, then when $p \in neg(r)$ is forgotten, $head(r)$'s entire support is lost as well. Clearly, such treatment of negation as failure in forgetting is quite strong in the sense that more atoms may be lost together with $not p$. Therefore we call this kind of forgetting *strong forgetting*.

With a different way of dealing with negation as failure, we have a weak version of forgetting. We define a program to be the result of *weakly forgetting* p in Π , denoted as $WForgetLP(\Pi, \{p\})$, exactly in the same way as in Definition 2 except that Step 5 is replaced by the following step:

$$\begin{aligned} \Pi' &= (\Pi' - \Pi^*) \cup \Pi^\dagger, \text{ where} \\ \Pi^* &= \{r \mid p \in neg(r)\} \text{ and } \Pi^\dagger = \{r' \mid r' : head(r) \leftarrow pos(r), not(neg(r) - \{p\}) \text{ where } r \in \Pi^*\}. \end{aligned}$$

Suppose we have a rule like $r : head(r) \leftarrow pos(r), not neg(r)$ where $p \in neg(r)$. Instead of viewing $neg(r)$ as part of the support of $head(r)$, we may treat it as a default evidence of $head(r)$, i.e. under the condition of $pos(r)$, if all atoms in $neg(r)$ are not presented, then $head(r)$ can be derived. Therefore, forgetting p will result in the absence of p in any case. So r may be replaced by $r' : head(r) \leftarrow pos(r), not(neg(r) - \{p\})$.

Strong and weak forgettings can be easily extended to the case of a set of atoms:

$$\begin{aligned} SForgetLP(\Pi, \emptyset) &= \Pi, \\ SForgetLP(\Pi, P \cup \{p\}) &= \\ SForgetLP(SForgetLP(\Pi, \{p\}), P), \end{aligned}$$

and $WForgetLP(\Pi, P)$ is defined accordingly. The following proposition ensures that our strong and weak forgettings in logic programs are well defined under strong equivalence.

Proposition 1 Let Π be a logic program and p, q two propositional atoms. Then

1. $SForgetLP(SForgetLP(\Pi, \{p\}), \{q\})$ is strongly equivalent to $SForgetLP(SForgetLP(\Pi, \{q\}), \{p\})$;
2. $WForgetLP(WForgetLP(\Pi, \{p\}), \{q\})$ is strongly equivalent to $WForgetLP(WForgetLP(\Pi, \{q\}), \{p\})$.

Example 2 Let $\Pi = \{b \leftarrow a, c, d \leftarrow not a, e \leftarrow not f\}$. Then we have $SForgetLP(\Pi, \{a\}) = \{e \leftarrow not f\}$, and $WForgetLP(\Pi, \{a\}) = \{d \leftarrow, e \leftarrow not f\}$. Now we consider $Forget([\Pi]^C, \{a\})$, which is logically equivalent to formula $(b \vee \neg c \vee d) \wedge (f \vee e)$. Then it is clear that $\models Forget([\Pi]^C, \{a\}) \supset [SForgetLP(\Pi, \{a\})]^C$, and $\models [WForgetLP(\Pi, \{a\})]^C \supset Forget([\Pi]^C, \{a\})$. $\square$

The above example motivates us to examine the deeper relationships between strong and weak forgettings in logic programs and forgetting in propositional theories. Let Π be a program and L a clause, i.e. $L = l_1 \vee \dots \vee l_k$ where each l_i is a propositional literal. We say that L is Π -coherent if there exists a subset Π' of Π and a set of atoms $P \subseteq Atom(\Pi)$ (P could be empty) such that L is a sub-clause of $[Reduct(\Pi', P)]^C$ (i.e. $[Reduct(\Pi', P)]^C$ is a single clause). The intuition behind this notion is to specify those clauses that are parts of clauses generated from program Π through reduction. Consider program $\Pi = \{a \leftarrow b, d \leftarrow a, not c, e \leftarrow not d\}$. Clause $d \vee b$ is Π -coherent, where clause $\neg d \vee e$ is not. Obviously, for each rule $r \in \Pi$, $[r]^C$ is Π -coherent. The following proposition provides a semantic account for Π -coherent clauses.

Proposition 2 Let Π be a program and L a Π -coherent clause. Then either $\models [\Pi]^C \supset L$ or $\models L \supset \Phi$ for some clause Φ where $\models [\Pi]^C \supset \Phi$.

Definition 3 Let Π be a logic program, φ, φ_1 and φ_2 three propositional formulas where φ_1 and φ_2 are in conjunctive normal forms.

1. φ_1 is called a *consequence* of φ with respect to Π if $\models \varphi \supset \varphi_1$ and each conjunct of φ_1 is Π -coherent. φ_1 is a *strongest consequence* of φ with respect to Π if φ_1 is a consequence of φ with respect to Π and there does not exist another consequence φ'_1 of φ ($\varphi'_1 \not\equiv \varphi_1$) with respect to Π such that $\models \varphi'_1 \supset \varphi_1$.
2. φ_2 is called a *premiss* of φ with respect to Π if $\models \varphi_2 \supset \varphi$ and each conjunct of φ_2 is Π -coherent. φ_2 is a *weakest premiss* of φ with respect to Π if φ_2 is a premiss of φ with respect to Π and there does not exist another premiss φ'_2 of φ ($\varphi'_2 \not\equiv \varphi_2$) with respect to Π such that $\models \varphi_2 \supset \varphi'_2$.

Example 3 (Example 2 continued) It is easy to verify that $[SForgetLP(\Pi, \{a\})]^C$ is a strongest consequence of $Forget([\Pi]^C, \{a\})$ and $[WForgetLP(\Pi, \{a\})]^C$ is a weakest premiss of $Forget([\Pi]^C, \{a\})$. In fact, the following theorem confirms that this is always true. $\square$

Theorem 1 Let Π be a logic program and P a set of atoms. Then $[SForgetLP(\Pi, P)]^C$ is a strongest consequence of $Forget([\Pi]^C, P)$ with respect to Π and $[WForgetLP(\Pi, P)]^C$ is a weakest premiss of $Forget([\Pi]^C, P)$ with respect to Π .

Theorem 1 actually provides a precise semantic characterization for strong and weak forgettings in logic programs in terms of the forgetting notion in the corresponding propositional theory.

4 Solving Conflicts in Logic Program Contexts

In this section, we define a general framework called logic program context to represent a knowledge system which consists of multiple agents' knowledge bases. We consider the issue of conflicts occurring in the reasoning within the underlying logic program context. As we will show, the notions of strong and weak forgettings provide an effective way to solve such conflicts.

Definition 4 (Logic program context) A logic program context is a n -ary tuple $\Sigma = (\Phi_1, \dots, \Phi_n)$, where each Φ_i is a triplet $(\Pi_i, \mathcal{C}_i, \mathcal{F}_i)$ - Π_i and $\mathcal{C}_i$ are two logic programs, and $\mathcal{F}_i \subseteq \text{Atom}(\Pi_i)$ is a set of atoms. We also call each Φ_i the i th component of Σ . Σ is consistent if for each i , $\Pi_i \cup \mathcal{C}_i$ is consistent. Σ is conflict-free if for any i and j , $\Pi_i \cup \mathcal{C}_j$ is consistent.

In the above definition, for a given logic program context Σ , each component Φ_i represents agent i 's local situation, where Π_i is agent i 's knowledge base, $\mathcal{C}_i$ is a set of constraints that agent i should comply and will not change in any case, and $\mathcal{F}_i$ is a set of atoms that agent i may forget if necessary. To simplify our following discussion, we assume that for each component Φ_i , the corresponding agent's knowledge base Π_i does not contain constraints (i.e. rules with empty heads). Alternatively such constraints will be contained in the constraint set $\mathcal{C}_i$ though $\mathcal{C}_i$ may also contain rules with nonempty heads.

Now the problem of conflict solving under this setting can be stated as follows: given a logic program context $\Sigma = (\Phi_1, \dots, \Phi_n)$, which may not be consistent or conflict-free, how can we find an alternative logic program context $\Sigma' = (\Phi'_1, \dots, \Phi'_n)$ such that Σ' is conflict-free and is closest to the original Σ in some sense?

Definition 5 (Solution) Let $\Sigma = (\Phi_1, \dots, \Phi_n)$ be a logic program context, where each $\Phi_i = (\Pi_i, \mathcal{C}_i, \mathcal{F}_i)$. We call a logic program context Σ' a solution that solves conflicts in Σ , if Σ' satisfies the following conditions:

1. Σ' is conflict-free;
2. $\Sigma' = (\Phi'_1, \dots, \Phi'_n)$, where $\Phi'_i = (\Pi'_i, \mathcal{C}_i, \mathcal{F}_i)$, and $\Pi'_i = S\text{ForgetLP}(\Pi_i, P_i)$ or $\Pi'_i = W\text{ForgetLP}(\Pi_i, P_i)$ for some $P_i \subseteq \mathcal{F}_i$.

We denote the set of all solutions of Σ as $\text{Solution}(\Sigma)$.

Definition 6 (Ordering on solutions) Let Σ, Σ' and Σ'' be three logic program contexts, where $\Sigma', \Sigma'' \in \text{Solution}(\Sigma)$. We say that Σ' is closer or as close to Σ as Σ'' , denoted as $\Sigma' \preceq_{\Sigma} \Sigma''$, if for each i , $\Phi'_i = (\Pi'_i, \mathcal{C}_i, \mathcal{F}_i) \in \Sigma'$ and $\Phi''_i = (\Pi''_i, \mathcal{C}_i, \mathcal{F}_i) \in \Sigma''$, where $\Pi'_i = S\text{ForgetLP}(\Pi_i, P_i)$ or $\Pi'_i = W\text{ForgetLP}(\Pi_i, P_i)$ for some $P_i \subseteq \mathcal{F}_i$, and $\Pi''_i = S\text{ForgetLP}(\Pi_i, Q_i)$ or $\Pi''_i = W\text{ForgetLP}(\Pi_i, Q_i)$ for some $Q_i \subseteq \mathcal{F}_i$ respectively, we have $P_i \subseteq Q_i \subseteq \mathcal{F}_i$. We denote $\Sigma' \prec_{\Sigma} \Sigma''$ if $\Sigma' \preceq_{\Sigma} \Sigma''$ and $\Sigma'' \not\preceq_{\Sigma} \Sigma'$.

Definition 7 (Preferred solution) Let Σ and Σ' be two logic program contexts. We say that Σ' is a preferred solution of Σ , if $\Sigma' \in \text{Solution}(\Sigma)$ and there does not exist another $\Sigma'' \in \text{Solution}(\Sigma)$ such that $\Sigma'' \prec_{\Sigma} \Sigma'$.

Example 4 Let $\Sigma = (\Phi_1, \Phi_2)$, where

$$\begin{array}{ll} \Phi_1: & \Phi_2: \\ \Pi_1: a \leftarrow, & \Pi_2: c \leftarrow, \\ & d \leftarrow \text{note}, \\ & e \leftarrow c, \\ & f \leftarrow d, \\ \mathcal{C}_1: \leftarrow d, \text{not} f, & \mathcal{C}_2: \leftarrow b, \text{not} c, \\ & \leftarrow \text{not} d, \text{not} f, \\ & b \leftarrow c, \\ \mathcal{F}_1: \{a, b, c\}, & \mathcal{F}_2: \{a, b, c, d, e, f\}. \end{array}$$

It is easy to see that Σ is consistent but not conflict-free because neither $\Pi_1 \cup \mathcal{C}_2$ nor $\Pi_2 \cup \mathcal{C}_1$ is consistent. Now consider two logic program contexts $\Sigma_1 = (\Phi'_1, \Phi'_2)$ and $\Sigma_2 = (\Phi''_1, \Phi''_2)$, where

$$\begin{array}{l} \Phi'_1 = (S\text{ForgetLP}(\Pi_1, \{c\}), \mathcal{C}_1, \mathcal{F}_1), \\ \Phi'_2 = (W\text{ForgetLP}(\Pi_2, \{e\}), \mathcal{C}_2, \mathcal{F}_2), \text{ and} \\ \Phi''_1 = (W\text{ForgetLP}(\Pi_1, \{a, c\}), \mathcal{C}_1, \mathcal{F}_1), \\ \Phi''_2 = (W\text{ForgetLP}(\Pi_2, \{e\}), \mathcal{C}_2, \mathcal{F}_2). \end{array}$$

It can be verified that both Σ_1 and Σ_2 are solutions of Σ , but only Σ_1 is a preferred solution. $\square$

5 Semantic and Computational Properties

In this section, we study some important semantic and computational properties in relation to strong and weak forgettings and conflict solving.

5.1 Semantic Characterizations

We observe that the consistency of program Π does not necessarily imply a consistent $S\text{ForgetLP}(\Pi, P)$ or $W\text{ForgetLP}(\Pi, P)$ for some set of atoms P , and vice versa. For example, consider program $\Pi = \{a \leftarrow, b \leftarrow \text{nota}, \text{not} b\}$, then weakly forgetting a in Π will result in an inconsistent program $\{b \leftarrow \text{not} b\}$. Similarly, strongly forgetting a from an inconsistent program $\Pi = \{b \leftarrow \text{nota}, c \leftarrow b, \text{not} c\}$ will get a consistent program $\{c \leftarrow b, \text{not} c\}$.

To understand why this may happen, we first introduce some notions. Given program Π and a set of atoms P , we specify two programs X and Y . Program X is a subset of Π containing three types of rules in Π : (1) for each $p \in P$, if $p \notin \text{head}(\Pi)$, then rule $r \in \Pi$ with $p \in \text{pos}(r)$ is in X ; (2) for each $p \in P$, if $p \notin \text{pos}(\Pi)$, then rule $r \in \Pi$ with $\text{head}(r) = \{p\} \subseteq X$; and (3) rule $r \in \Pi$ with $\text{neg}(r) \cap P \neq \emptyset$ but not of the types (1) and (2) is also in X . Clearly, X contains those rules of Π satisfying $\text{Atom}(r) \cap P \neq \emptyset$ but will not be affected by $\text{Reduct}(\Pi, P)$. On the other hand, program Y is obtained as follows: for each rule r in X of the type (3), a replacement of r of the form: $r' : \text{head}(r) \leftarrow \text{pos}(r), \text{not}(\text{neg}(r) - P)$ is in Y . It should be noted that both X and Y can be obtained in linear time in terms of the sizes of Π and P .

Theorem 2 Let Π be a program and P a set of atoms. A subset S of atoms occurring in $S\text{ForgetLP}(\Pi, P)$ (or in $W\text{ForgetLP}(\Pi, P)$) is a stable model of $S\text{ForgetLP}(\Pi, P)$

(or $WForgetLP(\Pi, P)$ resp.) iff program $\Pi - X$ (or $(\Pi - X) \cup Y$ resp.) has a stable model S' such that $S = S' - P$.

Theorem 2 presents an interesting result: given program Π and set of atoms P , although computing $SForgetLP(\Pi, P)$ or $WForgetLP(\Pi, P)$ may need exponential time (see Section 5.3), its stable models, however, can be computed through some program that is obtained from Π in linear time.

Now we consider the existence of (preferred) solutions for logic program contexts in conflict solving. It is easy to see that not every logic program context has a preferred solution. For instance, $\Sigma = ((\{a \leftarrow \text{nota}\}, \emptyset, \emptyset))$ has no solution, and hence has no preferred solution neither. The following result shows that the existence of a Σ 's solution always implies the existence of Σ 's preferred solution, and vice versa.

Theorem 3 *Let Σ be a logic program context. Σ has a preferred solution iff $\text{Solution}(\Sigma) \neq \emptyset$.*

Although deciding whether a Σ has a (preferred) solution is NP-hard (see Theorem 6 in section 5.3), we can identify a useful class of logic program contexts whose solutions always exist (we have showed in our full paper that all major logic program update approaches can be transformed into the following form of conflict solving context).

Proposition 3 *Let $\Sigma = (\Phi_1, \dots, \Phi_n)$ be a logic program context. If for each $\Phi_i = (\Pi_i, \mathcal{C}_i, \mathcal{F}_i)$, $\mathcal{C}_i$ is consistent, and for each $r \in \Pi_i$, $\mathcal{F}_i \cap \text{Atom}(r) \neq \emptyset$, then $\text{Solution}(\Sigma) \neq \emptyset$.*

Proof: We show that $\Sigma' = (\Phi'_1, \dots, \Phi'_n)$, where $\Phi'_i = (\emptyset, \mathcal{C}_i, \mathcal{F}_i)$ ($1 \leq i \leq n$) is a solution of Σ . Since for each i , $\mathcal{F}_i \cap \text{Atom}(r) \neq \emptyset$ for each $r \in \Pi_i$, we have $\Pi'_i = SForgetLP(\Pi_i, \mathcal{F}_i) = \emptyset$ (note that this is because we already assumed that Π_i does not contain any rules with empty heads. Instead, this type of rule is contained in $\mathcal{C}_i$). This follows that $\Pi'_i \cup \mathcal{C}_j = \mathcal{C}_j$ for all $j = 1, \dots, n$ are consistent. So $((\emptyset, \mathcal{C}_1, \mathcal{F}_1), \dots, (\emptyset, \mathcal{C}_n, \mathcal{F}_n))$ is a solution of Σ . $\square$

5.2 Representing Logic program Updates

One major advantage of the proposed framework of logic program contexts is that it can represent new conflict solving scenarios for which the traditional logic program update approaches may not handle properly, like the one discussed in section 1 (or Example 4). In fact, our framework can also represent previous logic program update approaches. To illustrate this, we take Sakama and Inoue's update approach [Sakama and Inoue, 1999] as an example (note that we need to restrict their approach to a normal logic program setting).

Definition 8 [Sakama and Inoue, 1999] *Let Π_1 and Π_2 be two consistent logic programs. Program Π' is a SI-result of a theory update of Π_1 by Π_2 if (1) Π' is consistent, (2) $\Pi_2 \subseteq \Pi' \subseteq \Pi_1 \cup \Pi_2$, and (3) there is no other consistent program Π'' such that $\Pi' \subset \Pi'' \subseteq \Pi_1 \cup \Pi_2$.*

Now we transform Sakama and Inoue's theory update into a logic program context. First, for each rule $r \in \Pi_1$, we introduce a new atom l^r which does not occur in $\text{Atom}(\Pi_1 \cup \Pi_2)$. Then we define a program Π'_1 : for each $r \in \Pi_1$, rule $r' : \text{head}(r) \leftarrow \text{pos}(r), \text{not}(\text{neg}(r) \cup \{l^r\})$ is in Π'_1 . That is, for each $r \in \Pi_1$, we simply extend its negative body with a unique atom l^r . This will make each r' in Π'_1 be

removable by strongly forgetting atom l^r without influencing other rules. Finally, we specify $\Sigma_{SI} = (\Phi_1, \Phi_2)$, where $\Phi_1 = (\Pi'_1, \emptyset, \{l^r \mid r \in \Pi_1\})$ and $\Phi_2 = (\emptyset, \Pi_2, \emptyset)$.

For convenience, we also use $\Pi^{-\text{not}P}$ to denote a program obtained from Π by removing all occurrences of atoms in P from the negative bodies of all rules in Π . For instance, if $\Pi = \{a \leftarrow b, \text{not}c, \text{not}d\}$, then $\Pi^{-\text{not}\{c\}} = \{a \leftarrow b, \text{not}d\}$. Now we have the following characterization result.

Theorem 4 *Let Π_1 and Π_2 be two consistent programs, and Σ_{SI} as specified above. Π' is a SI-result of updating Π_1 by Π_2 iff $\Pi' = \Pi^{-\text{not}\{l^r \mid r \in \Pi_1\}} \cup \Pi_2$, where $\Sigma' = ((\Pi, \emptyset, \{l^r \mid r \in \Pi_1\}), (\emptyset, \Pi_2, \emptyset))$ is a preferred solution of Σ_{SI} .*

In our full paper we have showed that other logic program update approaches such as Eiter et al's causal rejection and Dynamic Logic Programming [Eiter et al., 2002; Leite, 2003] for dealing with sequence of programs can also be embedded into our framework. In this sense, the logic program context provides a unified framework for logic program updates.

5.3 Complexity Results

We assume that readers are familiar with the complexity classes of P, NP, coNP, Σ_2^P and $\Pi_2^P = \text{co}\Sigma_2^P$. The class of DP contains all languages L such that $L = L_1 \cap L_2$ where L_1 is in NP and L_2 is in coNP. The class coDP is the complement of class DP (readers refer to [Papadimitriou, 1994] for further details).

We observe that the main computation of strong and weak forgettings relies on the procedure of reduction that further inherits the computation of program unfolding. Hence, in general computing strong and weak forgetting may need exponential steps of rule substitutions in terms of the sizes of the input program and the set of forgotten atoms. However, the following result shows that the inference problem associated to strong and weak forgettings still remains in coNP.

Proposition 4 *Let Π be a logic program, P a set of atoms, and a an atom. Then deciding whether $SForgetLP(\Pi, P) \models a$ (or $WForgetLP(\Pi, P) \models a$) is coNP-complete.*

Proof: (Sketch) The hardness is followed by setting $P = \emptyset$, and the membership can be proved by using Theorem 2. $\square$

Now we consider the complexity of *irrelevance* in relation to strong and weak forgettings and conflict solving. From a semantic viewpoint, the irrelevance tells us whether a strong/weak forgetting or conflict solving procedure will affect some particular atoms occurring in the underlying programs. Hence, studying its associated computational properties is important.

Definition 9 (Irrelevance) *Let Π be a logic program, P a set of atoms, and a an atom. We say that a is irrelevant to P in Π , if either $\Pi \models a$ iff $SForgetLP(\Pi, P) \models a$, or $\Pi \models a$ iff $WForgetLP(\Pi, P) \models a$.*

We generalize the notion of irrelevance to the logic program context. Formally, let Σ be a logic program context and a an atom, we say that a is *derivable* from Σ 's i th component, denoted as $\Sigma \models_i a$, if $\Phi_i = (\Pi_i, \mathcal{C}_i, \mathcal{F}_i) \in \Sigma$ and $\Pi_i \models a$.

Definition 10 (Irrelevance wrt logic program contexts) Let Σ and Σ' be two logic program contexts where $\Sigma' \in \text{Solution}(\Sigma)$, and a an atom. We say that a is irrelevant with respect to Σ and Σ' on their i th components, or simply say that a is $(\Sigma, \Sigma')^i$ -irrelevant, if $\Sigma \models_i a$ iff $\Sigma' \models_i a$.

Theorem 5 Let Π be a logic program, P a set of atoms, a an atom, and Σ and Σ' two logic program contexts where $\Sigma' \in \text{Solution}(\Sigma)$. The following results hold:

1. Deciding whether a is irrelevant to P in Π is coDP -complete;
2. Deciding whether a is $(\Sigma, \Sigma')^i$ -irrelevant is coDP -complete.

Proof: (Sketch) We describe the main idea of proving the hardness part of Result 1. a is irrelevant to P in Π if (1) $\Pi \models a$ iff $S\text{ForgetLP}(\Pi, P) \models a$, or (2) $\Pi \models a$ iff $W\text{ForgetLP}(\Pi, P) \models a$. Here we consider case (1) and proof for case (2) is the same. Let (Φ_1, Φ_2) be a pair of CNFs, where $\Phi_1 = \{C_1, \dots, C_m\}$ and $\Phi_2 = \{C'_1, \dots, C'_n\}$, and each C_i and C'_j ($1 \leq i \leq m$, $1 \leq j \leq n$) are sets of propositional literals respectively. We also assume $\text{Atom}(\Phi_1) \cap \text{Atom}(\Phi_2) = \emptyset$. We know that deciding whether Φ_1 is satisfiable or Φ_2 is unsatisfiable is coDP -complete [Papadimitriou, 1994]. We construct a program Π polynomially based on set $\text{Atom}(\Phi_1) \cup \text{Atom}(\Phi_2) \cup \hat{X} \cup \hat{Y} \cup \{l_1, \dots, l_n, p, a, \text{sat}^{\Phi_1}, \text{unsat}^{\Phi_1}, \text{unsat}^{\Phi_2}\}$, where any two sets of atoms are disjoint and $|\hat{X}| = |\text{Atom}(\Phi_1)|$ and $|\hat{Y}| = |\text{Atom}(\Phi_2)|$. Π consists of four groups of rules where atom p only occurs in Π_4 :

Π_1 : rules to generate all truth assignments of Φ_1 and Φ_2 ;
 Π_2 : rules to derive unsat^{Φ_1} and unsat^{Φ_2} if Φ_1 and Φ_2 are unsatisfiable respectively;
 Π_3 : rules to force a truth assignment of Φ_2 making $\neg\Phi_2$ true if unsat^{Φ_2} is derivable from Π ;
 Π_4 contains 4 rules: $\text{sat}^{\Phi_1} \leftarrow \text{notunsat}^{\Phi_1}$, $a \leftarrow \text{sat}^{\Phi_1}$, $\text{unsat}^{\Phi_2} \leftarrow \text{nota}$, $p \leftarrow$.

Then we can prove that Φ_1 is satisfiable or Φ_2 is unsatisfiable iff $(\Pi \models a \text{ and } S\text{ForgetLP}(\Pi, \{p\}) \models a) \text{ or } (\Pi \not\models a \text{ and } S\text{ForgetLP}(\Pi, \{p\}) \not\models a)$. $\square$

Finally, the following theorem summarizes major complexity results of conflict solving in logic program contexts.

Theorem 6 Let $\Sigma = (\Phi_1, \dots, \Phi_n)$ and $\Sigma' = (\Phi'_1, \dots, \Phi'_n)$ be two logic program contexts, where for each $\Phi_i = (\Pi_i, \mathcal{C}_i, \mathcal{F}_i) \in \Sigma$ ($1 \leq i \leq n$), $\Phi'_i \in \Sigma'$ is of the form $\Phi'_i = (\Pi'_i, \mathcal{C}_i, \mathcal{F}_i)$, where $\Pi'_i = S\text{ForgetLP}(\Pi_i, P_i)$ or $\Pi'_i = W\text{ForgetLP}(\Pi_i, P_i)$ for some $P_i \subseteq \mathcal{F}_i$.

1. Deciding whether Σ has a preferred solution is NP-hard;
2. Deciding whether Σ' is a solution of Σ is NP-complete;
3. Deciding whether Σ' is a preferred solution of Σ is in Π_2^P , if strong and weak forgettings in Σ can be computed in polynomial time¹;

¹In our full paper, we have classified certain classes of logic program contexts, where for each $\Phi_i = (\Pi_i, \mathcal{C}_i, \mathcal{F}_i) \in \Sigma$ and each $P_i \subseteq \mathcal{F}_i$, $S\text{ForgetLP}(\Pi_i, P_i)$ and $W\text{ForgetLP}(\Pi_i, P_i)$ can always be computed in polynomial time.

4. For a given atom a , deciding whether for each $\Sigma'' \in \text{Solution}(\Sigma)$, $\Sigma'' \models_i a$ is in Π_2^P , if strong and weak forgettings in Σ can be computed in polynomial time.

6 Conclusions

In this paper, we defined notions of strong and weak forgettings in logic programs, which may be viewed as an analogy of forgetting in propositional theories. Based on these notions, we developed a general framework of logic program contexts for conflict solving and studied the related semantic and computational properties.

Our work presented in this paper can be extended in several directions. One interesting topic is to associate dynamic preferences to sets of forgettable atoms and components in logic program contexts, so that the extended framework is more flexible to handle task-dependent conflict solving.

References

- [Brass and Dix, 1999] S. Brass and J. Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *Journal of Logic programming*, 40(1):1–46, 1999.
- [Eiter et al., 2002] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic programming*, 2:711–767, 2002.
- [Lang and Marquis, 2002] J. Lang and P. Marquis. Resolving inconsistencies by variable forgetting. In *Proceedings of KR2002*, pages 239–250, 2002.
- [Lang et al., 2003] J. Lang, P. Liberatore, and P. Marquis. Propositional independence - formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.
- [Leite, 2003] J.A. Leite. *Evolving Knowledge Bases: Specification and Semantics*. IOS Press, 2003.
- [Lifschitz et al., 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):426–541, 2001.
- [Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, 1994.
- [Lin, 2001] F. Lin. On the strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128:143–159, 2001.
- [Papadimitriou, 1994] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
- [Sakama and Inoue, 1999] C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Proceedings of LPNMR'99*, pages 2–17, 1999.
- [Su et al., 2004] K. Su, G. Lv, and Y. Zhang. Reasoning about knowledge by variable forgetting. In *Proceedings of KR2004*, pages 576–586, 2004.

LEARNING

Ranking Cases with Decision Trees: a Geometric Method that Preserves Intelligibility

Isabelle Alvarez^(1,2)

(1) LIP6, Paris VI University
4 place Jussieu, F-75005 Paris, France
isabelle.alvarez@lip6.fr

Stephan Bernard⁽²⁾

(2) Cemagref, LISC
F-63172 Aubiere Cedex, France
stephan.bernard@cemagref.fr

Abstract

This paper proposes a new method to rank the cases classified by a decision tree. The method applies *a posteriori* without modification of the tree and doesn't use additional training cases. It consists in computing the distance of the cases to the decision boundary induced by the decision tree, and to rank them according to this geometric score. When the data are numeric it is very easy to implement and efficient. The distance-based score is a global assess, contrary to other methods that evaluate the score at the level of the leaf. The distance-based score gives good results even with pruned tree, so if the tree is intelligible this property is preserved with an improved ranking ability. The main reason for the efficacy of the geometric method is that in most cases when the classifier is sufficiently accurate, errors are located near the decision boundary.

1 Introduction

Decision Trees (DT) are a very popular classification tool because they are easy to build and they provide an intelligible model of the data, contrary to other learning methods. The need for intelligibility is very important in artificial intelligence for applications that are not fully automatic, if there is an interaction with the end-user, expert or not. This is the reason why DT algorithm are widely used for classification purpose (see Murthy [1998] for examples of real world applications). But for some applications knowing the class of each case is not sufficient to make a decision, one needs to compare the cases to one another in order to select the most promising examples. This is often the case in marketing applications, allocation of resources or of grants, etc. (See [Zadrozny and Elkan, 2001] for a description of the charitable donation problem). The traditional idea in this case is to look for the probability of each case to belong to the predicted class, rather than just the class. The cases are then ranked according to the probability-based score. Unfortunately, methods that are highly suitable for probability estimate produces generally unintelligible models. This is the reason why some recent works aim at improving decision tree probability estimate. Smoothing methods are particularly interesting for that purpose. They consist in replacing the raw

conditional probability estimate at the leaf by some corrected ratio that shifts the probability toward the prior probability of the class. The raw conditional probability estimate at the leaf is defined by $p^r(c|x) = \frac{k}{n}$, where k is the number of training cases of the class label classified by the leaf, and n is the total number of training cases classified by the leaf. It is the same for all the cases that are classified by a leaf. The most general type of correction generally used are the m -estimate p^m (see equation (1)), which uses the prior probability of the class and a parameter m , and the Laplace correction p^L which is a particular case of m -correction when all the C classes have the same priors (see [Cestnik, 1990; Zadrozny and Elkan, 2001]).

$$p^m(c|x) = \frac{k + p(c).m}{n + m} \quad p^L(c|x) = \frac{k + 1}{n + C} \quad (1)$$

The main interest of smoothing methods is that they don't modify the structure of the tree. But in order to improve the probability estimate, these methods are often applied to unpruned trees (see [Provost and Domingos, 2003]), so the intelligibility of the model is very much reduced, although it is one of the main interest of decision trees compared to other classifiers (like Naive Bayes, Neural Networks for instance). Ensemble methods like bagging are also used successfully to rank cases, although the margin is not *a priori* an estimate of the class membership. Nevertheless, ensemble methods loose also the intelligibility of the model.

The method we propose here aims firstly at preserving the intelligibility of the model, so the objective is to improve the ranking without modifying the tree itself. This method is based on the computation of the distance of the cases from the decision boundary (the boundary of the inverse image of the different classes in the input space), when it is possible to define a metric on the input space. The distance of a case from the decision boundary defines a score that is specific to each case, unlike other methods for which the score is defined at the level of the leaf and so it is shared by all cases classified by the same leaf. In other geometric methods, like Support Vector Machine (SVM) it has been proved that the distance to the decision boundary can be used to estimate the posterior probabilities (see Platt [2000] for the details in the two-class problem): an additional database is needed in order to calibrate the probabilities. But since in many applications

Database	Size of dataset	ΔN (UT-PT)	N (UT)
bupa	345	19.53±0.68	30.52±0.55
glass	214	2.90±0.16	6.25±0.15
ionosphere	351	5.22±0.25	11.00±0.2
iris	151	1.49±0.12	4.94±0.11
letter	20000	31.71±0.76	61.10±0.58
newThyroid	215	2.95±0.19	7.25±0.13
optdigits	5620	9.14±0.35	19.35±0.3
pendigits	10992	8.91±0.35	23.33±0.3
pima	768	32.88±1.11	47.2±0.96
sat	6435	13.67±0.40	24.73±0.31
segmentati	210	1.48±0.13	4.66±0.09
sonar	208	6.30±0.24	11.36±0.15
vehicle	846	8.04±0.29	18.10±0.25
vowel	990	2.50±0.19	8.56±0.14
wdbc	569	5.10±0.23	10.03±0.18
wine	178	1.43±0.10	4.22±0.09

Table 1: Comparison of the size of pruned (PT) and uncollapsed unpruned (UT) trees: Mean and standard deviation of the difference of the number of leaves N over 100 resamples.

we don’t need the exact posterior probability, it is generally possible to use directly the score induced by the distance to rank and to select the most interesting cases.

The paper is organized as follow: Section 2 examines from the intelligibility viewpoint the methods applied to decision trees to rank cases or to estimate posterior probabilities. Section 3 presents our method for obtaining a distance-based score, and it explains why it is interesting from a theoretical point of view. Section 4 presents the experimental results which have been drawn from the numerical databases of the UCI repository, in comparison with the results obtained from the smoothing methods applied on the same databases. We make further comments about geometric score and hybrid method in the concluding section.

2 Decision Tree methods for ranking: the intelligibility viewpoint

The success of Decision Trees as classification method is for a good part due to the intelligibility of the model produced by the algorithms. Pruning methods [Breiman *et al.*, 1984; Bradley and Lovell, 1995; Esposito *et al.*, 1997] produce shorter trees with at least the same performance than longer trees, since the generalization performance are enhanced. They also produce shorter tree on purpose, seeking for a compromise between accuracy (or other performance criteria) and the size of the tree. Table 1 shows that unpruned trees can be very large compared to pruned trees with similar accuracy (the mean absolute difference over the databases is 0.38% and it is always less than 2.3%). Because of this size problem, it is desirable to improve the probability estimate given by DT, in order to allow a compromise between size and ranking ability.

With smoothing methods the probability estimate is the same for all the examples classified by a leaf. In order to produce more specific probability estimates, other methods learn directly the probability class membership at the leaf. For instance, [Smyth *et al.*, 1995] use kernel-based density

estimator at the leaf, without modification of the tree structure. This method improves significantly the class probability estimates. But the practical use of kernel density estimator is limited to very low dimension, and the setting of parameters is not easy. Kohavi [1996] builds Naive Bayes classifiers at the level of the leaf, using its own induction algorithm. The objective of the tree partition is not to separate the classes but to segment the data so that the conditional independence assumption is better verified. The size of the tree is limited to cover each leaf with enough data. In our experiment the size of the Naive Bayes Trees (NBT) is comparable to the size of the pruned trees (but the segmentation of the space is completely different). With different objectives and structures, the interpretation of DT and NBT cannot compare easily.

Other methods try to correct the probability estimate at each nodes by propagating a case through the different possible path from each node. These methods, like fuzzy trees [Umano *et al.*, 1994], fuzzy split [Quinlan, 1993], or more recently [Ling and Yan, 2003] deal with a different issue: Managing the uncertainty in the input case and in the training database. Generally the computation of the probability estimate is very complex and in some cases difficult to understand: a lot of nodes can be involved, although non-convex area of the input space corresponding to one class can be divided arbitrarily into several leaves. So from the point of view of intelligibility these methods are not totally convincing.

We propose here to keep the structure of the pruned tree but to rank the cases accordingly to their distance from the decision boundary which is defined by the tree.

3 Distance ranking methods for decision trees

We consider here axis-parallel DT (ADT) operating on numerical data: Each test of the tree involves a unique attribute. We note Γ the decision boundary induced by the tree. Γ consists of several pieces of hyperplanes which are normal to axes.

We consider a multi-class problem, with a class of interest c (the positive class). Let x be a case, $c(x)$ the class label assigned to x by the tree, $d = d(x, \Gamma)$ the distance of x from the decision boundary Γ . We use the distance of an example from the decision boundary to define its geometric score.

3.1 Global and local geometric ranking

Definition 1 Geometric score

The geometric score $g(x)$ of x is the distance of x from the decision boundary if $c(x) = c$ and its opposite otherwise.

$$g(x) = \begin{cases} d(x, \Gamma) & \text{if } c(x) \text{ is the positive class,} \\ -d(x, \Gamma) & \text{otherwise.} \end{cases} \quad (2)$$

Theorem 1 Global geometric ranking

The geometric score induce a quasi-order $\succeq$ over the examples classified by the tree.

$$x \succeq y \Leftrightarrow g(x) \geq g(y) \quad (3)$$

Cases are ranked in decreasing order relatively to the geometric score. The most promising cases have the highest geometric score, which means that their predicted class is the positive one and that they are far from the decision boundary.

With the geometric score, examples are ranked individually, not leaf by leaf.

The geometric score is specific to each example, so it is also possible to first rank the leaves with a smoothing method (or an equivalent method that ranks the leaves, not the cases) and then to rank the cases inside a leaf.

Theorem 2 Local geometric ranking

The geometric score induce a quasi-order $\succeq_L$ over the examples classified by a leaf.

$$x \succeq_L y \Leftrightarrow \text{or} \begin{cases} p(c|x) > p(c|y) \\ p(c|x) = p(c|y) \text{ and } g(x) \geq g(y). \end{cases} \quad (4)$$

With the local geometric score, the leaves are ranked according to the probability estimate, then inside each leaf (or inside each group of leaves with the same output probability estimate), examples are ranked according to their geometric score.

The distance of a case x to the decision boundary is computed with the algorithm described in [Alvarez, 2004]. It consists in projecting x onto all the leaves f which class label differs from $c(x)$. The nearest projection gives the distance.

Algorithm 1 distanceFrom(x,DT)

0. $d = \infty$;
1. Gather the set F of leaves f which class $c(f) \neq c(x)$;
2. For each $f \in F$ do: {
3. compute $p_f(x) = \text{projectionOntoLeaf}(x, f)$;
4. compute $d_f(x) = d(x, p_f(x))$;
5. if $(d_f(x) < d)$ then $d = d_f(x)$ }
6. Return $d = d(x, \Gamma)$

Algorithm 2 projectionOntoLeaf(x, f = (T_i)_{i∈I})

1. $y = x$;
2. For $i = 1$ to $\text{size}(I)$ do: {
3. if y doesn't verify the test T_i then $y_u = b$ }
where T_i involves attribute u with threshold value b
4. Return y

The projection onto a leaf is straightforward in the case of ADT since the area classified by a leaf f is a hyper-rectangle defined by its tests. The complexity of the algorithm is in $O(Nn)$ in the worst case where N is the number of tests of the tree and n the number of different attributes of the tree.

3.2 Theoretical viewpoint

We expect geometric ranking to give interesting results when errors occur near the decision.

If this property is verified, positive cases (cases which class is the class of interest) that are not recognized have negative geometric score but with a small absolute value. False positive, that is negative cases classified as positive have also small (but positive) geometric score. On the contrary, true positive have higher geometric score and true negative have negative geometric score with high absolute value. So independently from the score estimated at the leaf, the geometric score tends to bring side by side false negative and false positive, and to repel true positive and true negative. This can be seen on a Receiver Operating Characteristic (ROC) curve (in the way described in [Adams and Hand, 1999]). The ratio of

positive examples is plotted against the ratio of all other (negative) examples as the score varies. With methods that give constant probability estimates at the leaf, the points are plotted from one leaf to another. The affine interpolation between consecutive points assumes that examples are selected randomly inside a leaf (or a set of leaves with the same score). If we use a ROC curve to visualize the ranking, geometric ranking will be very good at the beginning of the curve, as seen in Figure 1.

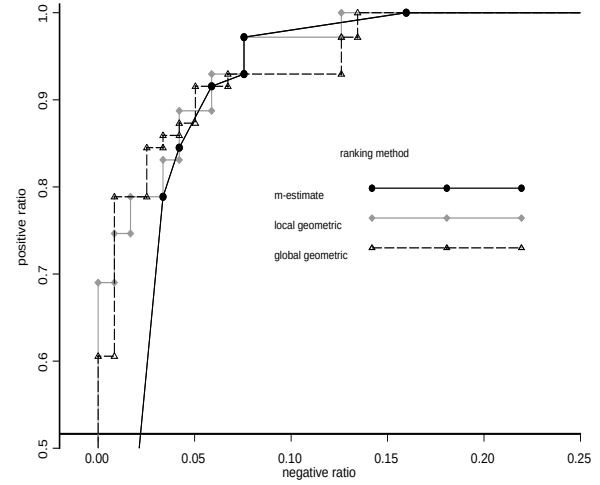


Figure 1: ROC curves of different ranking method (WD Breast-Cancer sample). The local geometric ranking curve intersects the basic m -estimate curve at leaf points.

Since DT algorithms are generally designed to maximize accuracy, it is not unreasonable to hypothesize that errors lie near the decision boundary. In a very ideal case, it is even possible to demonstrate that this hypothesis is true.

A DT builds a partition in the input space. In a two class problem, it is possible to associate to a DT a unique function $g : E \mapsto \{0, 1\}$ such that $g(x)$ is the predicted class of x . (But several decision trees are associated to the same function). We consider an ideal case of statistical decision, where the joint distribution P of observations would be uniform on the graph of the indicator function f of a set S in $E = [0, 1]^n$. We also suppose that the size of all the maximal hypercubes in S and $E \setminus S$ has a lower bound $\nu > 0$ (to prevent pathological situation for S and its boundary). In this case, decision trees built on samples drawn from P can approach f as closely as wanted if the size of the sample can grow indefinitely. For this particular case of function f , errors are located near the decision boundary.

Theorem 3 Proximity of errors. For a DT which associated function g is close enough to f , errors are near the decision boundary ∂g of g .

If we note A the area where $f - g \neq 0$ (set of errors), $\dot{A}$ the interior of A , and if we consider ϵ small enough so that $\epsilon < \nu^n$, we have:

$$\left(\int_E |f - g| < \epsilon \text{ and } x \in \dot{A} \right) \Rightarrow d(x, \partial g) < \alpha_n \sqrt[n]{\epsilon} . \quad (5)$$

Proof. Let x be in $\hat{A}$, we consider $B(x, \frac{d}{2})$ the maximal hypercube centered at x in its connected component. The volume of B is included into $\int_E |f - g|$, so we have: $d^n < \epsilon < \nu^n$. So the size of B is smaller than ν and $d < \sqrt[n]{\epsilon}$. The boundary of B , ∂B , encounters the decision boundary of $f - g$ on at least two meeting points of two different hyperplanes, since B is maximal. If f is not constant on B , then both ∂f and ∂g cross B , and so necessarily $d(x, \partial g) < \frac{d}{2}\sqrt{n}$. If f is constant on B , since the size of B is smaller than ν , at least one of the meeting point lies on ∂g (otherwise the size of B would be smaller than the lower bound of the maximal balls). So once again $d(x, \partial g) \leq \frac{d}{2}\sqrt{n} < \frac{\sqrt{n}}{2} \sqrt[n]{\epsilon}$. •

Even if real conditions are very far from this ideal case (in first place, generally f doesn't exist), we can test if the hypothesis of proximity of errors is generally verified. Table 2 shows the mean of the difference of the mean distance of correctly classified cases (hits) and errors from the decision boundary. We also computed for each sample λ , the inverse of the coefficient of variation for the difference of the means defined by (6), where d_h and σ_h are the mean and the standard deviation of the distance of correctly classified examples from the decision boundary, and d_e and σ_e the same magnitude for error examples.

$$\lambda = \frac{d_h - d_e}{\sqrt{\sigma_h^2 + \sigma_e^2}} \quad (6)$$

Table 2 shows the percentage of the samples for which $\lambda \geq 2$, which is the 97.5% confidence coefficient under the normal assumption (the test is unilateral). We can see that errors are closer from the decision boundary for a majority of databases. Datasets for which this property is not verified have generally a low mean accuracy (62% for bupa and 69% for sonar). If we consider only the samples for which the accuracy is better than 70%, the proportion shifts to 29% and 50% respectively.

Database	Δ of the means	λ	% of samples with $\lambda \geq 2$
bupa	0.009±0.002	0.86±0.12	17
glass	0.038±0.011	2.28±0.34	51
ionosphere	0.037±0.006	1.36±0.19	35
iris	0.092±0.003	5.34±0.28	95
newThyroid	0.059±0.002	3.46±0.14	91
optdigits	0.542±0.008	24.87±1.45	100
pendigits	0.339±0.005	24.6±1.24	100
pima	0.036±0.001	4.15±0.15	94
sat	0.135±0.005	12.59±0.56	100
segment.	0.181±0.005	7.28±0.2	98
sonar	0.027±0.003	1.44±0.14	34
vehicle	0.026±0.004	4.75±0.49	55
vowel	0.215±0.003	16.88±0.74	100
wdbc	0.113±0.002	10.24±0.29	100
wine	0.152±0.004	6.73±0.39	97.7

Table 2: Comparison of the mean distance of errors and hits to the decision boundary, over the test bases of 100 samples per database. The mean of the difference is estimated for each sample. (Bad results are bold)

A corollary of theorem 3 is that if a tree is not accurate, errors may lie everywhere, not only near the decision boundary.

In this case the geometric score cannot be good. So we expect the geometric score to be better with more accurate trees.

4 Experimental Results

4.1 Experimental Design

We have studied the geometric ranking on the database of the UCI repository [Blake and Merz, 1998] that have numerical attributes only and no missing values. We are not directly concerned in this study with the problem of the prevalence of the positive class, since our method doesn't build the decision tree: it applies on the grown tree. So we didn't pay any particular attention to the relative frequency of the classes in the datasets. We chose as positive class either the class with the lowest frequency in the database, either a class which grouped together several classes when it was more logical. When the classes were equiprobable and with no particular meaning we chose it randomly. Although there is a lot of work on the analysis of multi-class problem, for simplicity we have treated multi-class problem as a two class problem (class of the examples were modified before growing the trees).

For each database, we divided 100 bootstrap samples into separate training and test sets in the proportion 2/3 1/3, respecting the prior of the classes (estimated by their frequency in the total database). Even if it is not the best way to build accurate trees for unbalanced dataset or different error costs, here we are not interested in building the most accurate or efficient tree, we just want to study the effect of geometric ranking on pruned trees. For the same reason we grow trees with the default options of j48 (Weka's [Witten and Frank, 2000] implementation of C4.5) although in many cases different options would build better trees. For unpruned trees we disabled the collapsing function.

We used Laplace correction and m-estimate smoothing methods to correct the raw probability estimate at the leaf for reduced-error pruned tree and normal pruned tree. The value of m was chosen such that $m \times p(c) = 10$ where $p(c)$ is the prior probability of the class of interest (as suggested in [Zadrozny and Elkan, 2001]).

We used two different metrics in order to compute the distance from the decision boundary, the Min-Max (MM) metric and the standard (s) metric. Both metrics are defined with the basic information available on the data: An estimate of the range of each attribute i or an estimate of its mean E_i and of its standard deviation s_i . The new coordinate system is defined by (7).

$$y_i^{MM} = \frac{x_i - \text{Min}_i}{\text{Max}_i - \text{Min}_i} \quad \text{or} \quad y_i^s = \frac{x_i - E_i}{s_i} \quad (7)$$

The parameters of the metric are estimated on each sample. The choice of the metric has a very limited effect on the geometric score; If we measure the difference between the Area Under the ROC curve (AUC), for each database, it is always less than $2 \cdot 10^{-3} \pm 9 \cdot 10^{-4}$, except for the thyroid and vehicle databases (less than $4 \cdot 10^{-3}$) and the glass database ($9.5 \cdot 10^{-3}$).

4.2 Comparison between distance-based ranking and smoothing methods

The geometric score is only used to rank the examples without changing the tree structure. It is not used to estimate the

Dataset	Red.-Error pruning	Normal pruning	No pruning	NBTree
bupa	0.46±0.48	-0.14±0.47	-0.82±0.50	0.08 ±0.79
glass	1.78±0.72	-0.39±0.75	-1.87±0.73	-2.01±0.83
iono.	-1.11±0.4	-2.30±0.4	-2.85±0.42	-5.14± 0.72
iris	4.69±0.40	3.85±0.35	3.67±0.37	1.56±0.43
letter	0.18±0.09	0.37±0.07	-0.26±0.05	0.40±0.12
thyroid	4.48±0.43	3.08±0.38	2.54±0.38	-2.13±0.62
optdig.	0.53±0.08	0.34±0.06	0.07±0.06	-0.12±0.06
pendig.	0.46±0.04	0.40±0.03	0.28±0.03	0.58±0.05
pima	1.34±0.43	-0.98±0.25	-1.07±0.30	2.25±0.55
sat	1.01±0.09	0.89±0.07	0.46±0.05	0.97±0.09
segment.	8.16±0.75	5.27±0.63	5.31±0.64	3.25±0.61
sonar	2.55±0.47	1.99±0.51	1.80±0.49	-5.01±1.03
vehicle	0.35±0.16	0.64±0.14	-0.12±0.16	0.78±0.30
vowel	4.18±0.34	3.09±0.29	2.83±0.26	0.78±0.44
wdbc	3.75±0.24	2.24±0.18	2.15±0.18	2.09±0.22
wine	5.35±0.45	3.32±0.30	2.91±0.30	-0.06±0.25

Table 3: Absolute difference of the AUC between global geometric ranking with standard metric and smoothing methods at the leaf. The last column shows the difference between global geometric ranking on Red.-error pruning tree with NBTree. (All mean values and standard deviations are $\times 100$. Insignificant values are bold)

posterior probability of an example, so the appropriate measure of performance in that case is the AUC. Table 3 shows the difference between global geometric ranking and Laplace or m-estimate correction at leaf.

Apart from a few cases, global geometric ranking gives better values than either Laplace or m-estimate correction (with a 95% confidence coefficient). The differences are relatively small (from 0.004 to 0.08), but since they are absolute values the improvement can be important. We have also shown the difference of the AUC between global geometric ranking on reduced-error pruned tree and NBTree.

Table 4 shows the difference between local geometric ranking and smoothing correction at leaf. Local geometric ranking is always better (with a 95% confidence coefficient) than smoothing method alone, except in one case which is not significant. But like for global ranking, the improvement can vary a lot (absolute value from 0.002 to 0.078).

As we said in the theoretical viewpoint section, we expect geometric ranking to outperform smoothing method at the beginning of the ROC curve. To measure the relative behavior of ROC curves for increasing value of the negative ratio, we have computed $AUC(x)$, $0 \leq x \leq 0.5$, the integral function of the ROC curve, with a 0.001 step value, for the global geometric score (g) and the smoothing correction (s). Table 5 shows for normal pruned trees the maximum abscissa value x such that $AUC_g(y) \geq AUC_s(y)$ with a confidence coefficient of 0.95 (under the normal assumption) for every $y \leq x$. For all smaller values of the negative ratio, the global geometric ranking outperforms the other method (in term of AUC).

We can see in Table 5 that for most bases, the global geometric ranking methods is rather efficient at the beginning of the ROC curve, even when on the total range it performs badly (like for the Pima database, see Table 3). The

Dataset	Reduced-error pruning	Normal pruning	Unpruned
bupa	1.75±0.23	0.88±0.09	0.69±0.09
glass	4.21±0.44	3.39±0.34	2.61±0.31
iono	<i>0.04±0.29</i>	0.51±0.17	0.49±0.18
iris	3.71±0.25	3.31±0.26	2.92±0.25
letter	0.19±0.09	0.36±0.07	0.13±0.02
thyroid	3.62±0.39	2.70±0.29	2.33±0.27
optd.	0.67±0.06	0.43±0.04	0.27±0.03
pend.	0.34±0.03	0.28±0.02	0.20±0.02
pima	2.59±0.26	0.87±0.07	0.55±0.05
sat	1.09±0.08	0.89±0.06	0.45±0.04
segment.	7.78±0.72	4.83±0.54	4.34±0.49
sonar	3.02±0.29	2.81±0.21	2.70±0.2
vehicle	0.53±0.09	0.60±0.07	0.49±0.05
vowel	3.83±0.3	2.85±0.26	2.56±0.23
wdbc	3.75±0.22	2.14±0.14	2.04±0.14
wine	5.11±0.4	3.14±0.26	2.80±0.24

Table 4: Absolute difference of the AUC between local geometric ranking with standard metric and the best smoothing method. (All mean values and standard deviations are $\times 100$. Insignificant values are italic. There is no bad value.)

experiment partially confirms the theoretical viewpoint concerning the fact that geometric score gives interesting results when misclassified examples are near the decision boundary. This is particularly true for the bupa (liver-disorder) and ionosphere databases. Table 2 shows that these datasets doesn't verify the hypothesis of proximity of errors on a majority of samples, and actually the global geometric score give bad results for these datasets.

Concerning the improvement of the geometric ranking when the accuracy of the tree is better, the experiment is not conclusive. If we compute Table 3 and Table 4 for a subset of the samples, the best quartile for tree accuracy, the global geometric ranking is not improved (results are not significant). But local geometric ranking gives always better results than on the total sample, except on the glass and ionosphere database (for which the hypothesis of proximity of errors is not much improved on the subset of the samples).

5 Conclusion

We have presented in this article a geometric method to rank cases that are classified by a decision tree. It applies to every axis-parallel tree that classifies examples with numerical attributes. We were not concerned here with the problem of growing the tree (problem with unbalanced datasets or different misclassification costs which lead to pre-processing of the data or new pruning methods). The geometric method doesn't depend on the type of splitting or pruning criteria that is used to build the tree. It only depends on the shape of decision boundary induced by the tree. It consists in ranking the case according to their distance to the decision boundary, taking into account the class of interest and the class that is predicted by the decision tree. Theoretical arguments suggest that this method is interesting when the misclassified examples lie near the decision boundary, and this was partially confirmed by the experimentation. The combination of geomet-

Dataset	MM metric		Standard metric	
	m-estimate	Laplace	m-estimate	Laplace
bupa	0.001	0.001	0.001	0.001
glass	0.1	0.1	0.05	0.05
iono	0.02	0.02	0.02	0.02
iris	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
thyroid	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
optdigits	0.01	0.01	0.01	0.01
pendigits	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
pima	0.02	0.35	0.03	0.37
sat	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
segment.	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
sonar	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
vehicle	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
vowel	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
wdbc	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5
wine	≥ 0.5	≥ 0.5	≥ 0.5	≥ 0.5

Table 5: Abscissa below which the global geometric ranking AUC is always greater. (Bad results are bold).

ric ranking and smoothing methods almost always improve the global ranking (measured with the AUC). Different kind of experiment should be performed in order to compare geometric ranking (and particularly local geometric ranking) to NBTree or other algorithm: since the structure of the trees are different, the choice of pruning method can be important.

The main limit of the method is that it is limited to numerical attributes. It could be extended to ordered attributes, but without the definition of a utility function it cannot be used with attributes that have unordered modalities.

Further work is in progress in order to understand more precisely when the geometric ranking should perform well. Following the idea from [Smyth *et al.*, 1995], we think that density estimator could be used on the distance itself rather than on the attribute of the cases, in order to deal with 1-dimension estimator (which are very efficient). Another interesting point is the definition of a geometric score for real multi-class problem (with no particular class of interest). Actually the algorithm that computes the distance to the decision boundary computes already the distance of an example to the different classes, so these distances could be used for that purpose.

References

- [Adams and Hand, 1999] N. M. Adams and D. J. Hand. Comparing classifiers when the misallocation costs are uncertain. *Pattern Recognition*, 32(7):1139–1147, 1999.
- [Alvarez, 2004] Isabelle Alvarez. Explaining the result of a decision tree to the end-user. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 119–128, Valencia, Spain, Aout 2004. Morgan Kaufmann.
- [Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [Bradley and Lovell, 1995] Andrew P. Bradley and Brian C. Lovell. Cost-sensitive decision tree pruning: Use of the roc curve. In *Eighth Australian Joint Conference on Artificial Intelligence*, pages 1–8, 1995.
- [Breiman *et al.*, 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
- [Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 147–149, 1990.
- [Esposito *et al.*, 1997] F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.
- [Kohavi, 1996] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.
- [Ling and Yan, 2003] C. X. Ling and R. J. Yan. Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning*, pages 480–487, 2003.
- [Murthy, 1998] S.K. Murthy. A automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.
- [Platt, 2000] J. Platt. Probabilistic outputs for support vector machines. In Bartlett P. Schoelkopf B. Schuurmans D. Smola, A.J., editor, *Advances in Large Margin Classifiers*, pages 61–74, Cambridge, Massachusetts, 2000. MIT Press.
- [Provost and Domingos, 2003] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.
- [Quinlan, 1993] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.
- [Smyth *et al.*, 1995] Padhraic Smyth, Alexander Gray, and Usama M. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *International Conference on Machine Learning*, pages 506–514, 1995.
- [Umamano *et al.*, 1994] M. Umamano, K. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umezue, and J. Kinoshita. Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *3rd IEEE International Conference on Fuzzy Systems*, pages 2113–2118, 1994.
- [Witten and Frank, 2000] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.
- [Zadrozny and Elkan, 2001] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proc. 18th International Conf. on Machine Learning*, pages 609–616. Morgan Kaufmann, 2001.

Exploiting Informative Priors for Bayesian Classification and Regression Trees

Nicos Angelopoulos and James Cussens

Department of Computer Science

University of York

Heslington, York YO10 5DD, UK

{nicos,jc}@cs.york.ac.uk

Abstract

A general method for defining informative priors on statistical models is presented and applied specifically to the space of classification and regression trees. A Bayesian approach to learning such models from data is taken, with the Metropolis-Hastings algorithm being used to approximately sample from the posterior. By only using proposal distributions closely tied to the prior, acceptance probabilities are easily computable via marginal likelihood ratios, whatever the prior used. Our approach is empirically tested by varying (i) the data, (ii) the prior and (iii) the proposal distribution. A comparison with related work is given.

1 Introduction

A key feature of the Bayesian approach to statistical inference is that prior knowledge (i.e. relevant information distinct from the data) can be incorporated into the learning process in a mathematically rigorous and conceptually clear manner. On the assumption that a space of possible statistical models can be defined, prior knowledge is expressed via a *prior distribution* over this space. This distribution is then updated with the data using Bayes theorem to produce a posterior distribution. The posterior distribution is the end result of learning and can then be used to make predictions about future data or, once suitably summarised, to give the data analyst insight into the domain of interest.

The Bayesian framework is compellingly simple, but there are many complexities in applying it to real data analysis. Many of these difficulties are computational. For example, in high-dimensional spaces the posterior will be too complex to report directly so features of the distribution (e.g. mean and variance) must be extracted—requiring, in general, a difficult integration. These problems are being progressively alleviated by increased computing power and the subsequent use of Markov chain Monte Carlo (MCMC) techniques to approximately sample from the posterior.

In this paper an existing Bayesian method for defining and using a very flexible class of prior distributions over models is further developed and applied specifically to classification and regression tree (C&RT) models. C&RT models are so well known in machine learning (and increasingly statistics)

that this paper assumes familiarity with the basic properties of these models.

The method is fully Bayesian using a version of the Metropolis-Hastings MCMC algorithm to approximately sample from posterior distributions over the space of all possible C&RT models. Even after eliminating *a priori* uninteresting trees (for example, those containing sparsely populated leaves) this will be a very large space. The key feature of our approach is that the user can declare structural prior knowledge to restrict and/or bias the prior distribution over this space.

The rest of the paper is structured as follows. Section 2 describes our language for defining priors. Section 3 describes the specific priors that have been used for Bayesian inference of C&RT models. Section 4 presents our version of the Metropolis-Hastings (MH) algorithm and gives a convergence result for one special case. Section 5 gives a representative sample of the results of our experiments. The paper concludes with a comparison with related work (Section 6) followed by pointers to future work (Section 7).

2 Defining priors with stochastic logic programs

Our approach to defining priors is related to a line of research which was independently initiated by [Chipman *et al.*, 1998] and [Denison *et al.*, 1998]. The basic idea is to specify a prior distribution over a space of models with a *stochastic program* rather than by some closed-form expression:

Instead of specifying a closed-form expression for the tree prior, $p(T)$, we specify $p(T)$ implicitly by a tree-generating stochastic process. Each realization of such a process can simply be considered a random draw from this prior. [Chipman *et al.*, 1998]

A convenient language for defining such stochastic programs is *stochastic logic programming* [Muggleton, 1996]. A stochastic logic program (SLP) can be used to define a prior over a given space of statistical models by a two-step process. Firstly, a standard logic program is written which defines the desired model space (the *hypothesis space* in machine learning parlance). We simply define a unary predicate `cart/1` in a logic program such that each instantiation for `T` which makes `cart(T)` true is a term in first-order logic which represents a C&RT model. We can exploit the expressiveness

of first-order logic to represent C&RT models, in whichever logical representation is most convenient.

At this point any C&RT models we wish to exclude from the hypothesis space can be so excluded by adding the appropriate constraints to the logic program. For example, suppose the user is faced with a classification task and knows (or wishes to assume) that the distribution over classes is constant over some rectangular region (a ‘box’) of the attribute space. It follows that the box must be contained within a leaf of the ‘true’ tree so only trees with a leaf containing the box need be considered. In general, the user may wish to declare several such boxes. We have effected such constraints in our experiments by declaring facts such as those given in Fig 1, and altering the definition of `cart/1` to check that boxes determined by our declarations are not split in any tree.

```
%const_class_probs(boxname,var,min,max).
const_class_probs(box28,x2,-inf,127).
const_class_probs(box28,x8,-inf,28).

const_class_probs(box26,x2,128,inf).
const_class_probs(box26,x6,-inf,29.8).
```

Figure 1: Declaring rectangular regions within which the distribution over classes is constant.

The stochastic element of the SLP is most easily understood by first considering how the logic programming language Prolog searches for logical consequences of a logic program. Given a *query* such as `:-cart(T)` (“find *T* such that `cart(T)` is true”), Prolog will search for suitable *T*s with a deterministic depth-first search. When there is a choice of rules each of which might lead to a proof, Prolog always chooses whichever rule occurs first in the logic program. If this rule leads to a dead-end, then Prolog *backtracks* and tries the second rule, and so on.

The basic SLP idea is simple: probability labels are attached to rules in the logic program thus converting it into an SLP. Now when the search for proofs has to choose between rules, a rule is chosen probabilistically according to the probability labels. In this paper, if a chosen rule does not lead to a proof, then backtracking will try from amongst untried rules, revisiting the most recent choice-points first and renormalising the probabilities as necessary: we call this approach, introduced by [Cussens, 2000], *backtrackable sampling*.

Now when the query `:-cart(T)` is asked, there is a distribution over possible instantiations for *T*. Since each *T* represents a C&RT model, this provides a prior distribution over this model space.

In this paper we have extended the representational power of SLPs so that probability labels may be computed ‘on the fly’. SLP rules can now be of the form `ExpProb : Vars : Rule`. At each point at which *Rule* is a possible rule to use, it is required that *Vars* is a ground list and that `ExpProb` is a probability computable from *Vars*. Fig 2 has an example SLP where the probability of growing a tree by splitting a leaf depends on the depth of that leaf.

SLPs define a distribution over proofs, and to understand this distribution it is useful to represent each proof as a *proof*

```
1 - 1/D : [D] : cart(leaf).
1/D : [D] : cart(Split-[L,R]) :-
    NewD is D+1, splt(Split),
    [NewD] : cart(L), [NewD] : cart(R).

0.4:splt(x1). 0.3:splt(x2). 0.3:splt(x3).
```

Figure 2: Prior distribution over C&RT models where the probability of splitting a leaf depends on its depth *D*.

tree. Fig 3 displays one such proof tree which proves that `cart(x1-[x2-[leaf,leaf],leaf])` is true, i.e. that this 3-leaf tree is in our model space. (Figs 2 and 3 use an abbreviated representation of C&RT models, with split values omitted.) Each proof tree simply consists of the rules and facts used in the proof ‘bolted together’ by equality constraints. The initial query appears in an equality constraint in the root node. Conceptually, SLPs execute by probabilistically growing a proof tree, backtracking if the constraints become insoluble.

To sum up, our priors are defined by a sampling algorithm within a logical framework. The sampling algorithm could, of course, be implemented in any programming language. It involves progressively building up a data structure representing the C&RT model by (i) making choices, (ii) checking that constraints are not violated and (iii) backtracking if they have. The motivation for a logic programming implementation is that procedures (ii) and (iii) are built-in, and that logic is a convenient language for declaring constraints (such as our boxes).

Not all distributions can be directly sampled from (hence MCMC), so such distributions cannot be effectively represented by defining a direct sampling algorithm. However, when a statistical model can be viewed as the end result of some generative process, a generative description is well-motivated. Bayes nets can be generated by adding arcs [Angelopoulos and Cussens, 2001] and C&RT models by splitting leaves. Our addition of backtracking extends the applicability of this generative view.

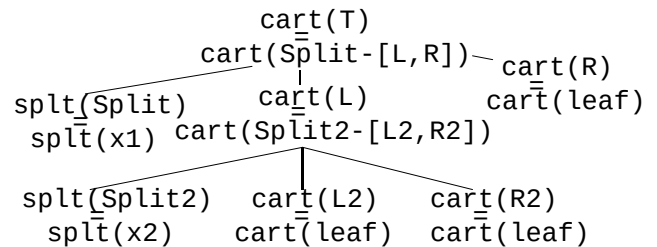


Figure 3: A proof tree which proves `cart(x1-[x2-[leaf,leaf],leaf])`

3 Priors for Bayesian C&RT

In this section we describe the actual priors we have used for the experiments described in Section 5. Fixing some notation, note that a given C&RT model *t* maps an example, described

by a set of p predictor variables $x = x_1, x_2, \dots, x_p$, to a distribution over a response variable y . In the case of classification trees (also known as decision trees) y is discrete and finite, and in the case of regression trees y is continuous. Some C&RT implementations throw away information given by the distribution of y and simply return the majority class or mean value as appropriate, but this is just a design decision. Fundamentally it is a distribution over y that is given.

It follows that each tree t defines a conditional probability distribution $P(y|t, x)$. This conditional distribution is the likelihood function of the following version of Bayes theorem: $P(t|y, x) = P(t|x)P(y|t, x)/P(y|x)$ whose LHS gives the desired posterior distribution over trees. Note that the prior $P(t|x)$ is only prior to y , it is conditional on the predictor variables so that observed values of x in the data can be used to define sensible priors.

Importantly, this allows us to rule out *a priori* trees with leaves which contain few examples. There is nothing to stop a user allowing trees with sparsely populated leaves, but ruling out such trees is normal in this research area. It ensures that the space of possible tree structures is a reasonable (finite) size. In our experiments only trees whose leaves have at least 5 examples are permitted, following [Denison *et al.*, 1998]. The predictors x could also be used to define a prior biased towards even splits of the data if so desired, but we have yet to investigate the desirability of such an option.

For most of our experiments we decided to use what we call a *GROWTREE prior*, which is essentially that used by [Chipman *et al.*, 1998]. Attempting to use an SLP to model an externally provided prior provides a test of the representational flexibility of SLPs: indeed we were unable to pass this test before we extended SLPs to have variable probability labels. Using the GROWTREE prior also allows better comparison with others' results.

The GROWTREE prior grows a C&RT tree by starting with a single leaf node and then repeatedly splits each leaf node η with a probability $\alpha(1 + d_\eta)^{-\beta}$, where d_η is the depth of node η and α and β are prior parameters set by the user to control the size of trees. Unsplit nodes become leaves of the tree. If a node is split, then the splitting rule for that split is chosen uniformly. An abbreviated fragment of the SLP which expresses this prior is given in Fig 4.

```
Splt:[Splt]:splt_lf(D, ..., A/B, ...) :- ..
  D1 is D + 1,
  NwSplit is A * exp((1+D1), -B), ...
  [NwSplit]:splt_lf(D1, ..., A/B, ...),
  [NwSplit]:splt_lf( D1, ..., A/B, ...).
(1-Splt):[Splt]:splt_lf(D, ..., AB, Lf) :-
  Lf = leaf(D, ...).
```

Figure 4: GROWTREE prior fragment defined with an SLP

We have also experimented with what we call *EDITTREE* priors. Here an 'initial' C&RT model is supplied. This can be any tree, for example the tree produced by the standard greedy algorithm or one manually entered by the user. This tree is then probabilistically 'edited', by either growing, pruning or changing a splitting rule. The number of such edits is

also probabilistic.

4 Using the Metropolis-Hastings algorithm

The Metropolis-Hastings (MH) algorithm is an MCMC algorithm which defines a Markov chain $(T^{(i)})$ with a transition kernel $K(t^i, t^{i+1})$ as follows. If $T^{(i)} = t^{(i)}$, then $T^{(i+1)}$ is produced by generating $T'_i \sim q(t'|t^{(i)})$ for some proposal distribution q . An acceptance probability $\alpha(t, t')$ is used to decide whether to accept the proposed t' . $t^{(i+1)} = t'$ with probability $\alpha(t, t')$ (the proposed t' is accepted) and $t^{(i+1)} = t^{(i)}$ with probability $1 - \alpha(t, t')$. By setting the acceptance probability like this:

$$\alpha(t, t') = \min \left\{ \frac{P(t'|x, y) q(t|t')}{P(t|x, y) q(t'|t)}, 1 \right\} \quad (1)$$

the chain $(T^{(i)})$ will converge to sampling from the desired posterior $P(T|x, y)$ under weak conditions (whatever the starting point $t^{(0)}$ of the chain). It is not difficult to see (via Bayes theorem) that if $q(t|t')P(t'|x) = R_q(t, t')q(t'|t)P(t|x)$, for some function R_q , then:

$$\alpha(t, t') = \min \left\{ R_q(t, t') \frac{P(y|t', x)}{P(y|t, x)}, 1 \right\} \quad (2)$$

In our approach we choose proposal distributions q which permit this simplification of the acceptance probability, and where the value $R_q(t, t')$ is easily computable for each pair of trees t, t' . The crucial point is this: such a choice of q ensures that $\alpha(t, t')$ is easily computable even though computing the prior probability for any given C&RT model (as opposed to merely sampling from the prior) may be prohibitively expensive. This contrasts with [Chipman *et al.*, 1998] where "specifications [which] allow for straightforward evaluation of $p(t)$ for any t " are required. For us the computation of prior probabilities can be expensive due to backtracking. Without backtracking SLPs are similar to stochastic-context free grammars (SCFGs)—particularly when probability labels are fixed, and not computed 'on the fly'—and SCFG dynamic programming algorithms could presumably be used to compute prior probabilities. Ruling out backtracking greatly restricts the constraints the user can declare so we always allow for backtracking.

Since our trees do not explicitly define a distribution over classes at the leaves of the tree, our likelihoods are marginal likelihoods—we integrate over all possible class distributions. This integration automatically prevents over-fitting: trees with 100% training set accuracy do not generally have high marginal likelihood. To actually do the integration, we use exactly the same approach as [Chipman *et al.*, 1998; Denison *et al.*, 1998; 2002]. For each leaf, the uniform Dirichlet distribution (all parameters set to 1) is the distribution over all possible distributions over classes. The Dirichlet assumption allows a closed form for the marginal likelihood $P(y|t, x)$.

Our proposals can best be understood by first recalling that sampling C&RT trees from an SLP prior is effected by sampling proofs from the prior as explained in Section 2, and that each such proof has an associated proof tree, such as shown

in Fig 3. Each proposal we have considered works by pruning the proof tree corresponding to the current C&RT model at some point, and then re-growing the proof tree from the pruning point using the sampling mechanism associated with the prior. Our proposals only differ in how they choose where to ‘snip’ the proof tree. Since only the sub-tree below the prune point is resampled, our proposals are *not* equivalent to proposing with the prior $P(t|x)$ (except for one special case to be discussed shortly). Instead each prune point k corresponds to proposing a tree according to $P(t|x, t \in T_k)$ where T_k is the set of trees which may differ from the current tree only in the sub-tree under prune point k .

One (overly) simple proposal corresponds to the *independent Metropolis-Hastings* algorithm which prunes away the current model completely so that a new model is sampled from the prior independently of the current model. For this proposal (denoted q_0) we have the following MCMC convergence result. Let μ_i be the distribution over trees produced at the i th iteration of this independent MH algorithm. Let $\hat{t}$ be a C&RT tree with maximal marginal likelihood: then

$$|\mu_i - P(\cdot|x, y)| \leq [1 - P(\hat{t}|x)/P(\hat{t}|x, y)]^i \quad (3)$$

where $|\cdot|$ denotes the total variation distance between two distributions. So there is exponentially fast convergence to the true posterior $P(\cdot|x, y)$ using this independent MH algorithm. The catch is that, $P(\hat{t}|x)/P(\hat{t}|x, y)$, the ratio between $\hat{t}$ ’s prior and posterior probability, is tiny for any reasonably sized dataset, so in experiments (not presented here) the independent M-H algorithm performs poorly. The convergence result can be derived from one by [Doob, 1953] which is given by [Rosenthal, 1995]. To apply Doob’s result it is necessary to produce an upper bound on $\sum_{t'} \min_t K_{q_0}(t, t')$, where K_{q_0} is the transition kernel associated with q_0 . In fact, we can show (proof omitted) that this sum is exactly $P(\hat{t}|x)/P(\hat{t}|x, y)$, thus establishing (3). This upper bound for our discrete finite space is exactly half the size of the bound for the general independent MH algorithm established by [Mengersen and Tweedie, 1996].

A somewhat better performing proposal (denoted q_{0-n}) cycles through proposals q_k for $k = 0, 1, 2, \dots, n$ for a user-supplied n , where q_k is the proposal which prunes the proof tree at the k th choice point. We have $R_{q_{0-n}}(t, t') \equiv 1$, so that the acceptance probability is simply a likelihood ratio. However, our best performing proposal (denoted q_{uc}) makes a uniform choice of a prune point from all those available in the current tree. For q_{uc} , $R_{q_{uc}}(t, t') = d_t/d_{t'}$, where d_t is the depth of tree t .

One recent innovation has greatly increased the efficiency with which the space is explored. In all previous work using SLPs for MCMC, the prune points to which our proposals jumped were not organised in a proof *tree*, but in a *sequence* (a branch of the relevant SLD-tree, in fact). Essentially, this was because our previous proposal mechanism was closely modelled on Prolog backtracking: if we backtracked to choice point k all C&RT building work done after that choice point was thrown away. Now, if we backtrack to point k then *only the proof tree below k* is (stochastically) rebuilt, *even if other parts of the proof tree were built chronologically later*. Translating this specifically to C&RT mod-

els, this means our proposals can now snip the current C&RT tree at any node, and re-grow the tree from there, leaving the rest of the C&RT tree unscathed. This makes it easier for the Markov chain to head in the direction of high likelihood trees: sub-trees with high-likelihood tend to be altered rarely whereas proposed replacements for low-likelihood sub-trees are accepted relatively often.

5 Experimental results

In our experiments we have used three datasets: the Wisconsin breast cancer data (BCW), the Kyphosis dataset (K) and the Pima dataset (P). BCW was originally donated to the UCI depository by [Wolberg and Mangasarian, 1990] and was used by [Chipman *et al.*, 1998]. Dataset K comes as part of the `rpart` R package for building and manipulating C&RTs. Dataset P is a UCI dataset which [Denison *et al.*, 2002] used for extensive Bayesian C&RT analysis. Following [Chipman *et al.*, 1998] we have simply deleted 16 datapoints from BCW which contain missing values. In all cases the machine learning task is binary classification using integer-valued predictors—9 predictors in the case of BCW, 3 for K and 8 for P. All splits are binary: made by splitting on some threshold. There are 683 datapoints for BCW, 81 for K and 768 for P.

We have performed a large number of experiments, and analysed the results in many ways. Only a small representative sample of this is reported here. Data and software for reproducing all of our experiments can be found at <http://www.cs.york.ac.uk/~nicos/sware/slps/mcmcs/>. We used Sicstus Prolog 3.11 running under Linux on two machines each with a 2.4GHz processor and at least 512Mb RAM. The computation of the log-likelihood ratio uses a C function called from Prolog. Running a chain for 50,000 iterations takes at most 30 minutes.

5.1 Comparison to the standard C&RT algorithm

The contrasts between Bayesian C&RT and the standard greedy algorithm for building a single C&RT model have been well explored in the existing Bayesian C&RT literature [Chipman *et al.*, 1998; Denison *et al.*, 1998; 2002]. The Bayesian method represents posterior uncertainty better and makes a more thorough exploration of the model space. This is at the cost of more computation. How well the Bayesian approach does in terms of predictive accuracy depends, of course, on the prior. When we do have useful prior knowledge it can help us if it can be incorporated into the prior distribution. Facilitating this is our central motivation.

A tree with maximal marginal likelihood is a maximum *a posteriori* (MAP) model when a uniform prior is assumed. Call such trees MAP_{unif} trees. Any MCMC run produces an approximation to a MAP_{unif} tree: it is the maximum marginal likelihood tree in the MCMC sample. Since the standard greedy algorithm is closely connected to a search for a MAP_{unif} tree [Buntine, 1990] it is interesting to compare the marginal likelihood of trees produced using the greedy algorithm to that of the MAP_{unif} tree approximations found in the the MCMC sample.

The trees found by `rpart` for datasets BCW and K using default settings have (rounded) marginal log-likelihoods -97

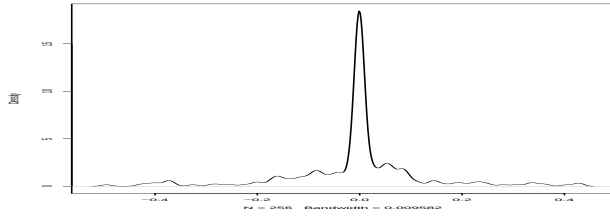


Figure 5: Distribution of differences in estimated class posterior probabilities from two MCMC runs only differing in random seed.

and -39, and sizes 7 and 5, respectively. Unsurprisingly, there are trees produced by MCMC runs with higher marginal log-likelihoods (i.e. with values closer to that of a MAP_{unif} tree). For BCW (resp. K) we can find a tree with 5 (resp. 3) leaves and log-likelihood of -86 (resp. -36).

5.2 Reproducibility of results

Since the aim of using MCMC is to approximate the posterior distribution, it is important that inferences drawn from any reasonably long realisation of the Markov chain are robust to changing the random seeds which determine the evolution of the chain. Using our original sequence-based proposals, there was strong evidence that this was *not* the case: for example, plots of log-likelihood against iteration number (*log-likelihood trajectories*) could be quite different for experiments which were identical except for the random seeds used.

To test our new proof-tree based proposals we performed the following experiment (amongst many others not reported here). Two MCMC runs of 50,000 iterations were performed using the GROWTREE prior ($\alpha = 0.95, \beta = 0.8$) using only 2/3 of the Pima dataset, differing only in the random seeds used. For each of the remaining 256 Pima dataset examples each of the two MCMC samples were used to approximate the posterior probability that the example had class label 0. This was done by simply getting each tree in the MCMC sample to make a class prediction for the example based on the majority class at the appropriate leaf and setting the class posterior probability equal to the relative frequency of class 0 predictions. Clearly, if both MCMC samples were perfect representations of the true posterior, then the difference in estimates of the class 0 posterior would be zero for each example. Naturally, this is not achieved but the distribution of the differences in probability estimates (shown in Fig 5) is highly concentrated about zero which provides evidence that our approach often produces values close to the true posterior, for different runs.

5.3 Using local jumps

Local jumping in tree space helps prevent getting stuck. The log-likelihood trajectory using q_{0-7} shown in Fig 6, shows that the big jumps which q_{0-7} proposes are rarely accepted, so the chain remains stuck at the same model for long periods. This problem is even more pronounced if q_0 , the independent MH algorithm, is used. Compare this with other figures where the q_{uc} is being used.

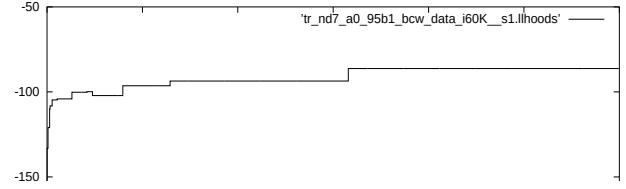


Figure 6: Log-likelihood trajectory. Prior=GROWTREE ($\beta = 1$), Data=BCW, MCMC= q_{0-7} .

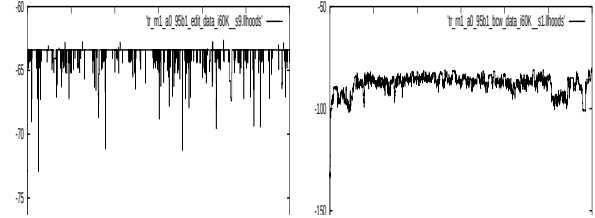


Figure 7: Log-likelihood trajectories. In both cases: Data=BCW, MCMC= q_{uc} , sequential. For LHS: Prior=EDITTREE. For RHS: Prior=GROWTREE ($\beta = 1$)

5.4 Influence of priors

Fig 7 compares log-likelihood trajectories using the EDITTREE prior and the GROWTREE prior respectively with all other parameters being equal. The distinct horizontal line in the EDITTREE trajectory is clear evidence that the EDITTREE prior is pulling the Markov chain back to the initial tree. Note that Figs 6-7 all contain initial very low log-likelihoods (corresponding to the chain's initial C&RT model) which have been truncated for space reasons.

[Denison *et al.*, 2002] examined how successful their chains were at finding C&RT trees with high marginal likelihood using the Pima (P) dataset. Their maximum marginal log-likelihood was -343; our highest was -347. To test a hypothesis that many high likelihood C&RT models did not have high posterior probability for our priors (and hence were unlikely to be visited), we started chains from the log-likelihood=-343 C&RT model of Denison *et al.*. Fig 8 shows the results where the RHS trajectory corresponds to a prior with a stronger bias towards smaller trees. In both cases the prior pulls away from high-likelihood C&RT models, all the more rapidly for the stronger prior.

Since the goal of including prior knowledge is to improve decision making under uncertainty (e.g. classifying test examples), we performed experiments where a known ‘true’ tree was used to generate synthetic train and test data. We then included boxes (see Section 2) consistent with the true tree as constraints on a GROWTREE prior, produced MCMC samples using the synthetic training data, and measured predictive accuracy on the synthetic test data. Each of the 500 test examples was classified into its most probable class as estimated by the MCMC sample. This was all repeated for 3 random seeds. With no box constraints test-set accuracies were 74.8%, 76.2% and 74.4%. With the box constraints given in Fig 1, the test-set accuracy was exactly 78.6% for each random seed. These results indicate that accurate prior

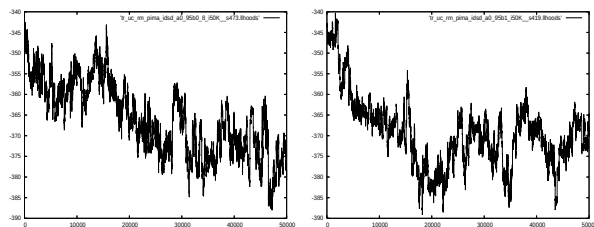


Figure 8: Log-likelihood trajectories. In both cases: Data=P, MCMC= q_{uc} , proof tree. Prior=GROWTREE. Initial tree = Denison’s maximal likelihood. For LHS: $\beta = 0.8$. For RHS: $\beta = 1$

knowledge (i) helps convergence and (ii) increases predictive accuracy.

6 Comparison with related work

This paper lies at the intersection of two lines of work: that on Bayesian C&RT [Chipman *et al.*, 1998; Denison *et al.*, 1998; 2002] and that on using SLPs for general Bayesian model inference [Angelopoulos and Cussens, 2001]. The SLP work had claimed to provide a general Bayesian machine learning method but had only results for model spaces composed of Bayesian nets. This paper backs up this initial claim by applying the basic framework of [Angelopoulos and Cussens, 2001] to C&RT model space. The SLP method has also been improved in a number of ways: (i) probabilities no longer need be hard-coded constants, (ii) the use of *backtrackable sampling* adds an element of search to the prior and (iii) the proposal mechanism now works on a (proof) tree structure which makes it easier to move through the model space.

The most obvious difference from other Bayesian C&RT work is that here all (non-parameter) priors are defined by an SLP—however as we have seen SLPs can encode (at least some) priors originally expressed otherwise. Secondly, in our approach there is only one way of proposing new C&RT models: by pruning and re-growing the proof tree. In the other Bayesian C&RT work a variety of moves are used—five are listed in [Denison *et al.*, 2002, p. 161]. The compensation for our severely restricted proposals is that no prior terms complicate the acceptance probability. Constraining the proposal mechanism to produce such an acceptance probability is also a choice taken in [Denison *et al.*, 2002]—although for this to work the proposal mechanism has to depend upon global parameters of the current tree. The biggest contrast with previous Bayesian C&RT work is that we permit big jumps by pruning the proof tree near the root. In [Denison *et al.*, 2002] it is noted that pruning off any given branch of a tree is straightforward in their approach but that how to generate a similar branch (to maintain reversibility) “is not obvious”. In the approach presented here it is obvious, we can just re-grow the proof tree *because the proposal is based on the prior*. In [Denison *et al.*, 2002] the danger that many big jumps will propose trees with sparsely populated leaves is also noted—we avoid this problem by simply compelling the proposal mechanism to search for trees (via backtracking) with adequately populated leaves.

7 Future work

The convergence result (3) for the independent MH algorithm is something we hope to generalise to other proposals. This will guide the choice of proposal mechanism in contrast to our current empirical approach. In ongoing work [Angelopoulos and Cussens, 2005] experimental results have been produced which show that *tempering* improves the rate of convergence of q_{uc} considerably. Thirdly, like all other work (that we know of) on Bayesian C&RT, we do not have full-blown convergence diagnostics implemented: a deficiency we hope to remedy.

Our current implementation does not fully exploit Prolog’s built-in backtracking: if this were possible this would permit a significant speed-up. A more radical approach is to adapt an existing Prolog system to have our probabilistic mechanisms built-in.

Acknowledgements

Thanks to 3 anonymous reviewers for useful criticisms. This work was supported by the UK EPSRC MathFIT project *Stochastic Logic Programs for MCMC* (GR/S30993/01).

References

- [Angelopoulos and Cussens, 2001] N. Angelopoulos and J. Cussens. Markov chain Monte Carlo using tree-based priors on model structure. In *Proc. UAI-01*, 2001.
- [Angelopoulos and Cussens, 2005] N. Angelopoulos and J. Cussens. Tempering for Bayesian C&RT. Sub’d, 2005.
- [Buntine, 1990] Wray Buntine. *A Theory of Learning Classification Rules*. PhD thesis, Uni. of Tech., Sydney, 1990.
- [Chipman *et al.*, 1998] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search. *JASA*, 39(443):935–960, 1998.
- [Cussens, 2000] J. Cussens. Stochastic logic programs: Sampling, inference and applications. In *Proc. UAI-00*, 2000.
- [Denison *et al.*, 1998] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. A Bayesian CART algorithm. *Biometrika*, 85(2):363–377, 1998.
- [Denison *et al.*, 2002] D. G. T. Denison, C. C. Holmes, B. K. Mallick, and A. F. M. Smith. *Bayesian Methods for Non-linear Classification and Regression*. Wiley, 2002.
- [Doob, 1953] J. I. Doob. *Stochastic Processes*. Wiley, New York, 1953.
- [Mengersen and Tweedie, 1996] K. L. Mengersen and R. L. Tweedie. Rates of convergence of the Hastings & Metropolis algorithms. *Ann. Stats.*, 24:101–121, 1996.
- [Muggleton, 1996] S. Muggleton. Stochastic logic programs. In *Advances in ILP*. 1996.
- [Rosenthal, 1995] J. S. Rosenthal. Convergence rates for Markov chains. *SIAM Review*, 37(3):387–405, 1995.
- [Wolberg and Mangasarian, 1990] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *PNAS*, 87:9193–9196, 1990.

Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery

Martin Atzmueller and Frank Puppe

Department of Computer Science
University of Würzburg
97074 Würzburg, Germany
{atzmueller, puppe}@informatik.uni-wuerzburg.de

Hans-Peter Buscher

DRK-Kliniken Berlin-Köpenick,
Clinic for Internal Medicine II,
12559 Berlin, Germany
buscher.dhp@t-online.de

Abstract

In general, knowledge-intensive data mining methods exploit background knowledge to improve the quality of their results. Then, in knowledge-rich domains often the interestingness of the mined patterns can be increased significantly.

In this paper we categorize several classes of background knowledge for subgroup discovery, and present how the necessary knowledge elements can be modelled. Furthermore, we show how subgroup discovery methods benefit from the utilization of background knowledge, and discuss its application in an incremental process-model. The context of our work is to identify interesting diagnostic patterns to supplement a medical documentation and consultation system. We provide a case study in the medical domain, using a case base from a real-world application.

1 Introduction

Knowledge-intensive learning methods usually exploit background knowledge, because this can improve the quality of their results significantly (c.f. [Richardson and Domingos, 2003]). In this paper we exploit background knowledge for subgroup discovery [Wrobel, 1997; Klösgen, 2002], applied for exploration or descriptive induction, to discover "interesting" subgroups concerning a certain property of interest. Background knowledge can help to focus the algorithm for subgroup search on the relevant patterns according to the goals of the user, thus reducing uninteresting patterns and restricting the search space. This can increase the quality of the discovered set of subgroups significantly, as well as the efficiency of the search method. It is obvious, that the amount of available background knowledge depends on the particular application domain. For example, in the medical domain usually a lot of background knowledge is available.

Besides (simple) constraints the background knowledge for the presented knowledge-intensive process consists of two categories: a task-specific subset of derived attributes and general ontological background knowledge, that can be refined incrementally. Certain knowledge classes can also be used to derive new (lower-level) knowledge. The knowledge

classes can potentially be used in other rule-based learning methods in a straightforward manner. Similar to the applied knowledge, subgroup discovery results can also be formalized incrementally and can be provided as input to the search method. We will introduce this approach in this paper.

Our implementation and evaluation is based on the knowledge-based documentation and consultation system for sonography SONOCONSULT [Huetting *et al.*, 2004], which is in routine use in the DRK-hospital in Berlin/Köpenick.

The rest of the paper is organized as follows: First we describe the knowledge-intensive process for subgroup discovery. We introduce the subgroup discovery method, present the different types of suitable background knowledge and discuss their characteristics in detail. After that, an experimental evaluation of the impact of the proposed method is demonstrated by a case study in the medical domain. Finally, we conclude the paper with a discussion of the presented work, and we show promising directions for future work.

2 Knowledge-Intensive Subgroup Discovery

In this section, we first give an overview of the subgroup discovery method and define the basic concepts of our knowledge representation schema. Furthermore, we present the knowledge-intensive subgroup discovery process, and describe the background knowledge in detail.

2.1 Subgroup Discovery

Subgroup discovery [Wrobel, 1997; Klösgen, 2002] is a method to discover "interesting" subgroups of individuals, e.g., "the subgroup of patients who are treated in a *small, understaffed* hospital are significantly more likely to suffer from complications in the future than the patients in the reference population". Subgroups are described by relations between independent (explaining) variables and a dependent (target) variable, rated by a certain interestingness measure. For example, two possible criteria are the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size. The main application areas of subgroup discovery are exploration and descriptive induction, to obtain an overview of the relations between a target variable and usually many explaining variables. Subgroup discovery does not necessarily focus on finding complete relations; instead partial relations, i.e., (small) subgroups with "interesting" characteristics can be sufficient.

Before defining the subgroup discovery task, we first introduce the necessary notions concerning our knowledge representation schema. Let Ω_A the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Furthermore, we assume $\mathcal{V}_A$ to be the (universal) set of attribute values of the form $(a : v)$, where $a \in \Omega_A$ is an attribute and $v \in dom(a)$ is an assignable value.

A subgroup discovery task mainly relies on the following four main properties: the target variable, the subgroup description language, the quality function, and the search strategy. The target variable may be binary, nominal or numeric. Depending on its type, there are different analytic questions, e.g., for a numeric target variable we can search for significant deviations of the mean of the target variable.

A subgroup discovery problem encapsulates the target variable, the search space of independent variables, the general population, and additional constraints.

Definition 1 (Subgroup Discovery Problem). A subgroup discovery problem SP is defined as the tuple

$$SP = (T, A, C, CB),$$

where $T \in \Omega_A \cup \mathcal{V}_A$ is a target variable. $A \subseteq \Omega_A$ is the set of attributes to be included in the subgroup discovery process. CB is the case base representing the general population used for subgroup discovery. C specifies (optional) constraints for the discovery method. We define Ω_{SP} as the set of all possible subgroup discovery problems.

The definition above allows for arbitrary target variables. However, for our analytic questions we will focus on binary target variables, i.e., $T \in \mathcal{V}_A$.

The description language specifies the individuals from the reference population belonging to the subgroup.

Definition 2 (Subgroup Description). A subgroup description $sd = \{e_i\}$ consists of a set of selection expressions (selectors) $e_i = (a_i, V_i)$ which are selections on domains of attributes, i.e., $a_i \in \Omega_A, V_i \subseteq dom(a_i)$. A subgroup description is defined as the conjunction of its contained selection expressions. We define Ω_{sd} as the set of all possible subgroup descriptions.

A quality function measures the interestingness of the subgroup (c.f., [Klösgen, 2002] for examples).

Definition 3 (Quality Function). A quality function

$$q : \Omega_{sd} \times \Omega_{SP} \rightarrow R$$

evaluates a subgroup description $sd \in \Omega_{sd}$ given a subgroup discovery problem $SP \in \Omega_{SP}$. It is used by the search method to rank the discovered subgroups during search.

For binary target variables, examples for quality functions are given by

$$q_{BT} = \frac{(p - p_0) \cdot \sqrt{n}}{\sqrt{p_0 \cdot (1 - p_0)}} \cdot \sqrt{\frac{N}{N - n}}, \quad q_{RG} = \frac{p - p_0}{p_0 \cdot (1 - p_0)},$$

where p is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, $N = |CB|$ is the size of the total population, and n denotes the size of the subgroup. In contrast to

the quality function q_{BT} , the quality function q_{RG} only compares the target shares of the subgroup and the total population measuring the relative gain. Therefore, a suitable support threshold is necessary to discover significant subgroups.

Considering the subgroup search strategy an efficient search is necessary, since the search space is exponential concerning all the possible selectors of a subgroup description. Commonly, a beam search strategy is used because of its efficiency [Klösgen, 2002]. For our search strategy, we use a modified beam search strategy, where an initial subgroup description can be selected as the initial value for the beam. Beam search adds a selector to the k best subgroup descriptions in each iteration. Iteration stops if the quality as evaluated by the quality function q does not improve any further.

2.2 The Knowledge-Intensive Process Model

Often suitable background knowledge can help both to focus the subgroup discovery task on the relevant patterns concerning the given analysis goals, and to restrict the search space. We propose an incremental approach in which background knowledge can be applied initially at the start, but also during the discovery process. The proposed knowledge-intensive process for subgroup discovery is depicted in Figure 1.

We start with a defined population given as a case base CB and existing background knowledge, if available. For the analysis task defined by a subgroup discovery problem the subgroup discovery method generates a set of subgroups. If these are considered as interesting by the user, the results are presented, and the process is finished. Otherwise the subgroups are analyzed, and background knowledge including constraints can be added to the subgroup discovery problem. Additionally, selected subgroups can be used as the basis for further refinement. Then the process continues with a new iteration: all constraints of the current subgroup discovery problem are applied for the search process. In summary, the knowledge-intensive process consists of three main steps:

1. DISCOVER: Discover subgroups – result set $\mathcal{SG}$
2. INSPECT: If $\mathcal{SG}$ is interesting – STOP and present the subgroup discovery results; otherwise
3. REFINE: Analyze the discovered subgroups; adapt the subgroup discovery problem, i.e., extend or modify background knowledge and/or use interesting subgroups as a starting point for step 1. GOTO step 1

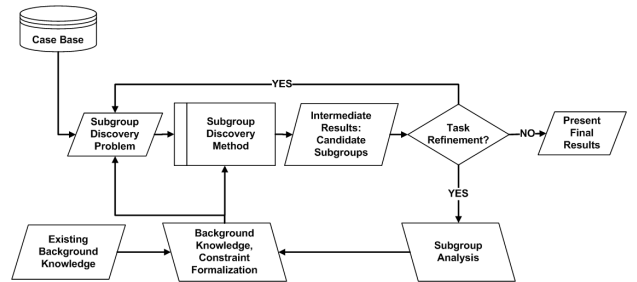


Figure 1: The Knowledge-Intensive Process Model

Step 3 is performed by the domain specialist according to the analysis goals. Examples for the adaptation of the subgroup discovery problem are given in the case study in Section 3.

2.3 Background Knowledge for Subgroup Discovery

We want to make the acquisition of helpful background knowledge as easy as possible. Therefore, we try utilize knowledge known from other knowledge systems, that the domain specialist is already familiar with. There are different classes of background knowledge which can be used in the knowledge-intensive process for subgroup discovery: constraints, ontological knowledge, abstraction knowledge, and pattern knowledge. Knowledge acquisition is always expensive, so its costs should be minimized. Sometimes knowledge can be derived from already formalized knowledge, e.g., we can derive constraints from ontological knowledge, and thus reduce its acquisition costs.

In the following table, we summarize the different classes and types of background knowledge (CK = constraint knowledge, OK = ontological knowledge, PK = pattern knowledge, AK = abstraction knowledge). We show their characteristics in terms of the 'derivable knowledge' if applicable, their costs, and their potential contribution to restricting the search space and/or focusing the search process. Considering the costs and the impact of the knowledge types on the search space, the label - indicates no cost/impact; the labels +, ++, and +++ indicate increasing costs and impact. A +(+) signifies, that the respective element has low costs if it can be derived/learned, and moderate costs otherwise. Similarly ++(+) indicates this for moderate and high costs, respectively.

Knowledge		Derivable Knowledge	Cost	Search Space	
Class	Type			Restr.	Focus
CK	Syntactical Constr.	-	+	+	+
CK	Quality Constr.	-	+	++	++
CK	Attr. Values Constr.	-	+(+)	+	+
CK	Meta Values Constr.	-	+(+)	-	++
CK	Attributes Constr.	-	+(+)	++	++
OK	Normality Info	Attr. Val. Constr.	+	+	++
OK	Abnormality Info	Attr. Val. Constr.	++	+	++
		Meta Val. Constr.	-	-	++
OK	Similarity Info	Meta Val. Constr.	++(+)	-	++
OK	Ordinality Info	Meta Val. Constr.	+	++	+++
OK	Attr. Weights	Attr. Constraints	+(+)	+	++
AK	Derived Attributes	Derived Attributes	+++	+++	+++
PK	Subgroup Pattern	Derived Attributes	+(+)	-	+
PK	Priority Groups	-	++	-	+

The most important types of background knowledge with an especially good cost/benefit ratio are indicated in bold type. Derived attributes are a special case with potentially high benefit as discussed below. Then, the need for derived attributes depends on the specific application domain.

Constraint Knowledge Constraints are a class of background knowledge, which is especially simple to apply. The different constraint types can be described as follows:

- Attribute values constraints: the attribute values can be restricted to the set of relevant values.
- Meta values constraints: specific value groups defining an abstracted 'meta value' can be specified, e.g., intervals for ordinal values. Meta values are not restricted to intervals, but can cover any combination of values.
- Constraints for attributes: specific attributes and/or combinations of attributes can be excluded.
- General constraints restricting the syntactical form of the discovered subgroups, or quality constraints for the discovered subgroups can be applied.

Constraints can significantly restrict the search space and focus the search process. The knowledge acquisition cost for constraints is moderate, depending on the number of relations that need to be modeled; the costs can also be decreased utilizing ontological knowledge as described below.

Pattern Knowledge In general, pattern knowledge specifies the kinds of patterns that the user is especially interested in. For example, pattern knowledge can be used to define already known subgroups, that can be directly applied in the discovery process for subgroup refinement. As another kind of pattern knowledge, the user can specify *priority groups* which are partial disjunctive sets of attributes, that have an assigned priority, e.g., depending on the association strength between an attribute and the target variable. Then, we can start with the set of attributes with the highest priority. If the discovered subgroups cannot be improved any further by the attributes in the current set, then we take the attributes in the next prioritized attribute set into account, in the next iteration. The costs for pattern knowledge are not too high, if the patterns are discovered automatically. For priority knowledge, the cost is encoded in the partially disjunctive separation of attributes. Pattern knowledge does not really restrict the search space, but can help to focus the search.

Ontological Knowledge We can utilize ontological knowledge which is commonly used in the development of knowledge systems, e.g., in case-based reasoning systems. The following elements need to be defined by the domain specialist if they cannot be learned (semi-)automatically, e.g., [Baumeister *et al.*, 2002].

- weights of attributes denoting their importance
- similarity information about the relative similarity between attribute values
- abnormality information about attribute values
- ordinality information about attributes

Abnormality/Normality Information If abnormality or normality information about attribute values is available, then each value $v \in \text{dom}(a)$ of an attribute a is attached with a label that explains, if v is describing a normal or an abnormal state of the attribute. Normality information only requires a binary label while abnormality information defines several categories. For example, consider the attribute temperature with the value range $\text{dom}(\text{temperature}) = \{\text{normal}, \text{marginal}, \text{high}, \text{very high}\}$. The values *normal* and *marginal* denote normal states of the attribute, while the values *high* and *very high* describe abnormal states. Several categories can be defined according to the degree of abnormality. We use five degrees of abnormality given by the symbolic values A_1, A_2, A_3, A_4, A_5 . Category A_1 denotes a normal value; the categories $\{A_2, A_3, A_4, A_5\}$ denote abnormal values in ascending order. Using abnormality/normality knowledge we can constrain the value range of an attribute: e.g., either all values corresponding to selected *abnormal* categories or the values marked with the *normal* category can be excluded from the domain of an attribute used in the subgroup discovery method, by a global exclusion condition. Then, attribute values constraints are derived accordingly.

Meta Values Abnormality information and similarity information concerning attribute values can be used to define additional *meta values*: e.g., if the similarity between two attribute values is very high, then they can potentially be analyzed as one value, thus forming a disjunctive selection expression on the value range of an attribute. Likewise, global abnormality groups can be defined by providing groups of abnormality degrees that specify which values should be combined. This is especially relevant in the medical domain with attribute values such as *probable*, *possible*, and *unverifiable*. In diagnostics the value *probable* contributes more evidence to the represented concept than the value *possible*. However, for analysis often the values *probable* and *possible* have an almost equivalent meaning and can be considered as one value.

Ordinality information that specifies if an attribute is ordinal can be used to define meta values relating to ordinal groups: we can generate meta values covering all adjacent combinations of attribute values, or all ascending/descending combinations of values, starting with the minimum or maximum, respectively. If abnormality information is available, then we partition the value range by the given *normal* value and only start with the most extreme value. For example, for the ordinal attribute *liver size* with the values 1:smaller than normal, 2:normal, 3:marginally increased, 4:slightly increased, 5:moderately increased and 6:highly increased, we partition by the *normal* value (2) and obtain the following meta values: (1), (3, 4, 5, 6), (4, 5, 6), (5, 6) and (6).

We summarize how constraints can be derived from ontological knowledge: The set of relevant attributes can be constrained using attribute weights, with a suitable threshold. Given similarity and abnormality/normality information about attribute values we can model/restrict the value ranges of attributes. Ordinality information about attribute values can be easily used to construct ordinal grouped meta values, which are often more meaningful for the domain specialist. Then, the original values can be replaced, optionally.

Abstraction Knowledge This class of background knowledge is given by derived attributes that are inferred from basic attributes or other derived attributes. Derived attributes often correspond to certain known dependencies between attributes, or known intermediate concepts, that are not stored as basic attributes. For example, in the medical domain, we can infer the derived attribute *body mass index*, given the attributes *height* and *weight*. Additionally, if there are a lot of basic attributes in the case base, (known) multi-correlations between basic attributes can cause unstructured subgroup discovery results. Then, data abstraction can increase the interpretability of the subgroup discovery results significantly. Simple concepts can be aggregated to intermediate concepts to form more potentially meaningful and interesting selectors. A nominal derived attribute $a \in \Omega_A$ is defined using abstraction rules, which are utilized to derive the values $v_{i_a} \in \mathcal{V}_A$ concerning attribute a . A rule of the form $r_{v_a} = \text{cond}(r_{v_a}) \rightarrow v_a$ is used for a value v_a of attribute a , where the rule condition $\text{cond}(r_{v_a})$ contains conjunctions and/or disjunctions of (negated) attribute values $v_i \in \mathcal{V}_A$. Furthermore, derived attributes with a numerical value range can be defined by algebraic formula expressions.

Improving the Handling of Missing Values Considering the quality functions abstraction knowledge contributes to one major point – handling *missing values*. Missing values in cases are a significant problem for subgroup discovery and machine learning in general. For example, in medical case bases for a specific patient usually only a subset of the possible examinations is performed, given a structured data gathering strategy in real-world applications. Then, only the relevant questions for the diagnostic tasks are presented to the user. This results in reduced costs for the examiner, however then a specific instance of the data set concerning the basic attributes may be quite sparse.

There exist several strategies for dealing with missing values: a common strategy [Tsumoto, 2002] removes objects (cases) with missing values from the set of analyzed objects. Other strategies try to fill in the missing values according to statistical evaluations, or try to model the distribution [Ragel and Crémilleux, 1999]. The subgroup quality functions basically perform a kind of statistical hypothesis testing given a subgroup description, the target variable and the general population. For such a test only the cases of the population can be considered in which all variables have defined values. The power of the test is decreased significantly, if many analysis objects are removed due to missing values.

In general, we cannot simply apply the “closed-world assumption”, i.e., that missing values of a concept indicate the non-existence or negation of the concept. For example, in the medical domain, a diagnosis may be missing, because either all its relevant observations are missing or they are known but denote the normal, i.e., the non-pathological state. Consequently the diagnosis is not inferred. If we construct a derived attribute to indicate the cases when the relevant observations are missing, then we can use this derived “helper” attribute to indicate when the diagnosis really has a missing value. Additionally, derived attributes besides the described “helper” attributes can also be constructed accordingly to minimize missing values themselves, such that a certain default value is provided which denotes the normal category. So, the derived attributes serve three purposes:

- They focus the subgroup discovery method on the relevant analysis objects,
- They decrease multi-correlations between attributes that are not interesting,
- Derived attributes can reduce missing values for a given concept, since they can be constructed such that a defined value is more often computed if the respective concept would have a missing value otherwise.

The derived attributes can either be constructed based on expert knowledge, or on specific discovered subgroups. A subgroup description is a set of selectors for a specific target concept that are highly correlated with the concept. If the selectors can be abstracted into a derived attribute, then it can be used as potential background knowledge as well. Furthermore, derived attributes can potentially be refined according to the subgroup discovery results, by specialization or generalization. Abstraction knowledge is probably the most costly class of background knowledge in the knowledge-intensive process. If the abstractions are not based on discovery results, then they have to be formalized manually by the expert.

2.4 Related Work

Using background knowledge to constrain the search space and pruning hypotheses during the search process has been proposed in ILP approaches. [Weber, 2000] proposes *require*- and *exclude*-constraints for attribute – value pairs, in order to prune the search space. [Zelezny *et al.*, 2003] integrate constraints into an ILP approach as well; the used constraints are mainly concerned with syntactical restrictions and constraints relating to the quality of the discovered subgroups.

The main difference between the presented approach and the existing approaches is the fact, that we are able to integrate several new types of additional background knowledge. This additional background knowledge can be refined incrementally according to the requirements of the discovery task, and can additionally be used to infer new background knowledge on the fly, e.g., constraints. As a major point we apply special abstraction knowledge, which can be defined by the expert, or can be constructed semi-automatically using the subgroup discovery results. This type of knowledge can be applied dynamically in the process and does not rely on a static data-preprocessing and cleaning task, for example.

3 Case Study

In this section we describe a case study for the application of the knowledge-intensive process. We use cases taken from the SONOCONSULT system [Huettig *et al.*, 2004] – a medical documentation and consultation system for sonography – which has been developed with the knowledge system D3 [Puppe, 1998]. The system is in routine use in the DRK-hospital in Berlin/Köpenick and documents an average of about 300 cases per month. These are detailed descriptions of findings of the examination(s), together with the inferred diagnoses (binary attributes). The derived diagnoses are usually correct as shown in a medical evaluation (c.f. [Huettig *et al.*, 2004]), resulting in a high-quality case base with detailed case descriptions. The applied SONOCONSULT case base contains 4358 cases. The domain ontology contains 427 basic attributes with about 5 symbolic values on average, 133 symptom interpretations, which are rule-based abstractions of the basic attributes, and 221 diagnoses. This indicates the potentially huge search space for subgroup discovery.

Subgroup discovery was performed using the [VIKAMINE, 2005] (*Visual, Interactive and Knowledge-Intensive Analysis and Mining Environment*) system, developed at the Department of Computer Science VI of the University of Würzburg. We used beam search with a beam size of 10 as the search strategy, and the quality function q_{RG} as defined in Section 2.1. The discovered subgroups were evaluated by domain specialists according to (clinical) novelty, interestingness, and actionability aspects.

First, we performed subgroup discovery only using basic attributes and general background knowledge. We used attribute weights for feature subset selection. The subgroup discovery algorithm presented many significant subgroups, that supported the validity of the subgroup discovery techniques. However the results at this stage were not really novel, since they indicated mostly already known dependencies, e.g., the relation between *relative body weight* and *body-mass index*.

In the next stage, the expert decided to define new attributes, i.e., abstracted attributes which described interesting concepts for analysis. The expert provided 45 derived attributes, that consist of symptom interpretations directly indicating a diagnosis and intermediate concepts which are used in clinical practice, for example *pleura-effusion*, *portal hypertension*, or *pathological gallbladder status*. Furthermore, constraints were formalized, that prevented the combination of certain attributes, to restrict the search space. Some examples are depicted in the Table 1, where the dependent and independent variables are directly related to each other. The

Target Variable	Independent Variable
Chronic Pancreatitis	Pancreas Disease
Pancreas Disease	Carcinoma of the Pancreas
Body Mass Index	Relative Body Weight

Table 1: Examples of known/uninteresting relations

newly defined abstraction knowledge was applied extending the search space to the expert-defined attributes. For each attribute a in the set of derived attributes and each value $v_i \in \text{dom}(a)$, a subgroup discovery problem $SP_{ai} \in \Omega_{SP}$ was generated using the binary target variable ($a = v_i$).

The impact of the added background knowledge was proven by a greater acceptance of the subgroup discovery results by the expert. However still too many subgroups were not interesting for the expert, because too many *normal* values were included in the results, e.g., *liver-size = normal*, or *fatty liver = unlikely*. This motivated the application of abnormality information to constrain the value space to the set of *abnormal* values of the attributes. Additionally, the expert suggested to group sets of values into disjunctive value sets defined by abnormality groups, e.g., grouping the values *possible* and *probable* for some attributes. Furthermore, ordinality information was applied to construct meta values of ordinal attributes like *age* or *liver size*.

Target Variable	Def. Pop.	Subgroup Description
Aorta-sclerosis = calcified	1418	Pancreas Disease={probable;possible}
Aorta-sclerosis = calcified	75	Pancreas Disease={probable;possible} AND Ascites = present

Table 2: Example: Missing Value Problem

Further investigation showed that missing values play a central role in the discovery process. Sometimes the defined population significantly decreased, when adding a selection expression to a subgroup description, compared to the parent subgroup. An example is given in Table 2; the defined population is indicated in the second column. In this example, the attribute *ascites* has many missing values. This problem was solved by adapting the derived attributes to indicate when a missing value corresponds to the value "disease not present". Then, the final set of interesting subgroups was obtained.

Figure 2 shows exemplary discovered subgroups concerning the target variable *gallstones*, that were considered as interesting for clinical practice (lines 1-6). All these subgroups were at least significant at the 10^{-6} level. The individual subgroups are shown in the rows of the table. Subgroup parameters given in the columns are: (Subgroup) Size, TP (true positives), FP (false positives), Pop. (defined population size), RG (relative gain) and the value of the binomial quality function q_{BT} (Bin. QF), c.f., Section 2.1. The domain specialists

evaluated the discovered subgroups concerning interestingness for clinical practice: for them, the relative gain was the primary criterion to rank the subgroups in a first step. Then, as a combined (helper) measure for subgroup size and gain quality, the value of the binomial quality function q_{BT} was used for post-processing the set of subgroups.

Target Variable: Gallstones																			
#	Age	Sex	Liver size						Aorta sclerosis										
	1	2	3	m	f	1	2	3	4	5	6	n	c	Size	TP	FP	Pop.	p0	p
1		X	X						X	X	X	89	37	52	3171	0.172	0.416	1.71	6.17
2		X	X						X	X	X	119	46	73	3171	0.172	0.387	1.5	6.31
3	X	X	X						X	X	X	132	51	81	3171	0.172	0.386	1.5	6.66
4			X			X	X	X	X	X	X	190	68	122	3177	0.172	0.358	1.3	6.99
5			X			X	X	X	X	X	X	207	72	135	3171	0.172	0.348	1.23	6.92
6	X	X	X						X	X	X	64	22	42	3171	0.172	0.344	1.2	3.67
7			X									1651	414	1237	3743	0.177	0.251	0.51	10.57
8										X	X	1334	310	1024	3749	0.177	0.232	0.38	6.66
9				X								1776	408	1368	3749	0.177	0.23	0.37	8.1
10									X	X	X	894	178	716	3177	0.172	0.199	0.19	2.52

Age: 1 = <50 2 = 50-69 3 = >=70

Sex: m = male f = female

Liver size: 1 = smaller than normal 2 = normal 3 = marginally increased 4 = slightly increased 5 = moderately increased 6 = highly increased

Aorta sclerosis: n = not calcified c = calcified

Figure 2: Examples of discovered subgroups (clinically interesting: lines 1-6): e.g., the first line depicts the subgroup (89 cases) described by $Age \geq 70$ AND $Sex=female$ AND $Liver\ size=\{slightly\ or\ moderately\ or\ highly\ increased\}$ AND $Aorta\ sclerosis=calcified$ with a target share (gallstones) of 41.6% (p) compared to 17.2% (p_0) in the general population, with a relative gain of 171% (RG).

Especially interesting for the expert proved the situations, when a specialization of a subgroup significantly improved its quality (#5 vs. #2). Additionally, an important criterion to determine sound subgroups was the comparison of possible value combinations for ordinal and nominal attributes (e.g., #6 vs. #2 or #3 vs. #2).

For clinical practice, the expert preferred small subgroup descriptions if the subgroups were comparable concerning the relative gain measure. This is in line with the heuristic of preferring simpler knowledge for actionability. However, significant improvements in a subgroup specialization countered the increase in the length of the subgroup description, e.g., #7, #9, #10 vs. #1. Furthermore, the baseline qualities of known factors (e.g., age, gender) (c.f., #6 to #10) were also very important for the domain specialist as a reference for subgroup comparison. In summary, the interpretation and judgment of the subgroup discovery results ultimately depends on the expert: even if a suitable quality function is applied, the expert still needs to semantically interpret the subgroup descriptions for the final assessment of a subgroup.

4 Summary and Future Work

In this paper we presented how exploiting background knowledge can help to improve subgroup discovery results in a knowledge-intensive approach. We described several classes of background knowledge in detail, that can potentially be used in other rule-based learning methods besides subgroup discovery as well. A case study using cases from a real-world application showed that applying background knowledge helped to focus the discovery algorithm on the interesting subspace of subgroup hypotheses, increasing the quality of the discovery results. Furthermore, we discussed how applying abstraction knowledge can help to handle the problem of missing values.

In the future we are planning to consider appropriate quality measures concerning the simplicity of the discovered subgroups. Primary work for learning rule bases was presented e.g., in [Atzmueller *et al.*, 2004]. We will especially focus on quality measures which are easy to interpret and tunable to the analysis goals.

References

- [Atzmueller *et al.*, 2004] Martin Atzmueller, Joachim Baumeister, and Frank Puppe. Quality Measures for Semi-Automatic Learning of Simple Diagnostic Rule Bases. In *Proc. 15th Intl. Conference on Applications of Declarative Programming and Knowledge Management (INAP 2004)*, pages 203–213, Potsdam, Germany, 2004.
- [Baumeister *et al.*, 2002] Joachim Baumeister, Martin Atzmueller, and Frank Puppe. Inductive Learning for Case-Based Diagnosis with Multiple Faults. In *Advances in Case-Based Reasoning*, volume 2416 of *LNAI*, pages 28–42, Berlin, 2002. Springer Verlag.
- [Huettig *et al.*, 2004] Matthias Huettig, Georg Buscher, Thomas Menzel, Wolfgang Scheppach, Frank Puppe, and Hans-Peter Buscher. A Diagnostic Expert System for Structured Reports, Quality Assessment, and Training of Residents in Sonography. *Medizinische Klinik*, 99(3):117–122, 2004.
- [Klösgen, 2002] Willi Klösgen. *Handbook of Data Mining and Knowledge Discovery*, chapter 16.3: Subgroup Discovery. Oxford University Press, New York, 2002.
- [Puppe, 1998] Frank Puppe. Knowledge Reuse among Diagnostic Problem-Solving Methods in the Shell-Kit D3. *Intl. Journal of Human-Computer Studies*, 49:627–649, 1998.
- [Ragel and Créwilleux, 1999] Arnaud Ragel and Bruno Créwilleux. MVC—a Preprocessing Method to Deal with Missing Values. *Knowledge-Based Systems*, 12(5–6):285–291, October 1999.
- [Richardson and Domingos, 2003] Matthew Richardson and Pedro Domingos. Learning with Knowledge from Multiple Experts. In *Proc. 20th International Conference on Machine Learning (ICML-2003)*, 2003.
- [Tsumoto, 2002] Shusaku Tsumoto. *Handbook of Data Mining and Knowledge Discovery*, chapter Chapter 43: Medicine. Oxford University Press, New York, 2002.
- [VIKAMINE, 2005] <http://ki.informatik.uni-wuerzburg.de/projects/dm>, 2005.
- [Weber, 2000] Irene Weber. Levelwise search and Pruning Strategies for First-Order Hypothesis Spaces. *Journal of Intelligent Information Systems*, 14:217–239, 2000.
- [Wrobel, 1997] Stefan Wrobel. An Algorithm for Multi-Relational Discovery of Subgroups. In *Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 78–87, Berlin, 1997. Springer Verlag.
- [Zelezny *et al.*, 2003] Filip Zelezny, Nada Lavrac, and Saso Dzeroski. Using Constraints in Relational Subgroup Discovery. In *International Conference on Methodology and Statistics*, pages 78–81. University of Ljubljana, 2003.

Fast and Complete Symbolic Plan Recognition

Dorit Avrahami-Zilberbrand and Gal A. Kaminka

Computer Science Department

Bar Ilan University, Israel

{avraham1,galk}@cs.biu.ac.il

Abstract

Recent applications of plan recognition face several open challenges: (i) matching observations to the plan library is costly, especially with complex multi-featured observations; (ii) computing recognition hypotheses is expensive. We present techniques for addressing these challenges. First, we show a novel application of machine-learning decision-tree to efficiently map multi-featured observations to matching plan steps. Second, we provide efficient lazy-commitment recognition algorithms that avoid enumerating hypotheses with every observation, instead only carrying out bookkeeping incrementally. The algorithms answer queries as to the current state of the agent, as well as its history of selected states. We provide empirical results demonstrating their efficiency and capabilities.

1 Introduction

Plan recognition is the process of inferring another agent's plans, based on observations of its interaction with its environment. Recent applications of plan recognition require observing agents whose behavior is reactive [Rao, 1994], in that the observed agent may interrupt plan execution, switch to a different plan, etc. Such applications include intrusion detection applications [Geib and Harp, 2004], virtual training environments [Tambe and Rosenbloom, 1995], and visual monitoring [Bui, 2003].

In general, plan recognition relies on a *plan library* of plans potentially executed by the observed agent. Typically, the plan library is composed of top-level plans that are hierarchically decomposed. The recognizer matches observations to specific plan steps in the library. It may then infer answers to plan recognition queries, such as the currently selected top-level plans (*current state* query), or the ordered sequence of (completed or interrupted) selected plans (*state history* query).

Existing work leaves several challenges open (see Section 2 for details). First, most existing work assumes observations to be of atomic instantaneous actions. However, observations in recent applications are often complex multi-featured tuples, involving symbolic, discrete, and continuous components (e.g., multiple actuators of the agent). Moreover, obser-

vations may be of continuous actions, maintained over time. For instance, an observation of a soccer player may include its team name, uniform number, current position, velocity, etc. The computational cost of matching such observations against all possible plan-steps is non-trivial, and grows in the complexity of the observation tuple. However, existing investigations ignore this cost. Second, with few exceptions, existing algorithms focus on the current state query (e.g., [Tambe and Rosenbloom, 1995; Bui, 2003]). Some go as far as to ignore all [Tambe and Rosenbloom, 1995] or portions [Wærn and Stenborg, 1995] of the observation history, and are unable to utilize negative evidence (where an expected action was *not* observed).

This paper focuses on a set of efficient algorithms that tackle these challenges: First, we develop a method for automatically generating a matching decision-tree that efficiently matches multi-feature observations, to the plan library. Second, we provide algorithms for efficiently answering the current state and state history queries. These lazy-commitment algorithms avoid computation of hypotheses on every step (as other algorithms do [Geib and Harp, 2004]). Instead, they use linear-time (current state) and polynomial-time (state-history) bookkeeping with every observation, which allows extraction of hypotheses only as needed. Our algorithms are *complete*, in that they admit all recognition hypotheses consistent with the history of observations; they are *symbolic* in that they provide hypotheses with no ranking (probabilistic or otherwise).

We explore the performance of the recognition algorithms with simulated data, and show that they are significantly faster than previous techniques. Their efficiency and completeness make them particularly suited for hybrid plan recognition approaches (e.g., [Geib and Harp, 2004]) in which a symbolic recognizer filters inconsistent hypotheses, passing them to a probabilistic inference engine.

2 Background and Related Work

We focus this brief discussion on closely related work, particularly allowing recognition of interruptible plans. See [Carberry, 2001] for a recent survey of general plan recognition.

Several investigations have utilized multi-featured observations (also of continuous actions), but did not address the efficiency of matching observations to the plan library, in contrast to our work: RESC [Tambe and Rosenbloom, 1995] and RESL [Kaminka and Tambe, 2000] use a hierarchical representation (similar to what we use) to maintain a single hy-

pothesis (RESC) or multiple hypotheses (RESL) as to the current state of an observed agent. Both algorithms ignore observation history in the current state hypotheses, and do not address state history, in contrast to our algorithms.

Geib et al. [Geib and Harp, 2004] developed PHATT, a hybrid recognizer, where a symbolic algorithm filters inconsistent hypotheses before they are considered probabilistically. PHATT assumes instantaneous, atomic actions, and takes a generate-and-test approach: With each observation, the symbolic algorithm generates a *pending set* of possible expected observations, which are matched against the next observation to maintain correct state history hypotheses. The size of the pending set may grow exponentially [Geib, 2004]. In contrast, our work decouples the current state and state history queries, and incrementally maintains hypotheses implicitly, without predicting impending observations. The hypotheses are thus computed only when needed (when hopefully, many of them have been ruled out).

Other investigations assume atomic observations, and do not consider the cost of matching. [Wærn and Stenborg, 1995] explores plan-recognition with limited observation history, to facilitate recognition of reactive behavior. [Retz-Schmidt, 1991] develops an approach based on matching sub-graphs of the plan library. However, the approach does not allow for interruptions of plans.

Our work differs significantly from probabilistic approaches, though it complements them in principle (as demonstrated by Geib et al.). [Bui, 2003] explores probabilistic reactive recognition extending hidden Markov models, focusing on current state query. [Pynadath and Wellman, 2000] explores a probabilistic grammar representation for efficient plan-recognition. None of these considers the cost of matching observations, nor separation of state history from current state queries.

3 Matching Observations to the Plan Library

As commonly done in plan recognition work (e.g., [Bui, 2003]), we utilize a hierarchical representation of the plan library (Section 3.1). We present a method for efficiently matching multi-feature observations to the library (3.2).

3.1 The Plan Library

We represent the plan library as a single-root directed acyclic connected graph, where vertices denote *plan steps*, and edges can be of two types: vertical edges decompose plan steps into sub-steps, and sequential edges specify the expected temporal order of execution. For the discussion, we refer to the children of the root node as *top-level plans*, and to all other nodes simply as *plans*. However, we use the term plan here in its general sense, inclusive of reaction plans, behaviors, and recipes. Indeed, to represent behavior-based agents (where typically behaviors execute over a number of time-steps), plans may have a sequential self-cycle, to allow them to be re-selected. However, no cycles are allowed hierarchically.

Each plan has an associated set of conditions on observable features of the agent and its actions. When these conditions hold, the observations are said to match the plan. For example, a kick-ball-to-goal plan (of a robotic soccer player) may have the following features: The ball must be visible, the distance to the ball is within a given range, and the opponent goal

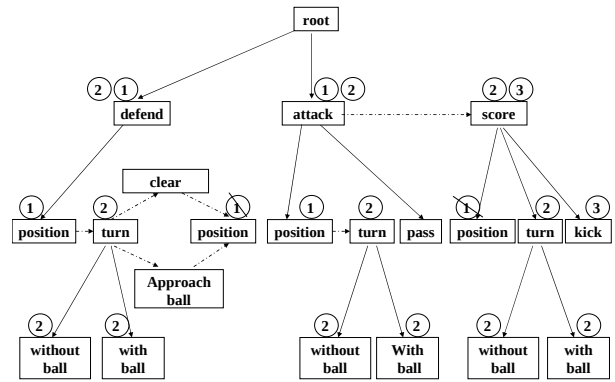


Figure 1: Example plan library. Circled numbers denote time-steps (Section 4).

is visible within shooting distance. If all the above conditions are satisfied, the plan matches the observation.

At any given time, the observed agent is assumed to be executing a *plan decomposition path*, root-to-leaf through decomposition edges. Figure 1 shows an example portion of a plan library, inspired by the behavior hierarchies of RoboCup soccer teams (e.g. [Kaminka and Tambe, 2000]). The figure shows decomposition edges (solid arrows) and sequential edges (dashed arrows). The top level plans are *defend*, *attack*, and *score*. The figure does not show the observation conditions associated with plan steps. For presentation clarity, we show the decomposition edges only to the first (in temporal order) child plans. Thus in the figure, the path *root* → *defend* → *turn* → *with ball* can be an hypothesis as to the current plan of an observed player. In realistic settings, likely more than one path will match an observation tuple, and this may result in a set of such decomposition paths, i.e., a set of hypotheses as to the current state of the observed agent (answering a current state query).

An observed agent is assumed to change its internal state in two ways. First, it may follow a sequential edge to the next plan step. Second, it may reactively interrupt plan execution at any time, and select a new (first) plan. For instance, suppose the agent was executing *root* → *defend* → *turn* → *with ball*, and then interrupted execution of this plan. It may now choose *root* → *attack* → *pass*, but not *root* → *attack* → *turn* → *withball*.

3.2 Efficient Matching

The first phase of recognition, common to all recognition approaches, matches the observations made by the recognizer to plans in the plan library. In contrast with previous work, we consider complex observations, that consist of a tuple of observed features, including states of the world that pertain to the agent (e.g., a soccer player’s uniform number), actions taken (e.g., *kick*), and execution conditions maintained (e.g., *speed* = 200). Matching such observations to plans can be expensive, if we go over all plans and for each plan check all observed features. This, in fact, is what previous work essentially proposes (e.g., [Kaminka and Tambe, 2000]).

To speed this process, we augment the plan library with a *Feature Decision Tree* (FDT), which efficiently maps observations to matching nodes in the plan library. An FDT

is a decision tree, where nodes correspond to features, and branches to conditions on their values. Determining the plans that match a set of observation features is efficiently achieved by traversing the FDT top-down, taking branches that correspond to the observed values of features, until a leaf node is reached. Each leaf node points to the plans that match the conjunctive set of observations along the top-down path. Ideally, each leaf nodes points to only one plan, though this may not be possible due to inherent ambiguity in the plan library.

An FDT can be automatically constructed, similarly to a machine learning (ML) decision tree [Ross, 1992] but with important differences (see below). We map the plan library into a set of training examples, and then use a modified tree construction algorithm to construct the FDT. Each plan step becomes an example, where the observation conditions become attribute values, and the class is the plan step. Features not tested by a plan step are treated as all attribute values. After generating the training set, the construction of the FDT is done similar to that of a decision tree with missing attribute values (for lack of space, see [Ross, 1992] for details).

There are important differences with ML tree construction processes. The goal is to construct an FDT that is specialized to the "training examples". Every plan step example appears exactly once, and no pruning step is taken (as is commonly done in ML decision trees).

The use of the FDT to efficiently match multi-featured observations to plans is a novel application of methods from machine learning to plan recognition. The benefits to plan recognition are significant: The matching time is dictated by the height of the FDT, rather than the size of the library (L). Let F be the number of distinct observable features. In a theoretical worst case, plans test all possible features, and thus the height of the FDT is $O(F)$. In the worst-case, the leaf in the FDT would point to $O(l)$ plans, where l is the maximum number of plans that are ambiguously consistent with a single observation ($l \ll L$). Thus the complexity of matching observations to plans would be at worst $O(F + l)$. This should be contrasted with a $O(FL)$ used in previous work ([Kaminka and Tambe, 2000]). As with any decision tree, there is a one-time cost of constructing the FDT, and storage overhead in using it. However, these costs were not found to be a hindrance in the experiments we conducted (see Section 5).

4 Recognition Algorithms

We now present algorithms for answering the current state query (Section 4.1), and the state history query (4.2).

4.1 Current State Query Algorithms

An important query in reactive plan recognition is with respect to the current plan step selected by the observed agent. In most hierarchical plan-libraries—as in ours—this query translates to determining the decomposition paths (root-to-leaves) that are consistent with the observations, and potentially are being executed by the observed agent. Each such path is a *current-state hypothesis*.

CSQ (Algorithm 1) is an efficient algorithm for answering the current state query. CSQ's inputs include the plan library and pointers to the plan-steps matching the current observation (e.g., as stored in an FDT leaf). It then works in two

Algorithm 1 CSQ(Matching results M , Library g , Time-stamp t)

```

1: for all  $v \in M$  do
2:    $PropagateUp(v, g, t)$ 
3: for all  $v \in M$  do
4:   while  $tagged(v, t) \wedge \neg \exists ChildTagged(t)$  do
5:      $delete\_tag(v, t)$ 
6:      $v \leftarrow parent(v)$ 

```

phases. First, it calls the *PropagateUp* algorithm (Algorithm 2), to tag complete paths in the plan-library that match the current observation, but taking into account previous observation. The set of matching plans M is assumed to be ordered by depth, parents before children (see below). Then (lines 3–6) it goes over the resulting tags to eliminate any that are hierarchically inconsistent, i.e., where a parent is tagged, but none of its children is tagged. CSQ is meant to be called with every new observation. The tags made on the plan-library are used to save information from one run to the next.

The *PropagateUp* algorithm (Algorithm 2) uses time-stamps to tag nodes in the plan library that are consistent with the current and previous observations. To do this, it propagates tags up along decomposition edges. However, the propagation process is not a simple matter of following from child to parent. A plan may match the current observation, yet be *temporally inconsistent*, when a history of observations is considered. For instance, suppose that the first observation matches the *position* plans (Figure 1). The FDT would point the propagation algorithm to the four instances of *position* (marked with a circled 1), under *defend* (twice), *attack*, and *score*. However, two of the instances are temporally inconsistent (crossed out in the figure): The second instance of *position* (under *defend*) cannot be a first observation, since we should have observed either *clear* or *approachball* before it. Similarly, *position* under *score* is inconsistent because its parent had to have followed *attack*, which was not yet observed. This reasoning about hypothesis consistency over time is a key novelty compared to [Tambe and Rosenbloom, 1995; Kaminka and Tambe, 2000].

To disqualify hypotheses that are temporally inconsistent, *PropagateUp* exploits the sequential edges and the time-stamps. It assumes that the calls to it have been made in order of increasing depth (as discussed above). This allows an assumption (line 5) that matching parents are already tagged or do not have any associated observable features (and are thus compatible with all observations). Line 6 checks for temporal consistency. Time stamp t is temporally consistent if one of three cases holds: (a) the node in question was tagged at time $t - 1$ (i.e., it is continuing in a self-cycle); or (b) the node follows a sequential edge from a plan that was successfully tagged at time $t - 1$; or (c) the node is a first child (there is no sequential edge leading into it). A first child may be selected at any time (e.g., if another plan was interrupted). If neither of these cases is applicable, then the node is not part of a temporally-consistent hypothesis (lines 11–12), and its tag should be deleted, along with all tags that it has caused in climbing up the graph. This final deletion of all failing tags takes place in lines 15–17.

Figure 1 shows the process in action (the circled numbers

Algorithm 2 PropagateUp(Node w , Plan Library g , Time-stamp t)

```

1: Tagged  $\leftarrow \emptyset$ 
2: propagateUpSuccess  $\leftarrow true$ 
3:  $v \leftarrow w$ 
4: while  $v \neq root(g) \wedge propagateUpSuccess \wedge \neg tagged(v, t)$ 
   do
5:   if  $tagged(parent(v), t) \vee features(parent(v)) = \emptyset$  then
6:     if  $tagged(v, t) = 1 \vee \exists PreviousSeqEdgeTaggedWith(v, t = 1) \vee NoSeqEdges(v)$  then
7:       tag( $v, t$ )
8:       Tagged  $\leftarrow Tagged \cup \{v\}$ 
9:        $v \leftarrow parent(v)$ 
10:      propagateUpSuccess  $\leftarrow true$ 
11:     else
12:       propagateUpSuccess  $\leftarrow false$ 
13:   else
14:     propagateUpSuccess  $\leftarrow false$ 
15: if  $\neg propagateUpSuccess$  then
16:   for all  $a \in Tagged$  do
17:     delete_tag( $a, t$ )

```

in the figure denote the time-stamps). Assume that after the matching algorithm returns (at time $t = 1$), the Propagate begins with the four *position* instances. It immediately fails to tag the instance that follows *clear* and *approachball*, since these were not tagged at $t = 0$. The *position* instance under *score* is initially tagged, but in propagating the tag up, the parent *score* fails, because it follows *attack*, and *attack* is not tagged $t = 0$. Therefore, all tags $t = 1$ will be removed from *score* and its child *position*. The two remaining instances successfully tag up and down, and result in possible hypotheses $root \rightarrow defend \rightarrow position$ and $root \rightarrow attack \rightarrow position$.

Complexity Analysis. For each plan instance that matches the observations, the propagation traverses the height of the plan library, expected to be $O(\log L)$. Note that previous works (e.g., [Kaminka and Tambe, 2000]) have the same propagation complexity, but do not filter temporal-inconsistency. Also, CSQ utilizes time-stamps on the plan library, rather than external data structures (e.g., [Geib and Harp, 2004]).

4.2 History of States Query Algorithms

The time stamps used by the CSQ algorithm can be used to also answer queries about *sequence* of plan steps taken by the agent from time $t = 0$ until now, given the history of observations. Answering this query, however, is not a trivial collection of all possible current state hypotheses as generated at times $t = 0, \dots, now$, since observations (and lack thereof—*negative evidence*) at time t_k may rule out current-state hypotheses that were consistent at time t_{k-1} .

An example may serve to illustrate. Continuing the example above, suppose the observation at time $t = 2$ matches the *turn* plan (three instances). The tag $t = 2$ propagates successfully and there are six possible current-state hypotheses for time $t = 2$ (we omit the common *root* prefix): $defend \rightarrow turn \rightarrow without\ ball$, $defend \rightarrow turn \rightarrow with\ ball$, $attack \rightarrow turn \rightarrow without\ ball$, $attack \rightarrow turn \rightarrow with\ ball$, $score \rightarrow turn \rightarrow without\ ball$, $score \rightarrow turn \rightarrow with\ ball$.

Suppose we now make observations at time $t = 3$ that match *kick*. The *score* plan is the only plan consistent with $t = 3$, though both *defend* and *attack* are tagged for times $t = \{1, 2\}$. However, after having made the observation at $t = 3$, we can safely rule out the possibility that *defend* was ever selected by the agent, because *score* can only follow *attack*, and the lack of evidence for either *clear* or *approach ball* at time $t = 3$ (which would have made *defend* a possibility at this time) can be used to rule it out. Thus we infer that the sequence of plan paths that was selected by the robot is $attack \rightarrow position$ (at $t = 1$), $attack \rightarrow turn$ at $t = 2$ (though we cannot be sure which one of *turn*'s children was selected), and finally $score \rightarrow kick$. If we had only wanted the current-state hypotheses for time $t = 3$, we would not need to modify hypotheses for time $t = 2$. However, generating the state history hypotheses requires us to do so.

We use an incrementally-maintained structure, the *Hypotheses Graph* (described below), that holds hypotheses according to time stamps. With every time stamp t , we can use the structure to eliminate hypotheses that were tagged at time $t - 1$, that have become invalid. This also allows separation of the current-state hypotheses from the state history hypotheses, something not addressed with previous work (e.g., [Geib and Harp, 2004; Bui, 2003]).

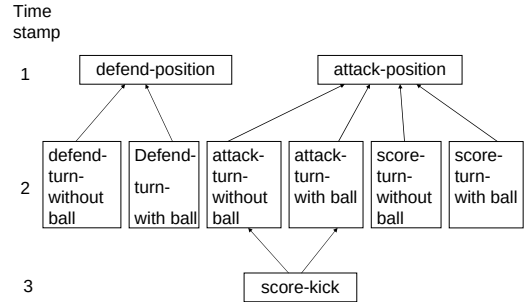


Figure 2: An example extracting graph G' .

The hypotheses graph is a connected graph G' , whose vertices correspond to successfully-tagged paths in the plan library (i.e., hypotheses). Edges in G' connect hypothesis vertices tagged with time stamp t to hypothesis vertices tagged with time stamp $t + 1$. G' is therefore constructed in levels, where each level represents hypotheses that hold at the corresponding time stamp. For each set of observations made at time t_i , we add to G' a level t_i , with nodes for all current state hypotheses that were successfully tagged $t = t_0$. We then create edges between vertices $x_1, \dots, x_n$ in level t_i to vertices $y_1, \dots, y_m$ in level t_{i-1} in the following manner: If x_i is not part of a sequence (i.e., it is a first child), then we connect x_i to each vertex y_j , ($j = 1 \dots m$); otherwise, if x_i is part of a sequence, we connect x_i to y_j if any of the plans in y_j has a sequential edge to a plan in x_i . Finally, if x_i is equal to y_j , we connect them to allow for the self-cycles.

To generate all sequences of plan paths that are consistent with the observations, we traverse G' from the last level t_i backwards, to level t_{i-1} , and on to the first level. Paths that connect level t_i to level $t = 1$ denote valid state histories. To illustrate, Figure 2 shows G' for the example above, $t = 3$.

Complexity Analysis. Let L be the worst-case number of plans that match a single observation. For each node in G' with time stamp t (of which there could be at most $O(L)$), we check all nodes in time stamp $t-1$ (again, $O(L)$), thus a factor of L^2 for each additional level. Thus over N observations, the worst-case runtime is $O(NL^2)$.

5 Experiments

We show results of experiments evaluating these techniques with simulated plan libraries and observation sequences. We controlled key parameters, such as the library size L , and the number of features used by each single plan f ($f \leq 10$), and the structure of temporal edges. In the experiments below, L was set by modifying the number of top-level plans (children of the root node), and the depth of the library. The branching factor was fixed at 3. The maximum length of observation sequence was 10.

We generated valid sequences of observations by simulating execution and selection of plan steps. The process randomly chose a path in the library and used all the features in this path to generate observations. Based on existence of sequential edges, the process chose to either continue execution along a sequential edge, or jump to a new first child. In all experiments, we contrasted the results with RESL [Kaminka and Tambe, 2000], the most relevant of the related works.

5.1 Matching Experiments

A first set of experiments compares matching run-time using FDT and RESL, as the plan library grows in size. We generated different-sized libraries by varying the library depth, and number of top-level plans. To demonstrate the scale-up offered by the FDT as the observation complexity grows, we also vary f (1,3,5,7). For each of these values, we generated 180 random observations sets based on the plan-libraries, and averaged the run-time for matching these using RESL and using an automatically constructed FDT.

The average matching runtime is shown in Figure 3. The figures are arranged in a 3×3 matrix. The rows correspond to the number of top-level plans (5, 50, and 100, respectively, top-down). The columns correspond to library depths of 3, 4 and 5. Thus the bottom right figure shows the results for the largest library, approximately 12,101 nodes. The FDTs for these included 200–2000 nodes, depending on the number of features associated with each plan (f). In all figures, the horizontal (X) axis shows f , and the vertical axis shows the average matching time in seconds. Each point in a figure is the average of 180 runs.

Clearly, the use of the FDT leads to very significant improvements in the matching time, compared to RESL—and even when each plan is associated with a single atomic observation. Furthermore, its growth curve indicates that its benefits are maintained even as the observations increase in complexity.

5.2 Query Answering Algorithms

We now turn to evaluate the algorithms answering the current-state and state-history queries. These answers depend critically on the temporal structure of the plan library, in terms of sequential edges. We follow [Geib and Harp, 2004] in varying temporal structure in several ways (Fig. 4): (a) *Totally*

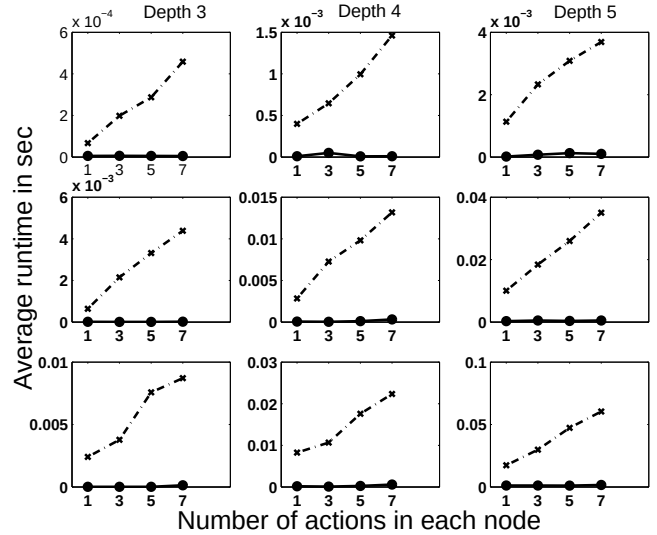


Figure 3: Runtime of FDT (solid line) and RESL (dashed).

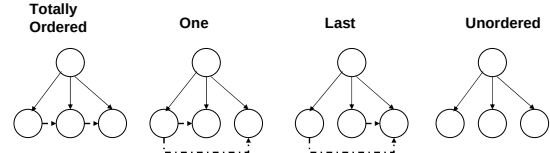


Figure 4: Sequential Links types.

ordered: all children of same parent form a single chain; (b) *First*: first child has sequential edge to all its siblings; (c) *Last*: siblings have sequential edges leading to the last sibling; (d) *Unordered*: no ordering constraints between nodes. Top-level plans are always unordered.

Propagation Accuracy. A key advantage of CSQ over RESL is its ability to use sequential edges and the history of observations to rule out hypotheses that are temporally inconsistent (Section 4.1). Given a sequence of observations, we expect to see fewer current state hypotheses in comparison with RESL.

We again varied the library size through the number of top-level plans (10,50,100) and the depth of the plan library (3–6). We vary the temporal structure of the library as described above. Trials were conducted with sequences of 10–40 observations. We recorded the number of hypotheses maintained after each observation was propagated.

Figure 5 shows the effect of sequential edges types on the number of hypotheses. There are four figures, each for different depth (3–6). The X axis shows the number of top-level plans, while the Y axis measures the average number of hypotheses across all trials of the same configuration. Each data point reflects the average number of hypotheses over 3000 individual observations, organized as 120 random observation sequences (each 10–40 observations in length), based on the generated libraries.

The figures show that *totally-ordered* libraries allow CSQ to maximally use past observations, and thus result in less hypotheses. In contrast, *unordered* libraries have no sequential edges, and thus do not gain information from a history of observations. Thus the number of hypotheses generated in this case is exactly the same as generated by RESL—which

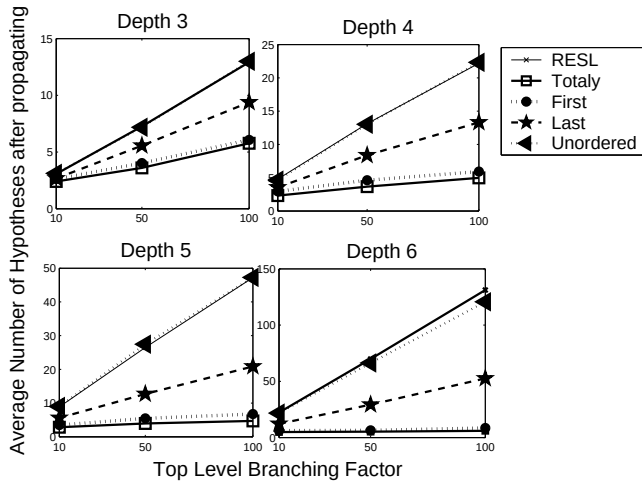


Figure 5: Average number of hypotheses after propagation.

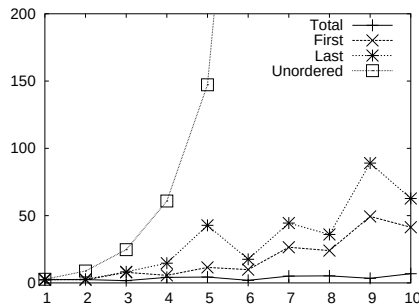


Figure 6: Average number of state history hypotheses with progress in observation.

ignores such history in any case (and thus its results for all library types are the same, shown in a single solid line). On average, more than 50% of the current state hypotheses generated by RESL were ruled out by the CSQ propagation.

We also show the results of evaluating the state history algorithm. Figure 6 shows how incoming observations affect the number of state history hypotheses. The X axis shows the progression of 10 observations with time. The Y axis shows the number of state history hypotheses generated by traversing the hypotheses graph G' . The totally ordered libraries have very few hypotheses, while the number of hypotheses for unordered libraries grows exponentially, since all combinations of current-state hypotheses are valid.

Propagation Runtime. Given the significant improvement in accuracy, one may expect an associated significant computational cost in CSQ, compared to RESL. We have argued analytically that this is not the case, and this is supported empirically. Figure 7 shows the average run-time of CSQ in the above experiments, in comparison to RESL's. The X axis shows the number of top-level plans, while the Y shows runtime in seconds. RESL is only slightly faster than CSQ. Indeed, the difference in propagating between CSQ and RESL amounts to a few additional checks (for incoming edges).

6 Summary and Future Work

This paper presents methods for efficient, complete, symbolic plan recognition that can answer a variety of recognition queries, with increased accuracy. The algorithms depart from previous approaches in that they take a lazy-commitment ap-

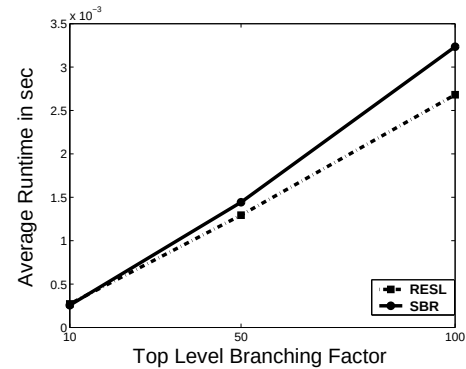


Figure 7: Average runtime of propagating RESL versus CSQ

proach, separating the current state and state history queries, and implicit representation of hypotheses, until required. In addition, we presented a novel application of decision tree construction for efficiently matching observations to the plan library—a step common to all plan recognition methods. We plan to further explore the use of symbolic algorithms in additional queries.

Acknowledgments. This research was supported in part by the Israeli Ministry of Commerce, and by ISF grant #1211/04. Special thanks to Nadav Zilberbrand and K.Ushi.

References

- [Bui, 2003] H. Bui. A general model for online probabilistic plan recognition. In *IJCAI-03*, 2003.
- [Carrbery, 2001] S. Carrbery. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.
- [Geib and Harp, 2004] Christopher W. Geib and Steven A. Harp. Empirical analysis of a probabilistic task tracking algorithm. In *AAMAS workshop on Modeling Other agents from Observations (MOO-04)*, 2004.
- [Geib, 2004] Christopher W. Geib. Assessing the complexity of plan recognition. 2004.
- [Kaminka and Tambe, 2000] Gal A. Kaminka and Milind Tambe. Robust multi-agent teams via socially-attentive monitoring. *JAIR*, 12:105–147, 2000.
- [Pynadath and Wellman, 2000] David V. Pynadath and Michael P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *UAI-2000*, pages 507–514, 2000.
- [Rao, 1994] Anand S. Rao. Means-end plan recognition – towards a theory of reactive recognition. In *Proceedings of the International Conference on Knowledge Representation and Reasoning (KR-94)*, pages 497–508, 1994.
- [Retz-Schmidt, 1991] G. Retz-Schmidt. Recognizing intentions, interactions, and causes of plan failures. *User Modeling and User-Adapted Interaction*, 2:173–202, 1991.
- [Ross, 1992] Quinlan J. Ross. *C4.5 Programs for machine learning*. Morgan Kaufmann Publishers, Inc, 1992.
- [Tambe and Rosenbloom, 1995] M. Tambe and P. S. Rosenbloom. RESC: An approach to agent tracking in a real-time, dynamic environment. In *IJCAI-95*, August 1995.
- [Wærn and Stenborg, 1995] Annika Wærn and Ola Stenborg. Recognizing the plans of a replanning user. In *Proceedings of the IJCAI-95 workshop on plan recognition*, pages 119–123, Montreal, Canada, 1995.

Unsupervised Learning of Semantic Relations between Concepts of a Molecular Biology Ontology *

Massimiliano Ciaramita[†] Aldo Gangemi[†] Esther Ratsch[‡] Jasmin Šarić[§] Isabel Rojas[§]

[†] Institute for Cognitive Science and Technology (CNR), Roma, Italy, *firstname.lastname@istc.cnr.it*

[‡] University of Würzburg, Würzburg, Germany, *esther.ratsch@biozentrum.uni-wuerzburg.de*

[§] EML-Research gGmbH, Heidelberg, Germany, *lastname@eml-r.de*

Abstract

In this paper we present an unsupervised model for learning arbitrary relations between concepts of a molecular biology ontology for the purpose of supporting text mining and manual ontology building. Relations between named-entities are learned from the GENIA corpus by means of several standard natural language processing techniques. An in-depth analysis of the output of the system shows that the model is accurate and has good potentials for text mining and ontology building applications.

1 Introduction

Bioinformatics is one of the most active fields for text mining applications because of the fast rate of growth of digital documents collections such as Medline and SwissProt. The ultimate goal of text mining in bioinformatics is knowledge discovery by means of natural language processing (NLP) and machine learning. To achieve this objective it is necessary to access the information contained “inside” the documents, i.e. their content. One possible strategy to get to the content is through *information extraction*. One starts with a conceptualization of the domain; i.e., a *domain ontology*, which specifies relevant *concepts* as well as *semantic relations*, such as is-a, part-of, and more complex relations encoding important interactions between concepts. Then it is necessary to apply extraction techniques to recognize where, in the documents, concepts are instantiated by specific entities, and where important interactions are expressed by linguistic structures.

Several ontologies which define concepts and structural semantic relations, e.g., is-a, are available. Instead there is a need for ontologies that specify relevant arbitrary semantic relations between concepts; for example, that “Cell express-the-receptor-for Protein” or that “Virus replicate-in Cell”. In this paper we investigate the problem of enriching an existing ontology with arbitrary semantic relations which are strongly associated with ordered pairs of concepts. We design an unsupervised system that combines an array of off-the-shelf NLP techniques such as syntactic parsing, collocation extraction

and selectional restriction learning. We apply our system to a corpus of molecular biology literature, the GENIA corpus [Ohta *et al.*, 2002], and generate a list of labeled binary relations between pairs of GENIA ontology concepts. An in-depth analysis of the learned templates shows that the model, characterized by a very simple architecture, has good potentials for text mining and ontology building applications.

In the next section we describe the problem of learning relations from text and related work. In Section 3 we describe our system and the data used in our study in detail. In Section 4 we discuss the evaluation of the system’s output.

2 Problem statement and related work

The GENIA ontology contains concepts related to gene expression and its regulation, including cell signaling reactions, proteins, DNA, and RNA. Much work in bioinformatics has focused on *named-entity recognition* (NER), or *information extraction* (IE), where the task is the identification of sequences of words that are instances of a set of concepts. As an example one would like to recognize that “NS-Meg cells”, “mRNA” and “EPO receptor” are, respectively, instances of the GENIA classes “Cell_line”, “RNA_family_or_group” and “Protein_molecule” in the following text (Example 1):

- (1) “Untreated [Cell_line NS-Meg cells] expressed [RNA_family_or_group mRNA] for the [Protein_molecule EPO receptor]”

A natural extension of NER is the extraction of relations between entities. NER and relation extraction could provide a better support for mining systems; e.g., patterns of entities and relations could be compared across document collections to discover new informative pieces of information (as in [Swanson and Smalheiser, 1997] for example). Currently most of the work on relation extraction applies hand-built rule-based extraction patterns; e.g., Friedman *et al.* [2001] on identifying molecular pathways and Šarić *et al.* [2004b] on finding information on protein interactions which use a manually-built ontology similar to that described in [Ratsch *et al.*, 2003]. One problem with rule-based IE is that systems tend to have good precision but low recall. Machine learning oriented work has focused on extracting manually-compiled lists of target relations; e.g., Rosario and Hearst [2004] address the relation extraction problem as an extension of NER and use sequence learning methods to recognize

*We would like to thank our colleagues in the Laboratory for Applied Ontology (LOA-CNR), and the Klaus Tschira Foundation for their financial support.

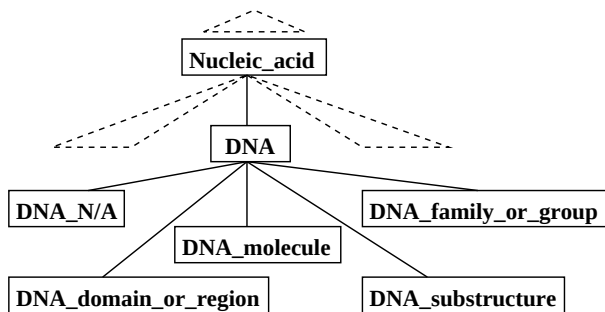


Figure 1: A small fraction of the GENIA ontology. Continuous lines represent unspecified taxonomic relations, dashed lines represent other regions.

instances of a set of 6 predefined relations about “Diseases” and “Treatments”. These systems yield good precision and recall but still need that sets of relations between classes be defined first. Yet another problem which deals with semantic relations is that addressed by Craven and Kumlien [1999] who present a model for finding extraction patterns for 5 binary relations involving proteins. A similar work is that of Pustejovsky et al. [Pustejovsky et al., 2002] on automatically extracting “inhibit” relations. Semantic relations have been used as templates, or guiding principles, for the generation of database schemata [Rojas et al., 2002]. Another application of ontological relations is that of consistency checking of data in molecular biology databases to individuate errors in the knowledge base (e.g., by checking the consistency of the arguments) or to align different databases.

Biological text mining systems that involve relations require predefined sets of relations that have to be manually encoded, a job which is complex, expensive and tedious, and that as such can only guarantee narrow coverage – typically a handful of relations and one pair of classes. Our aim is to automatically generate all relevant relations found in a corpus between all ontological concepts defined in an ontology. This work is also valuable to ontologists since ontology building and evaluation are becoming more and more automatized activities and most of the corpus-based work has focused only on structural relations such as is-a and part-of [Pantel and Ravichandran, 2004; Berland and Charniak, 1999].

3 Learning relations from text

Our model takes as input a corpus of texts in which named-entities, corresponding to ontology concepts, have been identified. Here we use the GENIA corpus, for which the tagging has been carried out manually, but the corpus data can be also generated automatically using an appropriate NER system. The model outputs a set of templates that involve pairs of GENIA ontology classes and a semantic relation. For example, a template might be “Virus infect Cell”.

3.1 Data

The GENIA ontology was built to model cell-signaling reactions in humans with the goal of supporting information extraction systems. It consists of a taxonomy of 46 nominal

concepts with underspecified taxonomic relations, see Figure 1. The ontology was used to semantically annotate biological entities in the GENIA corpus. We used the G3.02 version consisting of 2,000 articles, 18,546 sentences, roughly half a million word tokens, and 36 types of labels. This corpus has complex annotations for disjunctive/conjunctive entities, for cases such as “erythroid, myeloid and lymphoid cell types”. We excluded sentences that contained only instances of complex embedded conjunctions/disjunctions and also excessively long sentences (more than 100 words). The final number of sentences was 18,333 (484,005 word tokens, 91,387 tags). Many tags have nested structures; e.g. “[Other_name [DNA IL-2 gene] expression]”. Here we only considered the innermost labels, although the external labels contain useful information and should eventually be used.

One potential drawback of the GENIA ontology is the relatively small number of biological concepts and their coarse granularity which causes groups of similar but distinct entities to be assigned to the same class. Some relations fit very well to subsets of the entities of the related concepts, whereas they don’t fit well for other entities of the same concept. For example, the concept “DNA_domain_or_region” contains sequences with given start and end positions, as well as promoters, genes, enhancers, and the like. Even if promoters, genes, and enhancers are pieces of sequences too (with start and end positions), they also are functional descriptions of sequences. Therefore, different statements can be made about such kinds of DNA domains or regions and (pure) sequences. The relation “DNA_domain_or_region encodes Protein_molecule” makes sense for genes, but not for enhancers, and may make sense or not for (pure) sequences, depending on their (unknown) function. On the other hand any NLP oriented resource cannot have many fine-grained concepts defined, otherwise IE wouldn’t be accurate. In this respect the GENIA corpus is unique in that it provides extensive named-entity annotations which can be used to train appropriate NER systems (cf. [Kazama et al., 2002]).

3.2 Relations as dependency paths

The 18,333 sentences were parsed with a statistical parser [Charniak, 2000].¹ Since we are interested in relations that connect entities as chunks we want to avoid that the parser analyzes an entity that is split among different phrases. This can happen because entity names can be fairly long, complex and contain words that are unknown to the parser. To avoid this problem we substituted the entity tags for the actual named-entities; the result can be seen in Figure 2 which shows the substitution and the relative parse tree for the sentence of Example 1. Trees obtained in this way are simpler and don’t split entities across phrases. For each tree we generated a *dependency graph*: each word² is associated with one *governor*, defined as the *syntactic head*³ of the phrase closest

¹Which takes roughly three hours on a Pentium 4 machine.

²Morphologically simplified with the “morph” function from the WordNet library, plus morphological simplifications from UMLS.

³The word whose syntactic category determines the syntactic category of the phrase; e.g., a verb for a verb phrase (VP), a noun for a noun phrase (NP), etc.

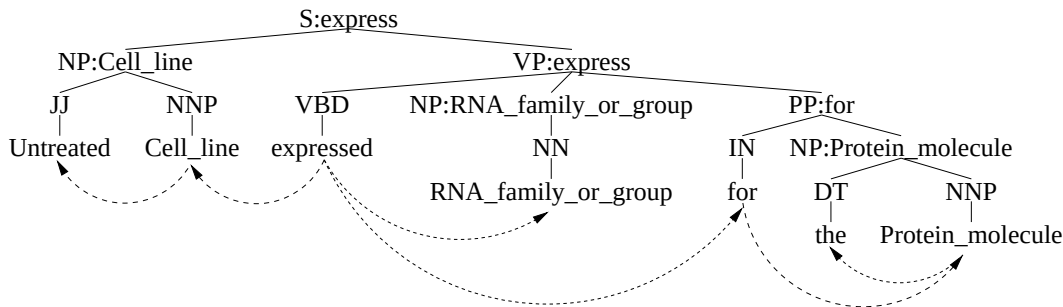


Figure 2: Parse tree for the sentence of Example 1. Entities are substituted with their tags. Phrases are labeled with their syntactic heads. The dependency graph is depicted with dashed directed edges pointing to the governed element.

to the word that differs from the word itself. For example, in Figure 2 “Cell_line” is governed by “express”, while “Protein_molecule” is governed by the preposition “for”.

In this way it is possible to formalize the notion of semantic relation between two entities. A related application of dependency relations concerns the recognition of paraphrase sentences [Lin and Pantel, 2001]. A relation r between two entities c_i and c_j in a tree is the shortest path between c_i and c_j following the dependency relations. For example, in Figure 2 the path between “Cell_line” and “Protein_molecule” is “←express→for→”. There is a path for every pair of entities in the tree. Paths can be considered from both directions since the reverse of a path from A to B is a path from B to A. A large number of different patterns can be extracted, overall we found 172,446 paths in the data. For the sake of interpretability, by inspection, of the outcome of the model in this paper we focused on a subset of these patterns. We selected paths from c_i to c_j where $j > i$ and the pivotal element, the word with no incoming arrows, is a verb v . In addition we imposed the following constraints: c_i is governed by v under an S phrase (i.e., is v ’s surface subject, SUBJ), e.g., “Cell_line” in Figure 2; and one of the following six constraints holds:

1. c_j is governed by v under a VP (i.e., is v ’s direct object, DIR_OBJ), e.g., “RNA_family_or_group” in Figure 2;
2. c_j is governed by v under a PP (i.e., is v ’s indirect object, IND_OBJ), e.g. “Protein_molecule” in Figure 2;
3. c_j is governed by v ’s direct object noun (i.e., is a modifier of the direct object, DIR_OBJ_MOD), e.g. “Virus” in “... influenced *Virus* replication”;
4. c_j is governed by v ’s indirect object noun (i.e., is the indirect object’s modifier, IND_OBJ_MOD), e.g., “Protein_molecule” in “..was induced by *Protein_molecule* stimulation”;
5. c_j is governed by a PP which modifies the direct object (DIR_OBJ_MOD_PP); e.g., “Protein_molecule” in “.. induce overproduction of *Protein_molecule*”;
6. c_j is governed by a PP which modifies the indirect object (IND_OBJ_MOD_PP); e.g., “Lipid” in “..transcribed upon activation with *Lipid*”.

For the sentence in Figure 2 we identify two good patterns: “SUBJ←express→DIR_OBJ” between “Cell_line” and “RNA_family_or_group”, and

“SUBJ←express→for→IND_OBJ”, between “Cell_line” and “Protein_molecule”. Overall we found 7,189 instances of such relations distributed as follows:

Type	Counts	RelFreq
SUBJ-DIR_OBJ	1,746	0.243
SUBJ-IND_OBJ	1,572	0.219
SUBJ-DIR_OBJ_MOD_PP	1,156	0.161
SUBJ-DIR_OBJ_MOD	943	0.131
SUBJ-IND_OBJ_MOD_PP	911	0.127
SUBJ-IND_OBJ_MOD	861	0.120

The data contained 485 types of entity pairs, 3,573 types of patterns and 5,606 entity pair-pattern types.

3.3 Learning relations (Stage 1)

Let us take A to be an ordered pair of GENIA classes; e.g. A = (Protein_domain,DNA_domain_or_region), and B to be a pattern; e.g., B = SUBJ←bind→DIR_OBJ. Our goal is to find relations strongly associated with ordered pairs of classes, i.e., bi-grams AB. This problem is similar to finding *collocations*; e.g., multi-word expressions such as “real estate”, which form idiomatic phrases. Accordingly the simplest method would be to select the most frequent bi-grams. However many bi-grams are frequent because either A or B, or both, are frequent; e.g., SUBJ←induce→DIR_OBJ is among the most frequent pattern for 37 different pairs. Since high frequency can be accidental and, additionally, the method doesn’t provide a natural way for distinguishing relevant from irrelevant bi-grams, we use instead a simple statistical method.

As with collocations a better approach is to estimate if A and B occur together more often than at chance. One formulates a *null hypothesis* H_0 that A and B do not occur together more frequently than expected at chance. Using corpus statistics the probability of $P(AB)$, under H_0 , is computed and H_0 is rejected if $P(AB)$ is beneath the significance level. We use a chi-square test for this purpose. For each observed bi-gram we create a contingency table of the frequencies AB, $\neg AB$, $A \neg B$, and $\neg A \neg B$; e.g., for A = Protein_molecule-DNA_domain_or_region and B = SUBJ←bind→DIR_OBJ the table computed from the corpus would contain respectively the values 6, 161, 24 and 6,998. The chi-square test compares the observed frequencies vs. the frequencies expected under H_0 . Together with the test we use

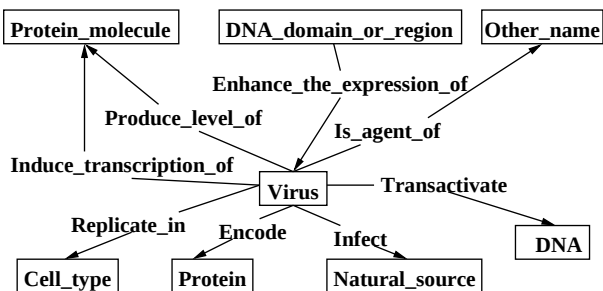


Figure 3: The “Virus” concept with the selected and generalized relations, and related concepts, in the enriched ontology.

the log-likelihood chi-squared statistic:⁴

$$(2) \quad G^2 = 2 \sum_{i,j} o_{ij} \log \frac{o_{ij}}{e_{ij}}$$

where i and j range over the rows and columns of the contingency table, and the expected frequencies are computed off the marginal frequencies in the table. In the previous example G^2 is equal to 16.43, which is above the critical value 7.88 for $\alpha = 0.005$, hence B is accepted as a relevant pattern for A. The following table shows the three highest ranked class pairs for pattern B. There is strong evidence that entities of the protein type tend to bind DNA locations, which is a reasonable conclusion:

B = SUBJ←bind→DIR_OBJ		
A	G^2	Sig
Protein_domain-DNA_domain_or_region	16.43	YES
Protein_family_or_group-DNA_d._or_r.	13.67	YES
Virus-Protein_molecule	7.84	NO

In this study we used $\alpha = 0.005$. In general, α is an adjustable parameter which might be set on held-out data in order to maximize an objective function. We also ignored bi-grams occurring less than 2 times and pairs A, patterns B, occurring less than 4 times. Overall there are 487 suitable AB pairs, 287 (58.6%) have a value for G^2 higher than α .

3.4 Generalization of relations (Stage 2)

Relations can share similar arguments as in “bind” above where, in both significant cases, the direct object is “DNA domain or region” while the subject is some kind of protein. This can be evidence that, in fact, we are facing a more general template holding between superordinates of the arguments found in the first stage. It is desirable, when possible, to learn more general relations such as “Protein SUBJ←bind→DIR_OBJ DNA”, because the learned ontology is more compact and has greater generalization power, i.e., relations apply to more entities. Finding such generalizations is similar to learning *selectional restrictions* of predicates, that is, the preferences that predicates place on the semantic category of their arguments; e.g., that “eat” prefers objects that are “foods”. Several methods have been proposed

⁴Dunning [1993] argues that G^2 is more appropriate than Pearson’s X^2 with sparse data; here they produce similar rankings.

for learning such restrictions. We use here the method proposed in [Clark and Weir, 2002] which is accurate and simple. We use the taxonomy defined in the GENIA ontology, see Figure 1, to generalize arguments of the learned patterns.⁵

Clark and Weir define an algorithm, $top(c, r, s)$, which (adjusting the terminology to our case) takes as input a relation r , a class c and a syntactic slot s , and returns a class c' which is c itself or one of its ancestors, whichever provides the best generalization for $p(r|c, s)$. The method uses the chi-squared test to check if the probability $p(r|c, s)$ is significantly different from $p(r|c', s)$, where c' is the parent of c . If this is false then $p(r|c', s)$ is supposed to provide a good approximation for $p(r|c, s)$, which is interpreted as evidence that (r, s) holds for c' as well. The procedure is iteratively applied until a significant difference is found. The last class considered is the output of the procedure, the concept that best summarizes the class that r “selects” in syntactic slot s . We computed the frequencies of patterns involving superordinate classes summing over the frequencies, from the GENIA corpus, of all descendants of that class for that pattern.

For each relation r , slot s and class c , learned in stage 1, we used Clark and Weir’s method to map c to $top(c, r, s)$. We again used the G^2 statistic and the same α value of 0.005. Using these maps we generalized, when possible, the original 287 patterns learned. The outcome of this process is a set of 240 templates 153 of which have generalized arguments. As an example, the templates above “Protein_domain binds DNA_domain_or_region” and “Protein_family_or_group binds DNA_domain_or_region” are mapped to the generalized template “Protein binds DNA”. Figure 3 depicts the set of labeled relations the concept Virus is involved in, and the respective paired concepts, after stage 1 and 2.

4 Evaluation

We discuss now an evaluation of the model carried out by a biologist and an ontologist, both familiar with GENIA. The biological evaluation focuses mainly on the *precision* of the system; namely, the percentage of all relations selected by the model that, according to the biologist, express correct biological interactions between the arguments of the relation. From the ontological perspective we analyze semantic aspects of the relations, mainly the consistency with the GENIA classes.

4.1 Biological evaluation

The output of stage 1 is a set of 287 patterns, composed of an ordered pair of classes and a semantic relation. 91 of these patterns, involving in one or both arguments the class “Other_name”, were impossible to evaluate and excluded altogether. This GENIA class is a placeholder, for very different sorts of things, which needs partitioning and structuring. Relations involving “Other_name” (e.g., “treat”) might prove correct for a subset of the entities tagged with this

⁵Four of the 36 GENIA corpus class labels, namely, “DNA_substructure”, “DNA_N/A”, “RNA_substructure” and “RNA_N/A”, have no entries in the GENIA ontology, we used them as subordinates of “DNA” and “RNA”, consistently with “Protein_N/A” and “Protein_substructure” which in the ontology are subordinates of “Protein”.

label (e.g., “inflammation”) but false for a different subset (e.g., “gene expression”). Of the remaining 196 patterns 76.5% (150) are correct, i.e., express valid biological facts such as “Protein_molecule induce-phosphorylation-of Protein_molecule”, while 23.5% (43) are incorrect, e.g. “Protein inhibit-expression-of Lipid”. Evaluation involved the exhaustive inspection of the original sentences to verify the intended meaning of the pattern and spot recurring types of errors. Half of the mistakes (22) depend on how we handle coordination, which causes part of the coordinated structure to be included in the relation. For example, the first two DNA entities in the noun phrase “DNA, DNA, and DNA” are governed by the head DNA rather than by, say, the main verb. Thus wrong relations such as “Protein bind-DNA DNA” are generated in addition to good ones such as “Protein bind DNA”. Fixing this problem involves simply the specification of a better dependency structure for coordinations. Finally, 5 errors involved the class “Other_name” embedded somewhere within the relation, suggesting again generalizations that cannot be judged with enough confidence. The remaining errors are probably due to sparse data problems. In this respect we plan to implement an NER system to produce more data and consequently more reliable distributional information. Finally, we notice that, although the GENIA ontology was intended to be a model of cell signaling reactions, it lacks important concepts such as *signaling pathway*. This leads to some errors as in the following case: “An intact TCR signaling pathway is required for p95vav to function.”. In this case we derive the relation: “Protein_molecule is-required-for Protein_molecule” since only “TCR” is annotated as “Protein_molecule” neglecting *signaling pathway*.

To the best of our knowledge we can compare these results with one other study. Reinberger et al. [2004] evaluate (also by means of experts) 165 subject-verb-object relations, extracted from data similar to ours⁶ but with a different approach. They report an accuracy of 42% correct relations. Their method differs from ours in three respects: relations are extracted between nouns rather than entities (i.e., NER is not considered), a shallow parser is used instead of a full parser, and relations are selected by frequency rather than by hypothesis testing. A direct comparison of the methods is not feasible. However, if the difference in accuracy reflects the better quality of our method this is likely to depend on any, or on a combination, of those three factors.

As far as stage 2 is concerned we first removed all relations involving “Other_name” (40 out of 153), which does not have superordinates nor subordinates, and evaluated if the remaining 113 generalized patterns were correct. Of these, 60 (53.1%) provided valid generalizations; e.g., “Protein_molecule induce-phosphorylation-of Amino_acid_monomer” is mapped to “Protein induce-phosphorylation-of Amino_acid_monomer”. Excluding mistakes caused by the fact that the original relation is incorrect, over-generalization is mainly due to the fact that the taxonomy of the GENIA ontology is not a *is-a* hierarchy; e.g., “DNA_substructure” is not a kind of “DNA”, and “Protein”

is not a kind of “Amino_acid”. Generalizations such as selectional restrictions instead seem to hold mainly between classes that share a relation of inclusion. In order to support this kind of inference the structural relations between GENIA classes would need to be clarified.

4.2 Ontological assessment

The 150 patterns validated by the expert are potential new components of the ontology. We compiled GENIA, including the newly learned relations, in OWL (Ontology Web Language [McGuinness and van Harmelen, 2004]) to assess its properties with ontology engineering tools. Ignoring “Other_name”, the GENIA taxonomy branches from two root classes: “Source” and “Substance”. GENIA classes, by design, tend to be *mutually exclusive*, meaning that they should be logically disjoint. Our main objective is to verify the degree to which the new relations adhere to this principle.

To analyze the relations we *align*, i.e., map, “Source” and “Substance” to equivalent classes of another more general ontology. Ideally, the alignment should involve an ontology of the same domain such as TAMBIS [Stevens *et al.*, 2000]. Unfortunately TAMBIS scatters the subordinates of “Source” (organisms, cells, etc.) across different branches, while “Substance” in TAMBIS does not cover protein and nucleic acid-related subordinates of “Substance” in GENIA.⁷ In GENIA substances are classified according to their chemical features rather than biological role, while sources are biological locations where substances are found and their reactions take place. This distinction assumes a stacking of ontology layers within the physical domain where the biological is superimposed to the chemical level. This feature of GENIA makes it suitable for alignment with DOLCE-Lite-Plus (DLP, <http://dolce.semanticweb.org>), a simplified translation of the DOLCE foundational ontology [Gangemi *et al.*, 2003]. DLP specify a suitable distinction between “chemical” and “biological” objects. It features about 200 classes, 150 relations and 500 axioms and has been used in various domains including biomedicine [Saric *et al.*, 2004a].

We aligned “Source” and “Substance” to the biological and chemical classes in DLP. There are 78 types of relations out of 150, 58% of them (45) occur only with one pair of classes, i.e., are monosemous, while 33 have multiple domains or ranges, i.e., are polysemous. Since the root classes of GENIA are disjoint we checked if there are polysemous relations whose domain or range mix up subclasses of “Source” with subclasses of “Substance”. Such relations might not imply logical inconsistency but raise doubts because they suggest the possibility that a class of entities emerged from the data, which is the union of two classes that by definition should be disjoint. Interestingly, there are only 4 such relations out of 78 (5.1%); e.g., “encode”, whose subject can be either “Virus” or “DNA”. In biology, DNA encodes a protein, but biologists sometimes use the verb “metonymically”. By saying that a virus encodes a protein, they actually mean that a virus’ genome contains DNA that encodes a protein. The small number of such cases suggests that relations emerging

⁶The SwissProt corpus, 13 million words of Medline abstracts related to genes and proteins.

⁷Notice that we are not questioning the quality of TAMBIS, but only its fitness for aligning GENIA.

from corpus data are consistent with the most general classes defined in GENIA.

At a finer semantic level relations are composed as follows: 54 (68%) are *eventive*, they encode a conceptualization of chemical reactions as events taking place in biological sources; 81% of the relations between biological and chemical classes are eventive, supporting the claim made in GENIA that biologically relevant chemical reactions involve both a biological and chemical object. Non-eventive relations have either a structural (e.g. “Consist-of”), locative (“Located-in”), or epistemological meaning (“identified-as”).

5 Conclusion and future work

We presented a study on learning semantic relations from text in the domain of molecular biology. We investigated an unsupervised approach using the GENIA ontology and its corpus. Our model is based on a representation of relations as syntactic dependency paths between an ordered pair of named-entities. To select “good” relations, class pairs and dependency paths can be interpreted as bi-grams, and scored with statistical measures of correlation. We showed that it is also possible to generalize over the arguments of the relation using a taxonomy and algorithms for selectional restrictions learning. The results of a biological and ontological analysis of the acquired relations are positive and promising.

Other aspects need to be addressed beyond precision, in particular we are interested in evaluating the recall, i.e., the coverage, of the system,⁸ the precision of alternative selection criteria, and the usefulness of automatically learned relations in information extraction tasks. The latter will imply the identification of synonymic relations; e.g., in the context of Protein-Protein interaction “positively-regulate” is equivalent to “activate”, “up-regulate”, “derepress”, “stimulate” etc. Representing relations as dependency paths one can frame this problem straightforwardly as that of finding paraphrases (e.g. as in [Lin and Pantel, 2001]).

References

- [Berland and Charniak, 1999] M. Berland and E. Charniak. Finding Parts in Very Large Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, 1999.
- [Charniak, 2000] E. Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, 2000.
- [Clark and Weir, 2002] S. Clark and D. Weir. Class-Based Probability Estimation Using a Semantic Hierarchy. *Computational Linguistics*, 28, 2002.
- [Craven and Kumlien, 1999] M. Craven and J. Kumlien. Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB 1999)*, 1999.
- [Dunning, 1993] T. Dunning. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1), 1993.
- [Friedman et al., 2001] C. Friedman, P. Kra, H. Yu, M. Krauthammer, and A. Rzhetsky. GENIES: A Natural-Language Processing System for the Extraction of Molecular Pathways from Journal Articles. *Bioinformatics*, 17(1), 2001.
- [Gangemi et al., 2003] A. Gangemi, Guarino N., C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3), 2003.
- [Kazama et al., 2002] J. Kazama, T. Makino, Y. Ohta, and J. Tsujii. Tuning Support Vector Machines for Biomedical Named Entity Recognition. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*, 2002.
- [Lin and Pantel, 2001] D. Lin and P. Pantel. DIRT - Discovery of Inference Rules from Text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2001)*, 2001.
- [McGuinness and van Harmelen, 2004] D. McGuinness and F. van Harmelen. Owl Web Ontology Language Overview. In *W3C Recommendations*: <http://www.w3c.org/TR/owl-features/>, 2004.
- [Ohta et al., 2002] Y. Ohta, Y. Tateisi, J. Kim, H. Mima, and J. Tsujii. The GENIA Corpus: An Annotated Research Abstract Corpus in the Molecular Biology Domain. In *Proceedings of Human Language Technology (HLT 2002)*, 2002.
- [Pantel and Ravichandran, 2004] P. Pantel and D. Ravichandran. Automatically Labeling Semantic Classes. In *Proceedings of HLT-NAACL 2004*, 2004.
- [Pustejovsky et al., 2002] J. Pustejovsky, J. Castaño, J. Zhang, B. Cochran, and M. Kotechi. Robust Relational Parsing over Biomedical Literature: Extracting Inhibit Relations. In *Proceedings of the Pacific Symposium on Biocomputing*, 2002, 2002.
- [Ratsch et al., 2003] E. Ratsch, J. Schultz, J. Saric, P. Cimiano, U. Wittig, U. Reyle, and I. Rojas. Developing a Protein Interactions Ontology. *Comparative and Functional Genomics*, 4(1):85–89, 2003.
- [Reinberger et al., 2004] M-L Reinberger, P. Spyns, and A.J. Pretorius. Automatic Initiation of an Ontology. In *Proceedings of ODBase 2004*, 2004.
- [Rojas et al., 2002] I. Rojas, L. Bernardi, E. Ratsch, R. Kania, U. Wittig, and J. Saric. A Database System for the Analysis of Biochemical Pathways. In *Silico Biology 2, 0007*, 2002.
- [Rosario and Hearst, 2004] B. Rosario and M. Hearst. Classifying Semantic Relations in Bioscience Text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, 2004.
- [Saric et al., 2004a] J. Saric, A. Gangemi, E. Ratsch, and I. Rojas. Modelling gene expression. In *Proceedings of the Workshop on Models and Metaphors from Biology to Bioinformatics Tools (NETTAB 2004)*, 2004.
- [Saric et al., 2004b] J. Saric, L.J. Jensen, R. Ouzounova, I. Rojas, and P. Bork. Extraction of Regulatory Gene Expression Networks from PubMed. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, 2004.
- [Stevens et al., 2000] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N.W. Paton, C.A. Goble, and A. Brass. TAM-BIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2), 2000.
- [Swanson and Smalheiser, 1997] D.R. Swanson and N.R. Smalheiser. An Interactive System for Finding Complementary Literatures: A Stimulus to Scientific Discovery. *Artificial Intelligence*, 91(2), 1997.

⁸Which is more problematic because it requires, in principle, considering a very large number of discarded relations.

Feature Selection Based on the Shapley Value

Shay Cohen and Eytan Ruppin

School of Computer Sciences
Tel-Aviv University, Tel-Aviv 69978, Israel
{cshay, ruppin}@post.tau.ac.il

Gideon Dror

Department of Computer Science
Academic College of Tel-Aviv Yaffo, Tel-Aviv, 64044, Israel
gideon@mta.ac.il

Abstract

We present and study the Contribution-Selection algorithm (CSA), a novel algorithm for feature selection. The algorithm is based on the Multiperturbation Shapley Analysis, a framework which relies on game theory to estimate usefulness. The algorithm iteratively estimates the usefulness of features and selects them accordingly, using either forward selection or backward elimination. Empirical comparison with several other existing feature selection methods shows that the backward elimination variant of CSA leads to the most accurate classification results on an array of datasets.

1 Introduction

Feature selection refers to the problem of selecting input variables, otherwise called features, that are relevant to predicting a target value for each instance in a dataset. Feature selection has several potential benefits: defying the curse of dimensionality to enhance the prediction performance, reducing measurement and storage requirements and reducing training and prediction times. This paper focuses on the first issue, namely selecting input variables in an attempt to maximize the performance of a classifier on previously unseen data.

In this paper, we suggest to recast the problem of feature selection in the context of coalitional games, a notion from game theory. This perspective yields an iterative algorithm for feature selection, the Contribution-Selection algorithm (CSA), intent on optimizing the performance of the classifier on unseen data. The algorithm combines both the filter and wrapper approaches. However, unlike filter methods, features are reranked on each step by using the classifier as a black box. The ranking is based on the Shapley value [Shapley, 1953], a well known concept from game theory, to estimate the importance of each feature for the task at hand, specifically taking into account interactions between features.

Throughout the paper we use the following notations. The distribution from which the dataset instances are drawn is represented by two variables $(\mathbf{X}, Y)$, where $\mathbf{X} = (X_1, \dots, X_n)$ represents the input variables (vector of features), and Y represents a discrete target value (class) for $\mathbf{X}$. Three sets

containing i.i.d. sampled instances of $(\mathbf{X}, Y)$ of the form $\{\mathbf{x}_k, y_k\}$ are available: Train, Validation and Test representing the training set, validation set and test set respectively. Given an induction algorithm and a set $S \subseteq \{1, \dots, n\}$, $f_S(x)$ stands for a classifier constructed from the training set using the induction algorithm, after its input variables were narrowed down to the ones in S , namely $f_S(x)$ labels each instance of the form $(x_{i_1}, \dots, x_{i_{|S|}})$, $i_j \in S$, $1 \leq j \leq |S|$ with a value in the domain of Y . The task of feature selection is to choose a subset S of the input variables, that would maximize the performance of the classifier on the test set. In what follows we shall focus on optimizing accuracy of the classifier, although we could as easily optimize other performance measure such as the area under the ROC curve, balanced error rate etc.

The rest of this paper is organized as follows: Section 2 introduces the necessary background from game theory with a detailed description of the CSA algorithm; Section 3 provides an empirical comparison of CSA with several other feature selection methods, accompanied by an analysis of the results; Section 4 discusses the empirical results and provides further insights to the success of the backward elimination version of the CSA algorithm.

2 Classification as a Coalitional Game

Cooperative game theory introduces the concept of “coalitional games”, in which a set of players is associated with a payoff, a real function that denotes the benefit achieved by different sub-coalitions in a game. Formally, a *coalitional game* is defined by a pair (N, v) where $N = \{1, \dots, n\}$ is the set of all *players* and $v(S)$, for every $S \subseteq N$, is a real number associating a worth with the *coalition* S . Game theory further pursues the question of representing the contribution of each player to the game by constructing a value function, which assigns a real-value to each player. The values correspond to the contribution of the players in achieving a high payoff.

The contribution value calculation is based on the Shapley value [Shapley, 1953]. An intuitive example of the potential use of the Shapley value can be provided in an academic setting. Assume that you are a Professor running a lab, and, once and for all, you have decided to distribute the yearly bonus to your students in a fair manner, that reflects the actual con-

tribution of each student to the academic success of the lab. During the year, the students form spontaneous “coalitions” of groups of students, each such group works and publishes a paper summarizing its work (these coalitions may also be assembled by the Professor). Every paper gets a rank, (e.g., its impact factor), composing its “payoff function”. Based on this annual data of the students’ coalitions and their associated payoffs, the Shapley value provides a fair and efficient way to distribute the bonus to each individual student according to his contribution over the year.

The Shapley value is defined as follows. Let the *marginal importance* of player i to a coalition S , with $i \notin S$, be

$$\Delta_i(S) = v(S \cup \{i\}) - v(S). \quad (1)$$

Then, the Shapley value is defined by the payoff

$$\Phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} \Delta_i(S_i(\pi)) \quad (2)$$

where Π is the set of permutations over N , and $S_i(\pi)$ is the set of players appearing before the i th player in permutation π . The Shapley value of a player is a weighted mean of its marginal value, averaged over all possible subsets of players.

Transforming these game theory concepts into the arena of feature selection, in which one attempts at estimating the contribution of each feature in generating a classifier, the players N are mapped to the features of a dataset and the payoff is represented by a real-valued function $v(S)$, which measures the performance of a classifier generated using the set of features S . Finally, the usage of the Shapley value for feature selection may be justified by its axiomatic qualities:

Axiom 1 (Normalization or Pareto optimality) For any game (N, v) it holds that $\sum_{i \in N} \Phi_i(v) = v(N)$

In the context of feature selection, this axiom implies that the performance on the dataset is divided fully between the different features.

Axiom 2 (Permutation invariance or symmetry) For any (N, v) and permutation π on N it holds that $\Phi_i(v) = \Phi_{\pi(i)}(\pi v)$

This axiom implies that the value is not altered by arbitrarily renaming or reordering the features.

Axiom 3 (Preservation of carrier or dummy-property) For any game (N, v) such that $v(S \cup \{i\}) = v(S)$ for every $S \subseteq N$ it holds that $\Phi_i(v) = 0$

This axiom implies that a dummy feature that does not influence the classifier’s performance indeed receives a contribution value 0.

Axiom 4 (Additivity or aggregation) For any two games (N, v) and (N, w) it holds that $\Phi_i(v + w) = \Phi_i(v) + \Phi_i(w)$ where $(v + w)(S) = v(S) + w(S)$

This axiom applies to a combination of two different payoffs based on the same set of features. For a classification task these may be, for example, accuracy and area under the ROC curve or false positive rate and false negative rate. In such

case, the Shapley value of a feature, that measures its contribution to the combined performance measure, is just the sum of the corresponding Shapley values. The linearity of the Shapley value is a consequence of this property. Namely, if the payoff function v is multiplied by a real number α then all Shapley values are scaled by α namely $\Phi_i(\alpha v) = \alpha \Phi_i(v)$. In other words, multiplying the performance measure by a constant does not change the ranking of the features, a vital property for any scheme that ranks features by their ‘importance’.

2.1 Estimating Features Contribution Using the MSA

The calculation of the Shapley value requires summing over all possible subsets of players, which is impractical in our case. [Keinan *et al.*, 2004] have presented an unbiased estimator for the Shapley value by uniformly sampling permutations from Π . Still, the estimator considers both large and small features sets to calculate the contribution values. In our feature selection algorithm, we use the Shapley value heuristically to estimate the contribution value of a feature for the task of feature selection. Since in most realistic cases we assume that the size d of significant interactions between features is much smaller than the number of features, n , we will limit ourselves to calculating the contribution value from permutations sampled from the whole set of players, with d being a bound on the permutation size. Notice that most filter methods are equivalent to using $d = 1$ where no interactions are taken into account. Feature selection using Random Forests [Breiman, 2001] is equivalent to $d \simeq \sqrt{n}$. The bounded estimated contribution value becomes

$$\varphi_i(v) = \frac{1}{|\Pi_d|} \sum_{\pi \in \Pi_d} \Delta_i(S_i(\pi))$$

where Π_d is the set of sampled permutations on subsets of size d . The usage of bounded sets coupled with the method for the Shapley value estimation, yields an efficient and robust way to estimate the contribution of a feature to the task of classification. For a detailed discussion of the MSA framework and its theoretical background see [Keinan *et al.*, 2004].

2.2 The Contribution-Selection Algorithm

The Contribution-Selection algorithm (CSA), described in detail in Figure 1, is iterative in nature, and can either adopt a forward selection or backward elimination approach. Using the subroutine *contribution*, it ranks each feature according to its contribution value, and then selects s features with the highest contribution values with forward selection (using the sub-routine *selection*)¹, or eliminates e features with the lowest contribution values with backward elimination (using *elimination*). It repeats the phases of calculating the contribution values of the remaining features given those already selected (or eliminated), and selecting (or eliminating) new features, until the contribution values of all candidate features

¹Alternatively, the *selection* sub-routine can use a forward selection technique instead of s ; features are added in ascending order of their contribution values, as long as the classifier’s performance improves.

Contribution-Selection-Algorithm($F; \Delta, d, s$)

1. $selected := \phi$
2. for each $f \in F \setminus selected$
 - 2.1. $C_f := \text{contribution}(f, selected; d)$
3. if $\max_f C_f > \Delta$
 - 3.1. $selected := selected \cup \text{selection}(\{C_f\}; s, \Delta)$
 - 3.2. goto 2
- else
- 3.3. return selected

Figure 1: The Contribution-Selection algorithm in its forward selection version. F is the input set of features, Δ is a contribution value threshold, d is the maximal permutation size for calculating the contribution values, s is the number of features selected in each phase. The *contribution* routine calculates the contribution value of feature f with the pay off function described in this section. The *selection* routine selects at most s features with highest contribution values that exceed Δ . In the backward elimination version, the *selection* sub-routine is replaced with an *elimination* sub-routine which eliminates e features in each phase and the halting criterion is changed accordingly.

exceed a contribution threshold Δ with forward selection (or fall below a contribution threshold Δ with backward elimination).

The algorithm, without further specification of the *contribution* sub-routine, is a generalization of filter methods. However, the main idea of the algorithm is that the *contribution* sub-routine, unlike common filter methods, returns a contribution value for each feature according to its assistance in improving the classifier’s performance, which is generated using a specific induction algorithm, and in conjunction with other features. Using the notation in Section 2 and assuming one optimizes the accuracy level of the classifier, the *contribution* sub-routine for forward selection calculates the contribution values using the following payoff function $v(S)$:

1. $S := S \cup selected$
2. Generate a classifier $f_S(x)$ from the training set, Train
3. Evaluate $f_S(x)$ for all examples of the validation set, Validation
4. Return the accuracy level, defined as $v(S) = \frac{|\{x | f_S(x)=y, (x,y) \in Validation\}|}{|Validation|}$

The case $S = \phi$ is an end case which is handled by returning the number of instances in the largest class divided by the total number of instances (a classifier which always selects the most frequent class). Backward elimination is quite similar, and the payoff is calculated by sampling permutations from the set of features left after each phase of elimination. The maximal permutation size d has an important role in deciding the contribution values of the different features, and should be selected in a way that ensures that different combinations of features that interact together are inspected. Its impact is demonstrated in Section 3.

The number of selected features s for the *selection* sub-routine controls the redundancies of the selected features; the

Name	Classes	Features	Train Size	Test Size
Reuters1	3	1579	145	145
Reuters2	3	1587	164	164
Arrhythmia	2	278	280	140
Internet Ads	2	1558	2200	800
Dexter	2	20000	300	300
Arcene	2	10000	100	100
I2000	2	2000	40	22

Table 1: Description of datasets used.

higher s is, the more likely that features with redundant contribution will be selected. Although $s = 1$ minimizes the redundancy dependencies of the features, increasing s accelerates the algorithm’s convergence. The algorithm’s halting criterion depends on Δ , which designates a trade-off between the number of selected features, and the performance of the classifier on the validation set. With the forward selection version, choosing $\Delta = 0$ means that CSA selects features as long as there exists a feature that is likely to improve the classifier’s performance, and selects smaller sets of features as Δ is increased. Increasing Δ has the opposite effect on the size of the final set of features. The more intuitive halting criterion, to stop when no further performance gain is achieved, is too restrictive, while CSA’s halting criterion enables the selection of features proved useful at later stages, as verified empirically over several datasets.

3 Results

3.1 The Data and Benchmark Algorithms

To test CSA empirically we ran a number of experiments on seven real-world datasets with number of features ranging from 278 to 20,000 (Table 1): the Reuters1 dataset and the Reuters2 dataset both constructed following [Koller and Sahami, 1996] using the Reuters-21578 document collection; the Arrhythmia database from the UCI repository [Perkins *et al.*, 2003]; the Internet Advertisements database from the UCI repository [Blake and Merz, 1998] which was collected for the research of identifying advertisements in web pages, the Dexter text categorization dataset and the Arcene cancer dataset, both from the NIPS 2003 workshop on feature selection [Guyon, 2003] and the I2000 microarray colon cancer dataset [Alon *et al.*, 1999].

In principle, CSA can work with any induction algorithm L . However, due to computational constraints we focused on fast induction algorithms or algorithms that may be efficiently combined into CSA. We experimented with Naive Bayes, C4.5 and 1NN. For each of the datasets, we measured the training set accuracy of each classifier using ten-fold cross validation on the whole set features. For each dataset, all subsequent work used the induction algorithm L , that gave the highest cross validation accuracy, as detailed in Table 2.

Eight different feature selection schemes were then compared on the datasets described above:

- The induction algorithm L without performing feature selection to serve as a baseline.
- Regularized linear SVM using the SVM^{light} package [Joachims, 1999]. Datasets that had more than two

Dataset	L	s (Fwd.)	e (Bwd.)	d	t
Reuters1	NB	1	100	20	1500
Reuters2	NB	1	100	20	1800
Arrhythmia	C4.5	1	50	20	500
Internet Ads	1NN	1	100	20	1500
Dexter	C4.5	50	50	12	3500
Arcene	C4.5	100	100	5	10000
I2000	C4.5	100	100	3	2000

Table 2: The parameters and the classifier used with the CSA algorithm for each dataset. L is in the induction algorithm used with CSA (NB being Naive Bayes), s is the number of features selected in forward selection in each phase, e is the number of features eliminated in backward elimination in each phase, d is the permutation size and t is the number of permutations sampled to estimate the contribution values. For an explanation how hyperparameters are chosen, see text.

classes were split into few binary classification problems.

- Filtering using mutual information and classification using L . We binned continuous domains to estimate the mutual information.
- Filtering using the Pearson correlation coefficient and classification using L .
- Random Forests feature selection [Breiman, 2001] and classification using L .
- Feature selection using forward selection wrapper. Since simple wrapper greedily selects a feature that most improves the classifier's validation performance, it is equivalent to forward selection CSA with $d = 1$.
- Classification using L after performing feature selection with forward selection CSA and parameters as described in Table 2. The parameters d and t were chosen such that the expected number of times that each feature is sampled is higher than 5. The contribution value threshold for stopping selection was $\Delta = 0$. termination of feature selection was fixed by choosing a contribution value threshold $\Delta = 0$. No hyperparameter selection was performed on either d , t or Δ .
- Classification using L after performing feature selection with backward elimination CSA and parameters as described in Table 2. The parameters d and t were chosen such that the expected number of times that each feature is sampled is higher than 5. The contribution value threshold for stopping elimination was $\Delta = 0$. No hyperparameter selection was performed on either d , t or Δ .

To avoid overfitting on the validation set used for calculating the payoff with CSA, we used m-fold cross validation instead of a single *Validation* set.

3.2 Feature Selection and Classification Results

Table 3 summarizes the classifiers' performance on the test set and the number of features selected in each of the experiments. The accuracy levels are the fraction of correctly classified test set instances:

- *The Reuters1 dataset.* Feature selection using Random Forests did best, yielding accuracy level of 100% with 30 features. Not too far behind is the CSA in its backward elimination version (98.6% with 51 features). [Koller and Sahami, 1996], for example, report that the Markov Blanket algorithm yields approximately 600 selected features with accuracy levels of 95% to 96% on this dataset.
- *The Reuters2 dataset.* CSA with backward elimination did best, yielding accuracy level of 93% with 109 features. For comparison, [Koller and Sahami, 1996] report that the Markov Blanket algorithm yields approximately 600 selected features with accuracy levels of 89% to 93% on this dataset.
- *The Arrhythmia dataset.* This dataset is considered to be a difficult one. CSA with backward elimination did best, yielding an accuracy level of 84% with 21 features. Forward selection with higher depth value ($d = 20$) did better than wrapper, implying that one should consider many features concomitantly to perform good feature selection for this dataset. For comparison, the grafting algorithm [Perkins *et al.*, 2003] yields an accuracy level of approximately 75% on this dataset.
- *The Internet Ads dataset.* All the algorithms did approximately the same, leading to accuracy levels between 94% and 96% with CSA slightly outperforming the others. Interestingly enough, the wrapper algorithm did not select any feature; in the first phase, the 1NN algorithm had neighbors from both classes with the same distance for each feature checked, leading to arbitrary selection of one of the classes, and the classifier's performance was constant through all the phase, yielding zero contribution values. However, when selecting the higher depth levels, the simple 1NN algorithm was boosted up to outperform classifiers such as SVM.
- *The Dexter dataset.* For the Dexter dataset, we used algorithm L (C4.5 decision trees) only for the process of feature selection, and Linear SVM to perform the actual prediction on the features selected. This was done because C4.5 did not give satisfying accuracy levels for any of the feature selection algorithms, and it is impractical to use SVM with CSA for large datasets. To overcome the difference between the classifiers performing feature selection and the classifier used for the actual classification, we added an optimization phase for the forward selection algorithm after it stopped. In this phase, a ten-fold cross-validation is performed on the dataset in a similar way to the one used to optimize filter methods. The simple mutual information feature selection performed best, followed closely by the Contribution-Selection algorithm in its backward elimination version and by Random Forests. This implies that in Dexter the contribution of single features significantly outweigh the contribution of feature combinations for the task of classification. The forward selection algorithm did as well as *Linear SVM* without feature selection, but with a significantly lower number of features.
- *The Arcene dataset.* Here, just as in the case of Dexter, we use C4.5 for the process of feature selection, and Linear SVM to perform the actual prediction on the features

Dataset	Wrapper	Fwd.	Bwd.
Reuters1	92.4 (7)	96.5 (10)	98.6 (51)
Reuters2	91.4 (5)	90.1 (14)	93.2 (109)
Arrhythmia	70 (5)	74.2 (28)	84.2 (21)
Internet Ads	-	95.6 (8)	96.1 (158)
Dexter	80 (10)	92.6 (100)	93.3 (717)
Arcene	58 (7)	81 (600)	86 (7200)
I2000	86.3 (550)	86.3 (500)	90.9 (1100)

No FS	SVM	Corr.	MI	RF
84.1	94.4	90.3 (20)	94.4 (20)	100 (30)
81.1	91.4	88.4 (20)	90.2 (5)	87.2 (21)
76.4	80	71.4 (20)	70 (20)	80 (40)
94.7	93.5	94.2 (15)	95.75 (70)	95.6 (10)
92.6	92.6	92.6 (1240)	94 (230)	93.3 (800)
83	83	83% (6600)	81 (5600)	82 (6000)
86.3	72.7	81.8 (260)	90.9 (1060)	86.3 (100)

Table 3: Comparison of accuracy levels and number of features selected in the different datasets. Upper table: Wrapper and Fwd/Bwd (CSA with forward selection/backward elimination with parameters from Table 2). Bottom table: No FS (no feature selection), SVM (linear SVM without feature selection), Corr (feature selection using Pearson correlation), MI (feature selection using mutual information), RF (feature selection using Random Forests). Accuracy levels are calculated by counting the number of misclassified instances and given in percentages. The number of features selected is given in brackets.

selected. The CSA with backward elimination obtained better performance than the rest of the algorithms.

- *The I2000 dataset.* CSA with backward elimination and feature selection using mutual information yielded the best results. The poor performance of CSA with forward selection can be explained by the poverty of data comparing to the number of features; the algorithm selected in the first phases features which explain well the training data by coincidence, and avoided from selecting features that truly contribution to the task of classification. This phenomenon is explained in portrait in Section 3.3.

In summary, in 5 out of the 7 datasets, CSA with backward elimination achieved the best results. In all other cases, CSA achieved the second best result.

3.3 A Closer Inspection of the Results

The MSA, intent on capturing correctly the contribution of elements to a task, enables us to examine the distribution of the contribution values of the features. Figure 2 depicts a log-log plot of the distribution of the contribution values in the first phase for *Arrhythmia* and *Dexter*, prior to making any feature selection. This distribution follows a scale-free power law, implying that large contribution values (in absolute value) are very rare, while small ones are quite common, justifying quantitatively the need of feature selection. The other datasets were also observed to possess similar power law characteristic.

The behavior of the algorithm through the process of feature selection/elimination is displayed in Figure 3; after the forward selection algorithm identifies the significant features in the first few phases, there is a sharp decrease in the contribution values of the features selected in the following phases,

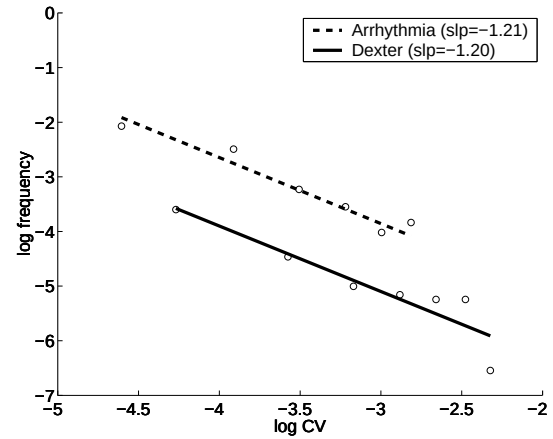


Figure 2: Power-law distribution of contribution values. This log-log plot of the distribution of the contribution values (absolute value) in the first phase for *Arrhythmia* and *Dexter*, prior to making any feature selection, demonstrates a power law behavior. The corresponding plots for the other datasets show identical power-law characteristics (though with different slopes), and were eliminated for the sake of clarity.

while with backward elimination, there is a gradual and rather stable increase in the contribution values of the eliminated features. The peaks in the graph of the contribution values in Figure 3A demonstrate that the contribution values do change as the CSA iterates. In this case, the selection of a single feature considerably increased the contribution value of another feature, pointing at intricate dependencies between features.

Figures 2 and 3 also assist in explaining why backward elimination usually outperforms several feature selection methods, including forward selection; due to the high dimensionality of the datasets, a feature that assists in prediction merely by coincidence, may be selected, on the account of other truly informative features. Forward selection is penalized severely in such case: among the few significant features, some will not be chosen. However, backward elimination always maintains the significant features in the non eliminated set; a feature that truly enhances the classifier's generalization will do so for the validation set as well, and will not be eliminated. This leads to a more stable generalization behavior for backward elimination on the test set through the algorithm's progress (Figure 3).

4 Final Notes

The Contribution-Selection algorithm presented in this paper views the task of feature selection in the context of coalitional games. It uses a wrapper-like technique combined with a novel ranking method which is based on the Shapley contribution values of the features to the classification accuracy. The CSA works in an iterative manner, each time selecting new features (or eliminating them) while taking into account the features that were selected (or eliminated) so far.

CSA, similarly to wrapper algorithms, is restricted in the selection of the induction algorithm used for evaluating features sets, due to time limitations. This problem can be reduced by parallelizing, an advantage not shared by other

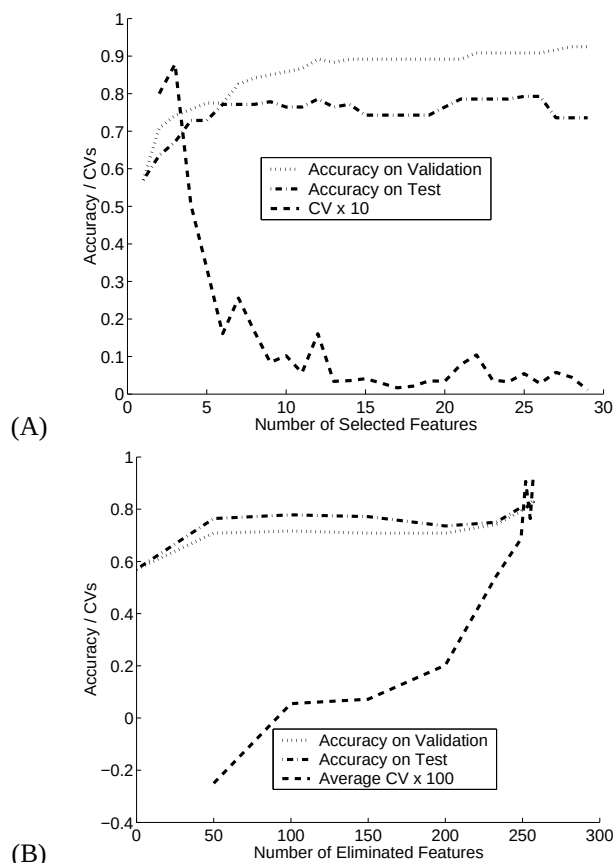


Figure 3: Prediction accuracy and feature contribution during forward selection (A) and backward elimination (B) for the Arrhythmia dataset. Both figures show how the performance of the C4.5 classifier improves on the validation set as the algorithm selects (eliminates) new features, while the contribution values of the selected features decrease (increase). The backward elimination generalizes better on the test set through the algorithm's progress. The behavior for the other datasets is similar.

wrapper algorithms which use search methods such as hill climbing; At each phase the permutations can be computed in parallel and upon completion combined to obtain an estimate of contribution values. Furthermore, as the algorithm progresses, the number of candidate features for either selection (forward selection) or elimination (backward elimination) decreases. Consequently, the number of permutations sampled may be reduced, speeding up the algorithm significantly. The restriction in selecting the learning algorithm for CSA does not apply to the prediction once the features are selected. After a set of features is found by the CSA, it may be used by any induction algorithm as demonstrated in section 3.2 with the Dexter and Arcene datasets.

We verified that the feature sets selected by CSA are significantly different than those selected by filter methods, and Random Forests. It turns out that the first “strong” features are selected by most methods. But within few iterations, filters and CSA select entirely different features due to the fact that the contribution values of the candidate features are modified, sometimes drastically, according to the already selected

features.

The CSA was tested on number of datasets, and the results show that the algorithm can improve the performance of the classifier, and successfully compete with an existing array of feature selection methods, especially in cases where the features interact with each other; in such cases performing feature selection with a permutation size higher than one, namely not using the common greedy wrapper approach, can enhance the classifier's performance significantly.

The results successfully demonstrate the value of applying game theory concepts to feature selection. While the forward selection version of the algorithm is competitive with other feature selection methods, our experiments show that overall, the backward elimination version is superior to them, and produces features sets which can be used to generate a highly performing classifier.

References

- [Alon *et al.*, 1999] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci.*, 96:6745–6750, 1999.
- [Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.
- [Breiman, 2001] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [Guyon, 2003] I. Guyon. Design of experiments for the NIPS 2003 variable selection benchmark. *NIPS 2003 workshop on feature extraction and feature selection*, 2003.
- [Joachims, 1999] T. Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.
- [Keinan *et al.*, 2004] A. Keinan, B. Sandbank, C. Hilgetag, I. Meilijson, and E. Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural Computation*, 16(9), 2004.
- [Koller and Sahami, 1996] D. Koller and M. Sahami. Toward optimal feature selection. *Proceedings of the 13th International Conference on Machine Learning (ML)*, pages 284–292, 1996.
- [Perkins *et al.*, 2003] S. Perkins, K. Lackner, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.
- [Shapley, 1953] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume II of *Annals of Mathematics Studies* 28, pages 307–317. Princeton University Press, Princeton, 1953.

Stacked Sequential Learning

William W. Cohen

Center for Automated Learning & Discovery
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Vitor R. Carvalho

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
vitor@cs.cmu.edu

Abstract

We describe a new sequential learning scheme called “stacked sequential learning”. Stacked sequential learning is a meta-learning algorithm, in which an arbitrary base learner is augmented so as to make it aware of the labels of nearby examples. We evaluate the method on several “sequential partitioning problems”, which are characterized by long runs of identical labels. We demonstrate that on these problems, sequential stacking consistently improves the performance of non-sequential base learners; that sequential stacking often improves performance of learners (such as CRFs) that are designed specifically for sequential tasks; and that a sequentially stacked maximum-entropy learner generally outperforms CRFs.

1 Introduction

In this paper, we will consider the application of sequential probabilistic learners to *sequential partitioning tasks*. Sequential partitioning tasks are sequential classification tasks characterized by long runs of identical labels: examples of these tasks include document analysis, video segmentation, and gene finding.

Motivated by some anomalous behavior observed for one sequential learning method on a particular partitioning task, we will derive a new learning scheme called *stacked sequential learning*. Like boosting, stacked sequential learning is a meta-learning method, in which an arbitrary base learner is augmented—in this case, by making the learner aware of the labels of nearby examples. Sequential stacking is simple to implement, can be applied to virtually any base learner, and imposes only a constant overhead in training time: in our implementation, the sequentially stacked version of the base learner A trains about seven times more slowly than A .

In experiments on several partitioning tasks, sequential stacking consistently improves the performance of non-sequential base learners. More surprisingly, sequential stacking also often improves performance of learners specifically designed for sequential tasks, such as conditional random fields and discriminatively trained HMMs. Finally, on our set of benchmark problems, a sequentially stacked maximum-

entropy learner generally outperforms conditional random fields.

2 Motivation: A Hard Task for MEMMs

To motivate the novel learning method that we will describe below, we will first analyze the behavior of one well-known sequential learner on a particular real-world problem. In a recent paper [Carvalho and Cohen, 2004], we evaluated a number of sequential learning methods on the problem of recognizing the “signature” section of an email message. Each line of an email message was represented with a set of hand-crafted features, such as “line contains a possible phone number”, “line is blank”, etc. Each email message was represented as a vector $\mathbf{x}$ of feature-vectors $x_1, \dots, x_n$, where x_i is the feature-vector representation of the i -th line of the message. A line was labeled as *positive* if it was part of a signature section, and *negative* otherwise. The labels for a message were represented as another vector $\mathbf{y}$, where y_i is the label for line i .

The dataset contains 33,013 labeled lines from 617 email messages. About 10% of the lines are labeled “positive”. Signature sections always fall at the end of a message, usually in the last 10 lines. In the experiments below, the data was split into a training set (of 438 sequences/emails), and a test set with the remaining sequences, and we used the “basic” feature set from Carvalho & Cohen.

The complete dataset is represented as a set S of examples $S = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_t, \mathbf{y}_t), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$. *Sequential learning* is the problem of learning, from such a dataset, a *sequential classifier*—i.e., a function f such that $f(\mathbf{x})$ produces a vector of class labels $\mathbf{y}$. Clearly, any ordinary non-sequential learning algorithm can be used for sequential learning, by ignoring the sequential nature of the data¹.

In the previous paper [Carvalho and Cohen, 2004], we reported results for several non-sequential and sequential learners on the signature-detection problem, including a non-sequential maximum entropy learner [Berger *et al.*, 1996]

¹Specifically, one could build a dataset of non-sequential examples $(x_{t,i}, y_{t,i})$ from S , and use it to train a classifier g that maps a single feature-vector x to a label y . One can then use g to classify each instance x_i in the vector $\mathbf{x} = \langle x_1, \dots, x_n \rangle$ separately, ignoring its sequential position, and append the resulting predictions y_i into an output vector $\mathbf{y}$.

Method	Noise	Error	Min Error
ME		3.47	3.20
MEMM		31.83	4.26
CRF		1.17	1.17
MEMM	10%	2.18	2.18
CRF	10%	1.85	1.84

Table 1: Performance of several sequential learners on the signature-detection problem.

(henceforth ME) and conditional random fields [Lafferty *et al.*, 2001] (henceforth CRFs). Another plausible sequential learning method to apply to this task are *maximum-entropy Markov models* (MEMMs) [McCallum *et al.*, 2000], also called *maximum-entropy taggers* [Ratnaparkhi, 1999], *conditional Markov models* [Klein and Manning, 2002], and *recurrent sliding windows* [Dietterich, 2002]. In this model, the conditional probability of a label sequence y given an instance sequence x is defined to be $\Pr(y|x) = \prod_i \Pr(y_i|y_{i-1}, x_i)$. The *local model* $\Pr(y_i|y_{i-1}, x_i)$ is learned as follows. First one constructs an *extended dataset*, which is a collection of non-sequential examples of the form $((x_i, y_{i-1}), y_i)$, where (x_i, y_{i-1}) denotes an instance in which the original feature vector for x_i is augmented by adding a feature for y_{i-1} . We will call (x_i, y_{i-1}) an *extended instance*, and call y_{i-1} a *history feature*. Note that y_i is the class label for the extended example $((x_i, y_{i-1}), y_i)$.

After constructing extended instances, one trains a maximum-entropy conditional model from the extended dataset. Inference is done by using a Viterbi search to find the best label sequence y .

MEMMs have a number of nice properties. Relative to the more recently-proposed CRF model, MEMMs are easy to implement, and (since no inference is done at learning time) relatively quick to train. MEMMs can also be easily generalized by replacing the local model with one that uses a longer “history” of k previous labels—i.e., a model of the form $\Pr(y_i|y_{i-1}, \dots, y_{i-k}, x_i)$ —and replacing the Viterbi search with a beam search. Such a learner scales well with the history size and number of possible classes y .

Unfortunately, as Table 1 shows, MEMMs perform extremely badly on the signature-detection problem, with an error rate many times the error rate of CRFs. In fact, on this problem, MEMMs perform much worse than the non-sequential maximum-entropy learner ME.²

The MEMM’s performance is better if one changes the threshold used to classify examples. Letting $\hat{p}_i$ be the probability $\Pr(y_i = +|x_i, y_{i-1})$ as computed by MEMM, we found, for each learner, the threshold θ such the rule $[(y_i = +) \Leftrightarrow (\hat{p}_i > \theta)]$ gives the lowest *test* error rate. The column labeled “Min Error” in Table 1 gives this “best possible” result. The “Min Error” for MEMMs is much improved, but still higher than non-sequential ME.

The high error occurs because on many test email mes-

sages, the learned MEMM makes a false positive classification somewhere before the signature starts, and then “gets stuck” and marks every subsequent line as part of a signature. This behavior is not consistent with previously-described limitations of MEMMs. It is known that MEMMs can represent only a proper subset of the distributions that can be represented by CRFs [Lafferty *et al.*, 2001]; however, this “label bias problem” does not explain why MEMMs perform worse than non-sequential ME, since MEMMs clearly can represent strictly more distributions than ME.

Klein and Manning [2002] also describe an “observation bias problem”, in which MEMMs give too little weight to the history features. However, in this case, relative to the weights assigned by a CRF, MEMM is seems to give *too much* weight to the history features. To verify this, we encouraged the MEMM to downweight the history features by adding noise to the training (not test) data: for each training email/sequence x , we consider each feature-vector $x_i \in x$ in turn, and with probability 0.10, we swap line x_i (together with its label y_i) with some other x_j, y_j chosen uniformly from the sequence. Adding this “sequence noise” almost doubles the error rate for CRFs, but reduces the error rate for MEMMs. (Of course, this type of noise does not affect non-sequential ME.) This experiment supports the hypothesis that MEMM is overweighting history features.

3 Stacked Sequential Learning

3.1 Description

The poor results for MEMM described above can be intuitively explained as a mismatch between the data used to *train* the local models $\Pr(y_i|y_{i-1}, x)$ of the MEMM, and the data used to *test* these models. With noise-free training data, it is *always* the case that a signature line is followed by more signature lines, so the MEMM’s local model tends to weight the history feature y_{i-1} heavily. However, this strong regularity does not hold when the model is applied to test data in the course of executing the Viterbi algorithm, because then the local model is applied to extended examples $(x_i, \hat{y}_{i-1})$, where $\hat{y}_{i-1}$ is a *prediction* made by the local model.

In theory, of course, this training/test mismatch is compensated for by the Viterbi search, which is in turn driven by the confidence estimates produced by the local model. However, if the assumptions of the theory are violated (for instance, if there are high-order interactions not accounted for by the maximum-entropy model), the local model’s confidence estimates may be incorrect.

One way to correct the training/test mismatch is to modify the extended dataset that is used to train the local model so that the true previous class y_{i-1} in an extended instance (x_i, y_{i-1}) is replaced by a *predicted* previous class $\hat{y}_{i-1}$. Below we will outline a way to do this.

Assume that one is given a sample $S = \{(x_t, y_t)\}$ of size m , and a sequential learning algorithm A . Previous work on a meta-learning method called *stacking* [Wolpert, 1992] suggests the following scheme for constructing a sample of $(x, \hat{y})$ pairs in which $\hat{y}$ is a vector of “predicted” class-labels for x . First, partition S into K equal-sized disjoint subsets $S_1, \dots, S_K$, and learn K functions $f_1, \dots, f_K$, where

²We used the implementations of ME, MEMMs, and CRFs provided by Minorthird [Minorthird, 2004], with a limit of 50 optimization iterations. This limit does not substantially change the results.

Stacked Sequential Learning.

Parameters: a history size W_h , a future size W_f , and a cross-validation parameter K .

Learning algorithm: Given a sample $S = \{(\mathbf{x}_t, \mathbf{y}_t)\}$, and a sequential learning algorithm A :

1. Construct a sample of predictions $\hat{\mathbf{y}}_t$ for each $\mathbf{x}_t \in S$ as follows:
 - (a) Split S into K equal-sized disjoint subsets $S_1, \dots, S_K$
 - (b) For $j = 1, \dots, K$, let $f_j = A(S - S_j)$
 - (c) Let $\hat{S} = \{(\mathbf{x}_t, \hat{\mathbf{y}}_t) : \hat{\mathbf{y}}_t = f_j(\mathbf{x}_t) \text{ and } \mathbf{x}_t \in S_j\}$
2. Construct an extended dataset S' of instances $(\mathbf{x}'_t, \mathbf{y}_t)$ by converting each $\mathbf{x}_t$ to $\mathbf{x}'_t$ as follows: $\mathbf{x}'_t = \langle x'_1, \dots, x'_{\ell_t} \rangle$ where $x'_i = (x_i, \hat{y}_{i-W_h}, \dots, \hat{y}_{i+W_f})$ and $\hat{y}_i$ is the i -th component of $\hat{\mathbf{y}}_t$, the label vector paired with $\mathbf{x}_t$ in $\hat{S}$.
3. Return two functions: $f = A(S)$ and $f' = A(S')$.

Inference algorithm: given an instance vector $\mathbf{x}$:

1. Let $\hat{\mathbf{y}} = f(\mathbf{x})$
2. Carry out Step 2 above to produce an extended instance $\mathbf{x}'$ (using $\hat{\mathbf{y}}$ in place of $\mathbf{y}_t$).
3. Return $f'(\mathbf{x}')$.

Table 2: The sequential stacking meta-learning algorithm.

$f_j = A(S - S_j)$. Then, construct the set

$$\hat{S} = \{(\mathbf{x}_t, \hat{\mathbf{y}}_t) : \hat{\mathbf{y}}_t = f_j(\mathbf{x}_t) \text{ and } \mathbf{x}_t \in S_j\}$$

In other words, $\hat{S}$ pairs each $\mathbf{x}_t$ with the $\hat{\mathbf{y}}_t$ associated with performing a K -fold cross-validation on S . The intent of this method is that $\hat{\mathbf{y}}$ is similar to the prediction produced by an f learned by A on a size- m sample that does not include $\mathbf{x}$.

This procedure is the basis of the meta-learning algorithm of Table 2. This method begins with a sample S and a sequential learning method A . In the discussion below we will assume that A is ME, used for sequential data.

Using S , A , and cross-validation techniques, one first pairs with each $\mathbf{x}_t \in S$ the vector $\hat{\mathbf{y}}_t$ associated with performing cross-validation with ME. These predictions are then used to create a dataset S' of extended instances $\mathbf{x}'$, which in the simplest case, are simply vectors composed of instances of the form $(x_i, \hat{y}_{i-1})$, where $\hat{y}_{i-1}$ is the $(i-1)$ -th label in $\hat{\mathbf{y}}$.

The extended examples S' are then used to train a model $f' = A(S')$. If A is the non-sequential maximum-entropy learner, this step is similar to the process of building a “local model” for an MEMM: the difference is that the history features added to x_i are derived not from the true history of x_i , but are (approximations of) the off-sample predictions of an ME classifier.

At inference time, f' must be run on examples that have been extended by adding prediction features $\hat{y}$. To keep the “test” distribution similar to the “training” distribution, f will not be used as the inner loop of a Viterbi or beam-search process; instead, the predictions $\hat{y}$ are produced using a non-sequential maximum-entropy model f that is learned from S . The algorithm of Table 2 simply generalizes this idea from

ME to an arbitrary sequential learner, and from a specific history feature to a parameterized set of features.

In our experiments, we introduced one small but important refinement: each “history feature” $\hat{y}$ added to an extended example is not simply a predicted class, but a numeric value indicating the log-odds of that class. This makes accessible to f' the confidences previously used by the Viterbi search.

3.2 Initial results

We applied stacked sequential learning with ME as the base learner (henceforth s-ME) to the signature-detection dataset. We used $K = 5$, $W_h = 1$, and $W_f = 0$. The s-ME method obtains an error rate of 2.63% on the signature-detection task—less than the baseline ME method (3.20%) but still higher than CRFs (1.17%). However, certain extensions dramatically improve performance.

For s-ME, the only impact of more “history” features is to add new features to the extended instances; hence like MEMMs, s-ME can efficiently handle large histories. On the signature-detection task, increasing the history size to 11 reduces error (slightly) to 2.38%.

For s-ME, the extended instance for x_i can include predicted classes not only of previous instances, but also of “future” instances—instances that follow x_i in the sequence $\mathbf{x}$. We explored different “window sizes” for s-ME, where a “window size” of W means that $W_h = W_f = W$, i.e., the W previous and W following predicted labels are added to each extended instance. A value of $W = 20$ reduces error rates to only 0.71%, a 46% reduction from CRF’s error rate of 1.17%. This improvement is statistically significant.³

Finally, stacked sequential learning can be applied to any learner—in particular, since the extended examples are sequential, it can be applied any sequential learner. We evaluated stacked sequential CRFs (henceforth s-CRFs) with varying window sizes on this problem. A value of $W = 20$ reduces error rates to 0.68%, a statistically significant improvement over CRFs. However, for moderately large window values, there was little performance difference between s-CRF and s-ME.

3.3 Discussion

A graphical view of a MEMMs is shown in Figure 1(a). We use the usual convention in which nodes for known values are shaded. Each node is associated with a maximum-entropy conditional model which defines a probability distribution given its input values.

Figure 1(b) presents a similar graphical view of the classifier learned by sequential stacking with $W_h = 1$ and $W_f = 0$. Inference in this model is done in two stages: first the middle layer is inferred from the bottom later, then the top layer is inferred from the middle layer. The nodes in the middle layer are partly shaded to indicate that their hybrid status—they are considered outputs by the model f , and inputs by the model f' .

³A two-tailed paired t -test rejects with $> 95\%$ confidence the null hypothesis that the difference in error rate between s-ME and CRF on a randomly selected sequence $\mathbf{x}$ has a mean of zero.

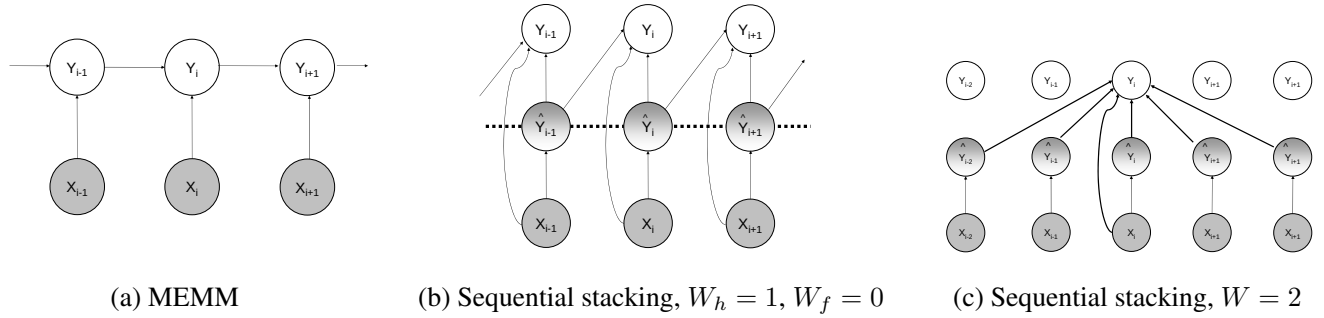


Figure 1: Graphical views of alternative sequential-stacking schemes.

One way to interpret the hybrid layer is as a means of making the inference more robust. If the middle-layer nodes were treated as ordinary unobserved variables, the top-layer conditional model (f') would rely heavily on the confidence assessments of the lower-layer model (f). Forcing f' treat these variables as *observed* quantities allows f' to develop its own model of how the $\hat{y}$ predictions made by f correlate with the actual outputs y . This allows f' to accept or downweight f 's predictions, as appropriate. As suggested by the dotted line in the figure, stacking conceptually creates a “firewall” between f and f' , insulating f' from possible errors in confidence made by f .

Figure 1(c) shows a sequential stacking model with a window of $W_h = W_f = 2$. To simplify the figure, only the edges that eventually lead to the node Y_i are shown.

To conclude our discussion, we note that as described, sequential stacking increases run-time of the base learning method by approximately a constant factor of $K + 2$. (To see this, note sequential stacking requires training $K + 2$ classifiers: the classifiers $f_1, \dots, f_K$ used in cross-validation, and the final classifiers f and f' .) When data is plentiful but training time is limited, it is also possible to simply split the original dataset S into two disjoint halves S_1 and S_2 , and train two classifiers f and f' from S_1 and S'_2 respectively (where S'_2 is S_2 , extended with the predictions produced by f). This scheme leaves training time approximately unchanged for a linear-time base learner, and decreases training time for any base learner that requires superlinear time.

4 Further Experimental Results

4.1 Additional segmentation tasks

We also evaluated non-sequential ME, MEMMs, CRFs, s-ME, and s-CRFs on several other sequential partitioning tasks. For stacking, we used $K = 5$ and a window size of $W_h = W_f = 5$ on all problems. These were the only parameter values explored in this section, and no changes were made to the sequential stacking algorithm, which was developed based on observations made from the signature-detection task only.

One set of tasks involved classifying lines from FAQ documents with labels like “header”, “question”, “answer”, and “trailer”. We used the features adopted by McCallum *et al*

Task	MEMM	ME	CRF	s-ME	s-CRF
A/aigen	53.61	8.02	20.35	6.91	5.78
A/ainn	70.09	6.61	2.14	3.65	1.67
A/aix	13.86	5.02	6.83	4.59	11.79
T/aigen	0.30	2.60	2.39	1.92	0.00
T/ainn	1.36	1.39	0.28	0.00	0.28
T/aix	3.51	1.25	5.26	0.05	4.44
1/video	11.39	12.66	12.66	12.66	13.92
2/video	8.86	8.86	7.59	3.80	7.59
mailsig	31.83	3.47	1.17	1.08	0.77

Table 3: Error rate comparison of different sequential algorithms on a set of nine benchmark tasks.

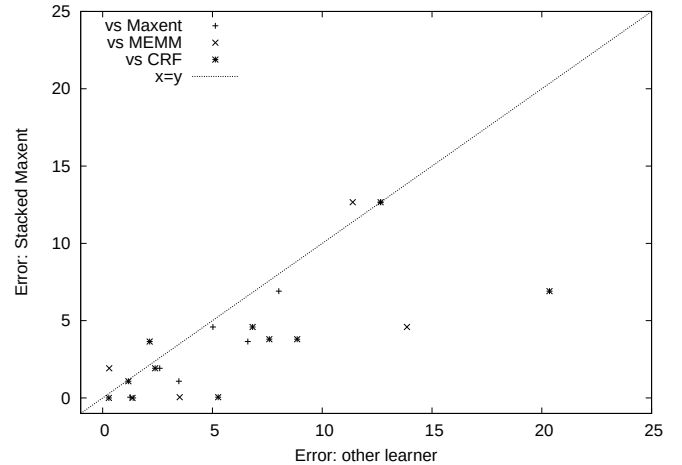


Figure 2: Comparison of the error rates for s-ME with the error rates of ME, MEMM, and CRFs.

[2000] and the three tasks (ai-general, ai-neural-nets, and aix) adopted by Dietterich *et al* [2004]. The data consists of 5-7 long sequences, each sequence corresponding to a single FAQ document; in total, each task contains between 8,965 and 12,757 labeled lines. Our current implementation of sequential stacking only supports binary labels, so we considered the two labels “trailer” (T) and “answer” (A) as separate tasks for each FAQ, leading to a total of six new benchmarks.

Another set of tasks were video segmentation tasks, in which the goal is to take a sequence of video “shots” (a sequence of adjacent frames taken from one camera) and classify them into categories such as “anchor”, “news” and “weather”. This dataset contains 12 sequences, each corresponding to a single video clip. There are a total of 418 shots, and about 700 features, which are produced by applying LDA to a 5x5, 125-bin RGB color histogram of the central frame of the shot. (This data was provided by Yik-Cheung Tam and Ming-yu Chen.) We constructed two separate video partitioning tasks, corresponding to the two most common labels.

All eight of these additional tasks are similar to the signature-detection task in that they contain long runs of identical labels, leading to strong regularities in constructed history features. Error rates for the learning methods on these eight tasks, in addition to the previous signature-detection task, are shown in Table 3. In each case a single train/test split was used to evaluate error rates. The bold-faced entries are the lowest error rate on a row.

We observe that MEMMs suffer extremely high error rates on two of the new tasks (finding “answer” lines for ai-general and ai-neural-nets), suggesting that the “anomalous” behavior shown in signature-detection may not be uncommon, at least in sequential partitioning tasks.

Also, comparing s-ME to ME, we see that s-ME improves the error rate in 8 of 9 tasks, and leaves it unchanged once. Furthermore, s-ME has a lower error rate than CRFs 7 of 9 times, and has the same error rate once. There is only one case in which MEMMs have a lower error rate than s-ME.

Overall, s-ME seems to be preferable to either of three older approaches (ME, MEMMs, and CRFs). This is made somewhat more apparent by the scatter plot of Figure 2. On this plot, each point is placed so the y -axis position is the error of s-ME, and the x -axis position is the error of an earlier learner; thus points below the line $y = x$ are cases where s-ME outperforms another learner. (For readability, the range of the x axis is truncated—it does not include the highest error rates of MEMM.)

Stacking also improves CRF on some problems, but the effect is not as consistent: s-CRF improves the error rate on 5 of 9 tasks, leaves it unchanged twice, and increases the error rate twice. In the table, one of the two stacked learners has the lowest error rate on 8 of the 9 tasks.

Applying a one-tailed sign test, the improvement of s-ME relative to ME is statistically significant ($p > 0.98$), but the improvement of s-CRF relative to CRF is statistically not ($p > 0.92$). The sign test does not consider the amounts by which error rates are changed, however. From the figures and tables, it is clear that error rates are often lowered substantially, and only once raised by more than a very small proportion (for the “A/aix” benchmark with CRFs).

4.2 A sequence classification task

As a final test, we explored one additional non-trivial task: classifying popular songs by emotion. This task was considered after all code development was complete, and hence is a completely prospective test of sequential stacking. A collection of 201 popular songs was annotated by two students on a five-point scale: “very happy”(5), “happy”(4), “neu-

Task	MEMM	ME	CRF	s-ME	s-CRF
music2	28.14	21.40	21.40	18.51	13.50
music5	64.97	67.00	67.00	64.46	63.45

Table 4: Comparison of different sequential algorithms on a music classification task.

tral”(3), “sad”(2) and “very sad”(1). The Pearson’s correlation [Walpole *et al.*, 1998] was used as a measure of inter-annotator agreement. The Pearson’s correlation coefficient ranges from -1 (perfect disagreement) to $+1$ (perfect agreement), and the inter-annotator agreement between the two students was 0.643.

To learn a song classifier, we represented each song as a sequence of 1-second long “frames”, each frame being labeled with the emotion for the song that contains it. We then learned sequence classifiers from these labeled sequences. To classify an unknown song, we used the sequence classifier to label the frames for the song, and finally labeled the song with the most frequent predicted frame label.

The “frames” are produced by extracting certain numerical properties from a waveform representation of the song every 20ms, and then averaging over 1-second intervals. Each 1-second frame has as features the mean and standard deviation of each property. The numerical properties were computed using the Marsyas toolkit [Tzanetakis and Cook, 2000], and are based on the short-time Fourier transform, tonality, and Cepstral coefficients.

The music dataset contains 52,188 frames from the 201 songs, with 130 features for each frame. We looked at two versions of the problem: predicting all of the five labels (music5), and predicting only “happy” versus “sad” labels (music2). Preliminary experiments suggested that large windows were effective, thus we used the following parameters in the experiments: $K = 2$ and a window size of $W_h = W_f = 25$ on music5 problem, and $K = 5$ and $W_h = W_f = 25$ on music2.

The results are summarized in Table 4. Both s-CRF and s-ME outperform their non-stacking counterparts, and s-ME outperforms CRFs. Furthermore, s-CRF improves substantially over unstacked CRFs on the two-class problem, reducing the error rate by more than 27%.

5 Conclusions

Sequential partitioning tasks are sequential classification tasks characterized by long runs of identical labels: examples of these tasks include document analysis, video segmentation, and gene finding. In this paper, we have evaluated the performance of certain well-studied sequential probabilistic learners to sequential partitioning tasks. It was observed that MEMMs sometimes obtain extremely high error rates. Error analysis suggests that this problem is neither due to “label bias” [Lafferty *et al.*, 2001] nor “observation bias” [Klein and Manning, 2002], but to a mismatch between the data used to train the MEMM’s local model, and the data on which the MEMM’s local model is tested. In particular, since MEMMs are trained on “true” labels and tested on “predicted” labels, the strong correlations between adjacent labels associated se-

quential partitioning tasks can be misleading to the MEMM's learning method.

Motivated by these issues, we derived a novel method in which cross-validation is used correct this mismatch. The end result is a meta-learning scheme called *stacked sequential learning*. Sequential stacking is simple to implement, can be applied to virtually any base learner, and imposes a constant overhead in learning time (the constant being the number of cross-validation folds plus two). In experiments on several partitioning tasks, sequential stacking consistently improves the performance of two non-sequential base learners, often dramatically. On our set of benchmark problems, sequential stacking with a maximum-entropy learner as the base learner outperforms CRFs 7 of 9 times, and ties once. Also, on a prospective test conducted on a completely new task, sequential stacking improves the performance of both CRFs and maximum-entropy learners, leading in one case to a 27% reduction in error over conventional CRFs.

One of the more surprising results (for us) is that sequential stacking also often improves performance of conditional random fields, a learner specifically designed for sequential tasks. In a longer version of this paper, we conducted similar with two margin-based base learners: the non-sequential voted perceptron algorithm (VP) [Freund and Schapire, 1998] and a voted-perceptron based training scheme for HMMs proposed by Collins (VPHMMs) [Collins, 2002], with qualitatively similar results.

Some initial experiments on a named entity recognition problem suggest that sequential stacking does not improve performance on non-partitioning problems; however, in future work, we plan to explore this issue with more detailed experimentation.

Acknowledgements

The authors wish to thank many friends and colleagues for input, in particular David McAllister. We are also grateful to Yik-Cheung Tam and Ming-yu Chen for providing the video-segmentation data, and to Chih-yu Chao for the popular music dataset labels. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency, or the Department of Interior-National Business Center.

References

- [Berger *et al.*, 1996] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, 1996.
- [Carvalho and Cohen, 2004] Vitor R. Carvalho and William W. Cohen. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam 2004*, Mountain View, California, 2004.
- [Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2002.
- [Dietterich *et al.*, 2004] Thomas G. Dietterich, Adam Ashenfelder, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21th International Conference on Machine Learning (ICML)*, 2004.
- [Dietterich, 2002] Thomas G. Dietterich. Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer Verlag, New York, 2002.
- [Freund and Schapire, 1998] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learning Theory*, pages 209–217, 1998.
- [Klein and Manning, 2002] Dan Klein and Christopher D. Manning. Conditional structure versus conditional estimation in nlp models. In *Workshop on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.
- [Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.
- [McCallum *et al.*, 2000] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML-2000)*, pages 591–598, Palo Alto, CA, 2000.
- [Minorthird, 2004] Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. <http://minorthird.sourceforge.net>, 2004.
- [Ratnaparkhi, 1999] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, 1999.
- [Toutanova *et al.*, 2003] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, 2003.
- [Tzanetakis and Cook, 2000] George Tzanetakis and Perry Cook. MARSyas: A Framework for Audio Analysis. *Organized Sound*, Cambridge University Press 4(3), 2000.
- [Walpole *et al.*, 1998] Ronald E. Walpole, Raymond H. Myers, Sharon Myers and Sharon L. Myers. Probability and Statistics for Engineers and Scientists. In *Prentice Hall*, 1998.
- [Wolpert, 1992] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

View Learning for Statistical Relational Learning: With an Application to Mammography

Jesse Davis, Elizabeth Burnside, Inês Dutra, David Page,
Raghu Ramakrishnan, Vítor Santos Costa, Jude Shavlik

University of Wisconsin - Madison
1210 West Dayton
Madison, WI 53706, USA
email: jdavis@cs.wisc.edu

Abstract

Statistical relational learning (SRL) constructs probabilistic models from relational databases. A key capability of SRL is the learning of arcs (in the Bayes net sense) connecting entries in different rows of a relational table, or in different tables. Nevertheless, SRL approaches currently are constrained to use the existing database schema. For many database applications, users find it profitable to define alternative “views” of the database, in effect defining new fields or tables. Such new fields or tables can also be highly useful in learning. We provide SRL with the capability of learning new views.

1 Introduction

Statistical Relational Learning (SRL) focuses on algorithms for learning statistical models from relational databases. SRL advances beyond Bayesian network learning and related techniques by handling domains with multiple tables, representing relationships between different rows of the same table, and integrating data from several distinct databases. SRL techniques currently can learn joint probability distributions over the fields of a relational database with multiple tables. Nevertheless, they are constrained to use only the tables and fields already in the database, without modification. Many human users of relational databases find it beneficial to define alternative *views* of a database—further fields or tables that can be computed from existing ones. This paper shows that SRL algorithms also can benefit from the ability to define new views, which can be used for more accurate prediction of important fields in the original database.

We will augment SRL algorithms by adding the ability to learn new fields, intensionally defined in terms of existing fields and intensional background knowledge. In database terminology, these new fields constitute a learned *view* of the database. We use Inductive Logic Programming (ILP) to learn rules which intensionally define the new fields.

We test the approach in the specific application of creating an expert system in mammography. We chose this application for a number of reasons. First, it is an important practical application with sizable data. Second, we have access

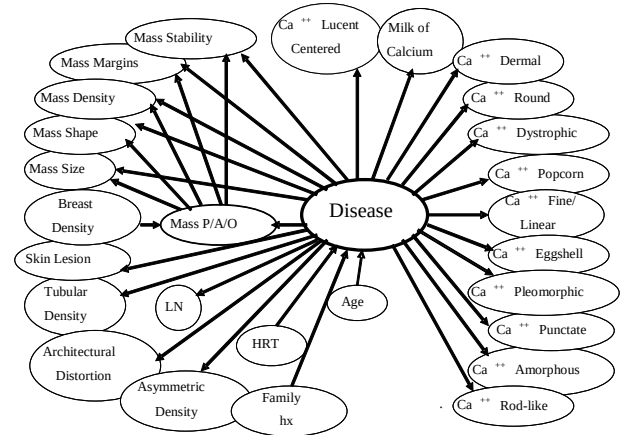


Figure 1: Expert Bayes Net

to an expert developed system. This provides a base reference against which we can evaluate our work. Third, a large proportion of examples are negative. This distribution skew is often found in multi-relational applications. Last, our data consists of a single table. This allows us to compare our techniques against standard propositional learning. In this case, it is sufficient for view learning to extend an existing table with new fields. It should be clear that for other applications the approach can yield additional tables.

2 View Learning for Mammography

Offering breast cancer screening to the ever-increasing number of women over age 40 represents a great challenge. Cost-effective delivery of mammography screening depends on a consistent balance of high sensitivity and high specificity. Recent articles demonstrate that subspecialist, expert mammographers achieve this balance and perform significantly better than general radiologists [24; 1]. General radiologists have higher false positive rates and hence biopsy rates, diminishing the positive predictive value for mammography [24; 1]. Despite the fact that specially trained mammographers detect breast cancer more accurately, there is a longstanding shortage of these individuals [6].

An expert system in mammography has the potential to help the general radiologist approach the effectiveness of a

subspecialty expert, thereby minimizing both false negative and false positive results.

Bayesian networks are probabilistic graphical models that have been applied to the task of breast cancer diagnosis from mammography data [12; 2; 3]. Bayesian networks produce diagnoses with probabilities attached. Because of their graphical nature, they are comprehensible to humans and useful for training. As an example, Figure 1 shows the structure of a Bayesian network developed by a subspecialist, expert mammographer. For each variable (node) in the graph, the Bayes net has a conditional probability table giving the probability distribution over the values that variable can take for each possible setting of its parents. The Bayesian network in Figure 1 achieves accuracies higher than that of other systems and of general radiologists who perform mammograms, and commensurate with the performance of radiologists who specialize in mammography [2].

Figure 2 shows the main table (with some fields omitted for brevity) in a large relational database of mammography abnormalities. Data was collected using the National Mammography Database (NMD) standard established by the American College of Radiology. The NMD was designed to standardize data collection for mammography practices in the United States and is widely used for quality assurance. Figure 2 also presents a hierarchy of the types of learning that might be used for this task. Level 1 and Level 2 are standard types of Bayesian network learning. Level 1 is simply learning the parameters for an expert-supplied network structure. Level 2 involves learning the actual structure of the network in addition to its parameters.

Notice that to predict the probability of malignancy of an abnormality, the Bayes net uses only the record for that abnormality. Nevertheless, data in other rows of the table may also be relevant: radiologists may also consider other abnormalities on the same mammogram or previous mammograms. For example, it may be useful to know that the same mammogram also contains another abnormality, with a particular size and shape; or that the same person had a previous mammogram with certain characteristics. Incorporating data from other rows in the table is not possible with existing Bayesian network learning algorithms and requires statistical relational learning (SRL) techniques, such as probabilistic relational models [8]. Level 3 in Figure 2 shows the state-of-the-art in SRL techniques, illustrating how relevant fields from other rows (or other tables) can be incorporated in the network, using aggregation if necessary. Rather than using only the size of the abnormality under consideration, the new aggregate field allows the Bayes net to also consider the average size of all abnormalities found in the mammogram.

Presently, SRL is limited to using the original view of the database, that is, the original tables and fields, possibly with aggregation. Despite the utility of aggregation, simply considering only the existing fields may be insufficient for accurate prediction of malignancies. Level 4 in Figure 2 shows the key capability that will be introduced and evaluated in this paper: using techniques from rule learning to learn a new view. The new view includes two new features utilized by the Bayes net that cannot be defined simply by aggregation of existing features. The new features are defined by two learned rules

that capture “hidden” concepts central to accurately predicting malignancy, but that are not explicit in the given database tables. One learned rule states that a change in the shape of the abnormality at a location since an earlier mammogram may be indicative of a malignancy. The other says that an *increase* in the average of the sizes of the abnormalities may be indicative of malignancy. Note that both rules require reference to other rows in the table for the given patient, as well as intensional background knowledge to define concepts such as “increases over time.” Neither rule can be captured by standard aggregation of existing fields.

3 View Learning Framework

One can imagine a variety of approaches to perform view learning. Our closing section discusses a number of alternatives, including performing view learning and structure learning at the same time, in the same search. For the present work, we apply existing technology in a new fashion to obtain a view learning capability.

Any relational database can be naturally and simply represented in a subset of first-order logic [21]. Inductive logic programming (ILP) provides algorithms to learn rules, also expressed in logic, from such relational data [15], possibly together with background knowledge expressed as a logic program. ILP systems operate by searching a space of possible logical rules, looking for rules that score well according to some measure of fit to the data. We use ILP to learn rules to predict whether an abnormality is malignant. We treat each rule as an additional binary feature; true if the body, or condition, of the rule is satisfied, and otherwise false. We then run the Bayesian network structure learning algorithm, allowing it to use these new features in addition to the original features. Below is a simple rule, covering 48 positive examples and 123 negative examples:

```
Abnormality A in mammogram M
    may be malignant if:
    A's tissue is not asymmetric,
    M contains another abnormality A2,
    A2's margins are spiculated, and
    A2 has no architectural distortion.
```

This rule can now be used as a field in a new view of the database, and consequently as a new feature in the Bayesian network. The last two lines of the rule refer to other rows of the relational table for abnormalities in the database. Hence this rule encodes information not available to the current version of the Bayesian network.

4 Experiments

The purposes of the experiments we conducted are two-fold. First, we want to determine if using SRL yields an improvement compared to propositional learning. Secondly, we want to evaluate whether using Inductive Logic Programming (ILP) to create features, which embody a new “view” of the database, adds a benefit over current SRL algorithms. We looked at adding two types of relational attributes, aggregate features and horn-clause (ILP) rules. Aggregate features represent summaries of abnormalities found either in a particular mammogram or for a particular patient. We performed

Patient	Abnormality	Date	Mass Shape	...	Mass Size	Location	Benign/Malign
P1	1	5/02	Spic		0.03	RU4	B
P1	2	5/04	Var		0.04	RU4	M
P1	3	5/04	Spic		0.04	LL3	B
...	...	...	...		...	...	...

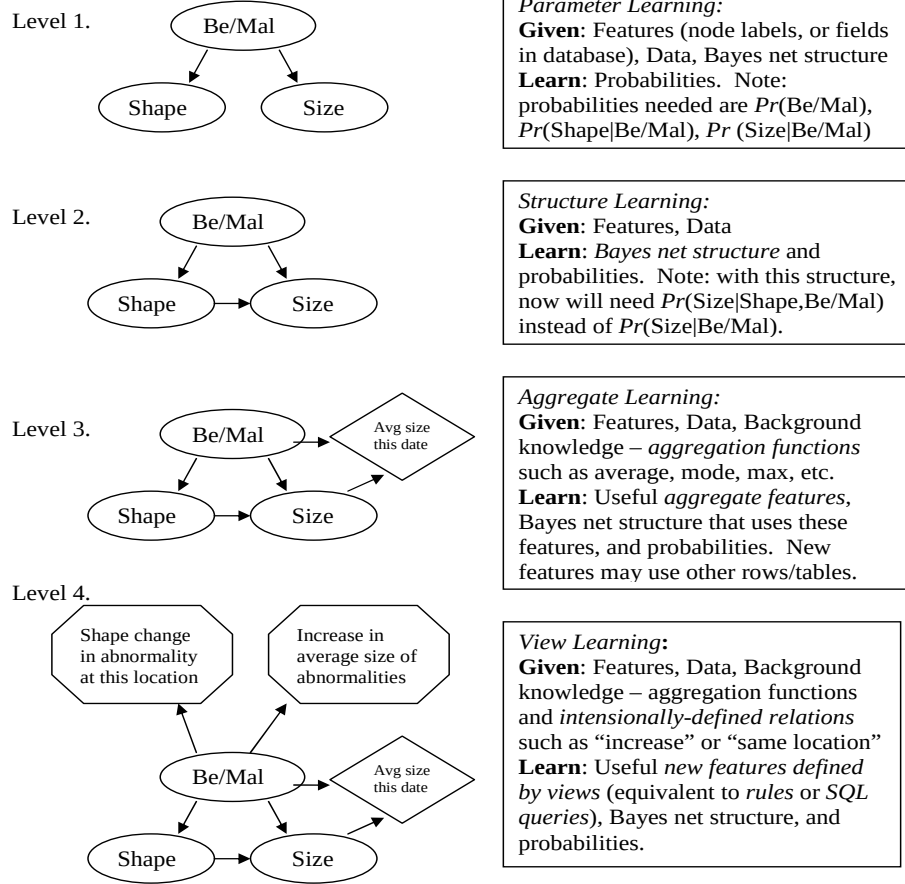


Figure 2: Hierarchy of learning types. Levels 1 and 2 are available through ordinary Bayesian network learning algorithms, Level 3 is available only through state-of-the-art SRL techniques, and Level 4 is described in this paper.

a series of experiments, aimed at discovering if moving up in the hierarchy outlined in Figure 2 would improve performance. First, we tried to learn a structure with just the original attributes which performed better than the expert structure. This corresponds to Level 2 learning. Next, we added aggregate features to our network. Finally, we created new features using ILP. We investigated adding the features proposed by the ILP system as well as the aggregate to the network.

We experimented with a number of structure learning algorithms for the Bayesian Networks, including Naïve Bayes, Tree Augmented Naïve Bayes [7], and the sparse candidate algorithm [9]. However, we obtained best results with the TAN algorithm in all experiments, so we will focus our discussion on TAN. In a TAN network, each attribute can have at most one other parent in addition to the class vari-

able. The TAN model can be constructed in polynomial time with a guarantee that the model maximizes the Log Likelihood of the network structure given the dataset [10; 7].

4.1 Methodology

The dataset contains 435 malignant abnormalities and 65365 benign abnormalities. To evaluate and compare these approaches, we used stratified 10-fold cross-validation. We randomly divided the abnormalities into 10 roughly equal-sized sets, each with approximately one-tenth of the malignant abnormalities and one-tenth of the benign abnormalities. When evaluating just the structure learning and aggregation, 9 folds were used for the training set. When performing aggregation, we used binning to discretize the created features. We took care to only use the examples in the train set to determine the cut bin widths. When performing “view learning”, we had

two steps in the learning process. In the first part, 4 folds of data were used to learn the ILP rules. Afterwards, the remaining 5 folds were used to learn the Bayes net structure and parameters.

When using cross-validation on a relational database, there exists one major methodological pitfall. Some of the cases may be related. For example, we may have multiple abnormalities for a single patient. Because these abnormalities are related (same patient), having some of these in the training set and others in the test set may cause us to perform better on those test cases than we would expect to perform on cases for other patients. To avoid such “leakage” of information into a training set, we ensured that all abnormalities associated with a particular patient are placed into the same fold for cross-validation. Another potential pitfall is that we may learn a rule that predicts an abnormality to be malignant based on properties of abnormalities in *later* mammograms. We never predict the status of an abnormality at a given date based on findings recorded with later dates.

We present the results of our first experiment using both ROC and precision recall curves. Because of our skewed class distribution, or large number of benign cases, we prefer precision-recall curves over ROC curves because they better show the number of “false alarms,” or unnecessary biopsies. Therefore, we use precision-recall curves for the remainder of the results. Here, precision is the percentage of abnormalities that we classified as malignant that are truly cancerous. Recall is the percentage of malignant abnormalities that were correctly classified. To generate the curves, we pooled the results over all ten folds by treating each prediction as if it had been generated from the same model. We sorted the estimates and used all possible split points to create the graphs.

4.2 Results

The relational database containing the mammography data contains one row for each abnormality in a mammogram. Fields in this relational table include all those shown in the Bayesian network of Figure 1. Therefore it is straightforward to use existing Bayesian network structure learning algorithms to learn a possibly improved structure for the Bayesian network. We compared the performance of the best learned networks against the expert defined structure shown in Figure 1. We estimated the parameters of the expert structure from the dataset using maximum likelihood estimates with Laplace correction. Figure 3 shows the ROC curve for these experiments, and Figure 4 shows the Precision-Recall curves. Figure 7 shows the area under the precision-recall curve for the expert network (L1) and with learned structure (L2). We only consider recalls above 50%, as for this application radiologists would be required to perform at least at this level. We further use the paired t-test to compare the areas under the curve for every fold. We found the difference to be statistically significant with a 99% level of confidence.

With the help of a radiologist, we selected the numeric and ordered features in the database and computed aggregates for each of these features. We determined that 27 of the 36 attributes were suitable for aggregation. We computed aggregates on both the patient and the mammogram level. On the patient level, we looked at all of the abnormalities for a spe-

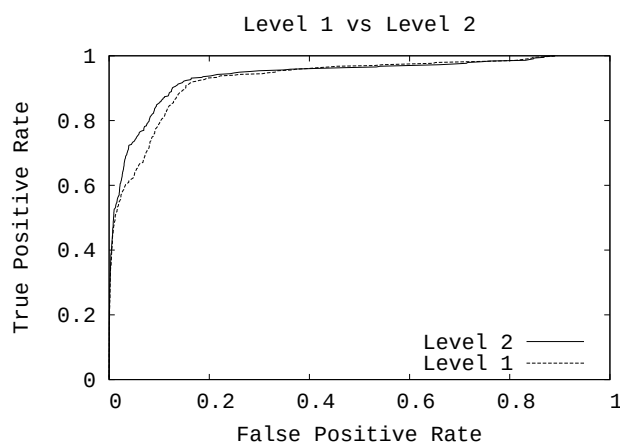


Figure 3: ROC Curves for Structure Learning. (Level 2)

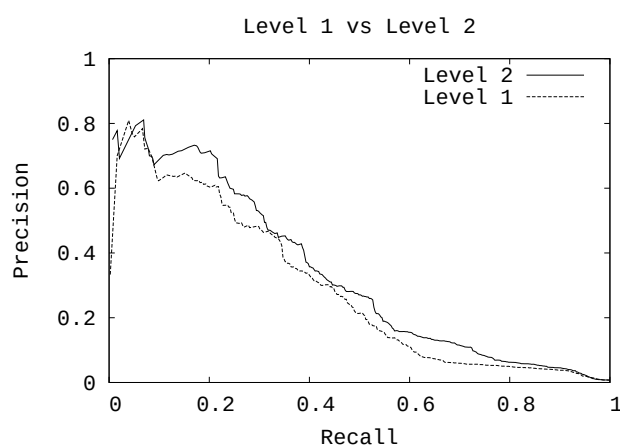


Figure 4: Precision Recall Curves for Structure Learning. (Level 2)

cific patient. On the mammogram level, we only considered the abnormalities present on that specific mammogram. To discretize the averages, we divided each range into three bins. For binary features we had predefined bin sizes, while for the other features we attempted to get equal numbers of abnormalities in each bin. For aggregation functions we used maximum and average. The aggregation introduced 108 new features. For the interested reader, the following paragraph presents further details of our aggregation process.

We used a three step process to construct aggregate features. First, we chose a field to aggregate. Second, we selected an aggregation function. Third, we needed to decide over which rows to aggregate the feature, that is, which keys or links to follow. This is known as a slot chain in PRM terminology. In our database, two such links exist. The patient id field allows access to all the abnormalities for a given patient, providing aggregation on the patient level. The second key is the combination of patient id and mammogram date, which returns all abnormalities for a patient on a specific mammogram, providing aggregation on the mammogram level. To demonstrate this process we will work through an example of

Patient	Abnormality	Date	Mass Shape	...	Mass Size	Location	Average Patient Mass Size	Average Mammogram Mass Size	Be/Mal
P1	1	5/02	Spic	...	0.03	RU4	0.0367	0.03	B
P1	2	5/04	Var	...	0.04	RU4	0.0367	0.04	M
P1	3	5/04	Spic	...	0.04	LL4	0.0367	0.04	B
...	...	...	...	...	...	...	...	...	...

Figure 5: Database after Aggregation on Mass Size Field

computing an aggregate feature for patient 1 in the database given in Figure 2. We will aggregate on the Mass Size field and use average as the aggregation function. Patient 1 has three abnormalities, one from a mammogram in May 2002 and two from a mammogram in May 2004. To calculate the aggregate on the patient level, we would average the size for all three abnormalities, which is .0367. To find the aggregate on the mammogram level for patient 1, he have to perform two separate computations. First, we follow the link P1 and 5/02, which yields abnormality 1. The average for this key mammogram is simply .03. Second, we follow the link P1 and 5/04, which yields abnormalities 2 and 3. The average for these abnormalities is .04. Figure 5 shows the database following construction of these aggregate features.

Next, we tested whether useful new fields could be computed by rule learning. Specifically, we used the ILP system Aleph [26] to learn rules predictive of malignancy. Several thousand distinct rules were learned for each fold, with each rule covering many more malignant cases than (incorrectly covering) benign cases. In order to obtain a varied set of rules, we ran Aleph using every positive example in each fold as a seed for the search. We avoid the rule overfitting found by other authors [18] by doing breadth-first search for rules and by having a minimal limit on coverage. Each seed generated anywhere from zero to tens of thousands of rules. We post processed the rules using a greedy algorithm, where we selected the best scoring rule that covers new examples first. For each fold, the 50 best clauses were selected based on 3 criteria: (1) they needed to be multi-relational; (2) they needed to be distinct; (3) they needed to cover a significant number of malignant cases. The resulting views were added as new features to the database. Figure 6 includes a comparison of all levels of learning.

We can observe very significant improvements when adding multi-relational features. Both rules and aggregates achieved better performance. Aggregates do better for higher recalls, while rules do better for medium recalls. We believe this is because ILP rules are more accurate than the other features, but have limited coverage.

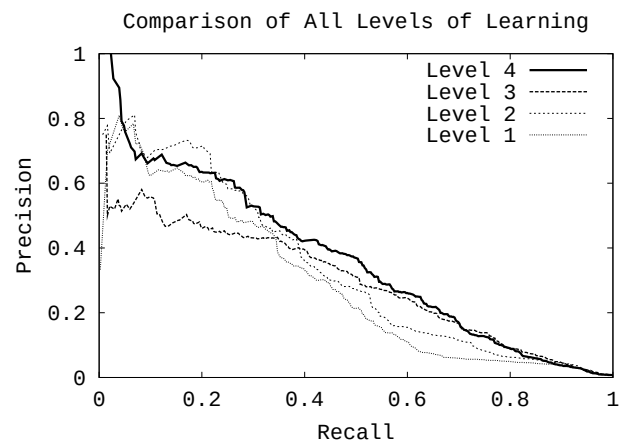


Figure 6: Precision Recall Curves for Each Level of Learning

Level 4 performs as well as aggregates for high recalls, and close to ILP for medium recalls. According to the paired t-test the improvement of Level 4 over Level 2 is significant, using the area under the curve metric, at the 99% level. Meanwhile, Level 3 presents an improvement over Level 2, using the area under the curve metric, at the 97% confidence level.

Levels 1 and 2 correspond to standard propositional learning whereas levels 3 and 4 incorporate relational information. In this task, considering relational information is crucial for improving performance. Furthermore, the process of generating the views in Level 4 has been useful to the radiologist as it has identified novel correlations between attributes.

5 Related Work

Research in SRL has advanced along two main lines: methods that allow graphical models to represent relations, and frameworks that extend logic to handle probabilities. Along the first line, probabilistic relational models, or PRMs, introduced by Friedman, Getoor, Koller and Pfeffer, represent one of the first attempts to learn the structure of graphical mod-

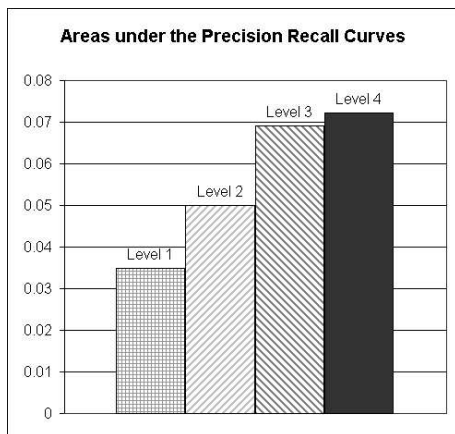


Figure 7: Area Under the Curve For Recalls Above 50%

els while incorporating relational information[8]. Recently Heckerman, Meek and Koller have discussed extensions to PRMs and compared them to other graphical models[11]. A statistical learning algorithm for probabilistic logic representations was first given by Sato [23] and later, Cussens [5] proposed a more general algorithm to handle log linear models. Additionally, Muggleton [16] has provided learning algorithms for stochastic logic programs. The structure of the logic program is learned using ILP techniques, while the parameters are learned using an algorithm scaled up from that used for stochastic context-free grammars.

Newer representations garnering arguably the most attention are Bayesian logic programs [13] (BLPs), constraint logic programming with Bayes net constraints, or CLP($\mathcal{BN}$) [4], and Markov Logic Networks (MLNs) [22]. Markov Logic Networks are most similar to our approach. Nodes of MLNs are the ground instances of the literals in the rule, and the arcs correspond to the rules. One major difference is that, in our approach, nodes are the rules themselves. Although we cannot work at the same level of detail, our approach makes it straightforward to combine logical rules with other features, and we now can take full advantage of propositional learning algorithms.

The present work builds upon previous work on using ILP for feature construction. Such work treats ILP-constructed rules as Boolean features, re-represents each example as a feature vector, and then uses a feature-vector learner to produce a final classifier. To our knowledge, Pompe and Kononenko [19] were the first to apply Naïve Bayes to combine clauses. Other work in this category was by Srinivasan and King [25], who use rules as extra features for the task of predicting biological activities of molecules from their atom-and-bond structures. Popescul et al. [20] use $k - means$ to derive cluster relations, which are then combined with the original features through structural regression. In a different vein, Relational Decision Trees [17] use aggregation to provide extra features on a multi-relational setting, and are close to our Level 3 setting. Knobbe et al. [14] proposed numeric aggregates in combination with logic-based feature construction for single attributes. Perlich and Provost discuss several

approaches for attribute construction using aggregates over multi-relational features [18]. The authors also propose a hierarchy of levels of learning: feature vectors, independent attributes on a table, multidimensional aggregation on a table, and aggregation across tables. Some of these techniques in their hierarchy could be applied to perform view learning in SRL.

6 Conclusions and Future Work

We presented a method for statistical relational learning which integrates learning from attributes, aggregates, and rules. Our example application shows benefits from the several levels of learning we proposed. Level 2, structure learning, clearly outperforms the expert structure. We further show that multi-relational techniques can achieve very significant improvements, even on a single table domain, and that the most consistent improvement is obtained by using Level 4, both aggregates and new views.

We believe that further improvements are possible. It makes sense to include aggregates in the background knowledge for rule generation. Alternatively, one can extend rules with aggregation operators, as proposed in recent work by Vens et al. [27]. We have found the rule selection problem to be non-trivial. Our greedy algorithm often generates too similar rules, and is not guaranteed to maximize coverage. We would like to approach this problem as an optimization problem weighing coverage, diversity, and accuracy.

Our approach of using ILP to learn new features for an existing table merely scratches the surface of the potential for view learning. A more ambitious approach would be to more closely integrate structure learning and view learning. A search could be performed in which each “move” in the search space is either to modify the probabilistic model or to refine the intensional definition of some field in the new view. Going further still, one might learn an intensional definition for an entirely new table. As a concrete example, for mammography one could learn rules defining a binary predicate that identifies “similar” abnormalities. Because such a predicate would represent a many-to-many relationship among abnormalities, a new table would be required.

7 Acknowledgments

Support for this research was partially provided by U.S. Air Force grant F30602-01-2-0571. Elizabeth Burnside is supported by a General Electric Research in Radiology Academic Fellowship. Inês Dutra and Vítor Santos Costa are on leave from Federal University of Rio de Janeiro, Brazil. Vítor Santos Costa was partially supported by the Fundação para a Ciência e Tecnologia. We would like to thank Rich Maclin, Jill Davis and Allison Holloway for reading over drafts of this paper. We would also like to thank the referees for their insightful comments.

References

- [1] M.L. Brown, F. Houn, E.A. Sickles, and L.G. Kessler. Screening mammography in community practice: positive predictive value of abnormal findings and yield of

- follow-up diagnostic procedures. *AJR Am J Roentgenol*, 165:1373–1377, 1995.
- [2] E.S. Burnside, D.L. Rubin, and R.D. Shachter. A Bayesian network for screening mammography. In *AMIA*, pages 106–110, 2000.
 - [3] E.S. Burnside, D.L. Rubin, and R.D. Shachter. Using a Bayesian network to predict the probability and type of breast cancer represented by microcalcifications on mammography. *Medinfo*, 2004:13–17, 2004.
 - [4] V. Santos Costa, D. Page, M. Qazi, and J. Cussens. CLP($\mathcal{BN}$): Constraint logic programming for probabilistic knowledge. In *UAI-03*, pages 517–524, Aca-pulco, 2003.
 - [5] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
 - [6] G. Ecklund. Shortage of qualified breast imagers could lead to crisis. *Diagn Imaging*, 22:31–33, 2000.
 - [7] Nir Friedman, David Geiger, and Moises Goldszmidt. Bayesian networks classifiers. *Machine Learning*, 29:131–163, 1997.
 - [8] Nir Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. Stockholm, Sweden, 1999.
 - [9] Nir Friedman, I. Nachman, and D. Pe’er. Learning Bayesian Network Structure from Massive Datasets: The “Sparse Candidate” Algorithm. In *UAI-99*, pages 206–215, San Francisco, CA, 1999. Morgan Kaufmann Publishers.
 - [10] Dan Geiger. An entropy-based learning algorithm of Bayesian conditional trees. In *UAI-92*, pages 92–97, San Mateo, CA, 1992. Morgan Kaufmann Publishers.
 - [11] D Heckerman, C Meek, and D Koller. Probabilistic Entity-Relationship Models, PRMs, and Plate Models, Technical Report MSR-TR-2004-30, Microsoft Research. Technical report, Microsoft Research, 2004.
 - [12] C.E. Kahn Jr, L.M. Roberts, K.A. Shaffer, and P. Haddawy. Construction of a Bayesian network for mam-mographic diagnosis of breast cancer. *Comput Biol Med.*, 27:19–29, 1997.
 - [13] K. Kersting and L. De Raedt. Basic principles of learning Bayesian logic programs. Technical report, Institute for Computer Science, University of Freiburg, Germany, 2002.
 - [14] Arno J. Knobbe, Marc de Haas, and Arno Siebes. Propositionalisation and aggregates. In *PKDD01*, pages 277–288, 2001.
 - [15] S.H. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8:295–318, 1991.
 - [16] S.H. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 4(041), 2000.
 - [17] Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *KDD ’03*, pages 625–630. ACM Press, 2003.
 - [18] Claudia Perlich and Foster Provost. Aggregation-based feature invention and relational concept classes. In *KDD ’03*, pages 167–176, 2003.
 - [19] U. Pompe and I. Kononenko. Naive Bayesian classifier within ILP-R. In L. De Raedt, editor, *ILP95*, pages 417–436, 1995.
 - [20] Alexandrin Popescul, Lyle H. Ungar, Steve Lawrence, and David M. Pennock. Statistical relational learning for document mining. In *ICDM03*, pages 275–282, 2003.
 - [21] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw Hill, 2000.
 - [22] Matt Richardson and Pedro Domingos. Markov logic networks. <http://www.cs.washington.edu/homes/pedrod/kbmn.pdf>, 2004.
 - [23] T. Sato. A statistical learning method for logic programs with distributional semantics. In L. Sterling, editor, *Proceedings of the Twelfth International conference on logic programming*, pages 715–729, Cambridge, Massachusetts, 1995. MIT Press.
 - [24] E.A. Sickles, D.E. Wolverton, and K.E. Dee. Performance parameters for screening and diagnostic mam-mography: specialist and general radiologists. *Radiology*, 224:861–869, 2002.
 - [25] A. Srinivasan and R. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In *ILP97*, pages 89–104, 1997.
 - [26] Ashwin Srinivasan. *The Aleph Manual*, 2001.
 - [27] Celine Vens, Anneleen Van Assche, Hendrik Blockeel, and Sašo Džeroski. First order random forests with complex aggregates. In *ILP*, pages 323–340, 2004.

Learning Web Page Scores by Error Back-Propagation

Michelangelo Diligenti, Marco Gori, Marco Maggini

Dipartimento di Ingegneria dell'Informazione

Università di Siena

Via Roma 56, I-53100 Siena (Italy)

{michi, marco, maggini}@dii.unisi.it

Abstract

In this paper we present a novel algorithm to learn a score distribution over the nodes of a labeled graph (directed or undirected). Markov Chain theory is used to define the model of a random walker that converges to a score distribution which depends both on the graph connectivity and on the node labels. A supervised learning task is defined on the given graph by assigning a target score for some nodes and a training algorithm based on error back-propagation through the graph is devised to learn the model parameters. The trained model can assign scores to the graph nodes generalizing the criteria provided by the supervisor in the examples. The proposed algorithm has been applied to learn a ranking function for Web pages. The experimental results show the effectiveness of the proposed technique in reorganizing the rank accordingly to the examples provided in the training set.

1 Introduction

In this paper we propose an algorithm able to learn a generic distribution of values over the nodes of a graph with symbolic labels. In many applications, the input data is naturally represented by a labeled graph. However, the actual lack of adequate algorithms to directly process this structured data commonly forces to convert the input patterns to vectors of real numbers, losing significant information. In particular, “Web like” data structures are common in many pattern recognition tasks and possible applications range from object recognition in segmented images to molecular analysis in bio-informatics, and so on.

Like in the PageRank algorithm [Page *et al.*, 1998], the score of a node is assumed to correspond to the probability that a random walker will visit that node at any given time. The score distribution computed by the random walker depends both on the connectivity graph and the labels assigned to each node. Markov Chain theory allows to define a set of constraints on the parameters of the random walker in order to guarantee some desired features like convergence and independence of the final distribution on the initial state. A learning algorithm which is based on error back-propagation through the graph is used to optimize the parameters of the

random walker minimizing the distance of the scores assigned by the walker from the target values provided by a supervisor for some nodes in the graph.

Previous work in the same field achieved very limited results. For example, in [Kondor and Lafferty, 2002] the authors present an algorithm that processes only non labeled graphs, while the technique presented in [Chang *et al.*, 2000] is very application specific and features very limited generalization capabilities. In [Tsoi *et al.*, 2003] a method to modify PageRank scores computed over a collection of pages on the basis of a supervisor’s feedback is proposed. This algorithm has limited approximation capabilities, since it controls only a term of the PageRank equation and it does not optimize how the scores flow through the connectivity graph.

The algorithm presented in this paper allows to learn target scores assigned as real numbers, and this makes it more general than the one proposed in [Diligenti *et al.*, 2003] which allows only to specify a set of nodes whose scores should be increased or decreased.

The paper is organized as follows. In the next section the web surfer model is introduced. Then, in section 3 the learning algorithm based on error back-propagation is described, while the experimental results are presented in section 4. Finally, in section 5 the conclusions are drawn.

2 Random Walks

We consider the model of a graph walker that can perform one out of two atomic actions while visiting the Web graph: jumping to a node of the graph (action J) and following a hyperlink from the current node (action l).

In general, the action taken by the surfer will depend on the label and outlinks from the node where it is located. We can model the user’s behavior by a set of probabilities which depend on the current node: $x(l|q)$ is the probability of following one hyperlink from node q , and $x(J|q) = 1 - x(l|q)$ is the probability of jumping from node q .

The two actions need to specify their targets: $x(p|q, J)$ is the probability of jumping from node q to node p , and $x(p|q, l)$ is the probability of selecting a hyperlink from node q to node p . $x(p|q, l)$ is not null only for the nodes p linked directly by node q , i.e. $p \in ch(q)$, being $ch(q)$ the set of the children of node q in the graph G . These sets of values must satisfy the probability normalization constraints $\forall q \in G \sum_{p \in G} x(p|q, J) = 1$ and $\sum_{p \in ch(q)} x(p|q, l) = 1$.

The model considers a temporal sequence of actions performed by the surfer and it can be used to compute the probability $x_p(t)$ that the surfer is located in node p at time t . The probability distribution on the nodes of the Web is updated by taking into account the actions at time $t + 1$ using the following equation

$$x_p(t+1) = \sum_{q \in \mathbf{G}} x(p|q, J) \cdot x(J|q) \cdot x_q(t) + \sum_{q \in pa(p)} x(p|q, l) \cdot x(l|q) \cdot x_q(t), \quad (1)$$

where $pa(p)$ is the set of the parents of node p . The score x_p of a node p is computed using the stationary distribution of the Markov Chain defined by equation (1) (i.e. $x_p = \lim_{t \rightarrow \infty} x_p(t)$). The probability distribution over all the nodes at time t is represented by the vector $\mathbf{x}(t) = [x_1(t), \dots, x_N(t)]'$, being N the total number of nodes.

An interesting property of this model is that, under the assumption that $x(J|q) \neq 0$ and $x(p|q, J) \neq 0$, $\forall p, q \in \mathbf{G}$, the stationary distribution $\mathbf{x}^*$ does not depend on the initial state vector $\mathbf{x}(0)$ (see [Diligenti *et al.*, 2002]). For example this is true for the PageRank equation for which $x(J|q) = 1 - d$, being $0 < d < 1$, and $x(p|q, J) = \frac{1}{N}$.

The random walker model of equation (1) is a simplified version of the model proposed in [Diligenti *et al.*, 2002], where two additional actions are considered by allowing the surfer to follow backlinks or to remain in the same node. The learning algorithm proposed in the next section can be easily extended to this more general model.

3 Learning the score distribution

The random surfer assigns the scores according to the probabilities $x(p|q, J)$, $x(J|q)$, $x(p|q, l)$, and $x(l|q)$. In most of the approaches proposed in the literature, these parameters are predefined or computed from some features describing the node p and/or the node q . For example, setting the parameters to $x(l|q) = d$, $x(J|q) = 1 - d$, $x(p|q, J) = \frac{1}{N}$, and $x(p|q, l) = \frac{1}{ch(q)}$, we obtain the popular PageRank random surfer [Page *et al.*, 1998].

In the general case, the model has a number of parameters quadratic in the number of nodes and it is infeasible to estimate these parameters without any assumption to reduce their number. In particular, we assume that the labels of nodes can assume n distinct values $C = \{c_1, \dots, c_n\}$ and that the surfer's behavior for any node p is dependent only on its label $c^p \in C$ and the labels of the nodes pointed by p . Therefore, the parameters of the model are rewritten as:

$$\begin{aligned} x(p|q, l) &= \frac{x(c^p|c^q, l)}{\sum_{k \in ch(q)} x(c^k|c^q, l)} \\ x(p|q, J) &= \frac{x(c^p|c^q, J)}{|c^p|} \\ x(l|p) &= x(l|c^p) \quad x(J|p) = x(J|c^p) \end{aligned}$$

where $c^q \in C$ is the label of node q and $|c^p|$ is the number of nodes with label c^p . The parameter $x(c^p|c^q, l)$ represents the

tendency of the surfer to follow a hyperlink from a node with label c^q to a node with label c^p , the parameter $x(c^p|c^q, J)$ is the probability of the surfer to jump from a node with label c^q to a node with label c^p and $x(l|c_i) = 1 - x(J|c_i)$ represents the probability that the surfer decides to follow a link out of a node with label c_i . In the following, we indicate as $\mathcal{P}$ the set of model parameters. $\mathcal{T}$, $\mathcal{J}$, $\mathcal{B}$ indicate *transition*, *jump* and *behavior* parameters, respectively. In particular, it holds that: $\mathcal{P} = \{\mathcal{T}, \mathcal{J}, \mathcal{B}\}$, $\mathcal{T} = \{x(c_i|c_j, l) \mid i, j = 1, \dots, n\}$, $\mathcal{J} = \{x(c_i|c_j, J) \mid i, j = 1, \dots, n\}$, and $\mathcal{B} = \{x(l|c_i) \mid i = 1, \dots, n\}$. Under these assumptions, the random surfer equation becomes,

$$x_p(t+1) = \sum_{q \in \mathbf{G}} \frac{x(c^p|c^q, J)}{|c^p|} \cdot x(J|c^q) \cdot x_q(t) + \sum_{q \in pa(p)} \frac{x(c^p|c^q, l)}{\sum_{k \in ch(q)} x(c^k|c^q, l)} \cdot x(l|c^q) \cdot x_q(t), \quad (2)$$

Typically, the values $x(c^p|c^q, l)$ are initialized to 1, such that all links are equally likely to be followed. Because of the model assumptions, the stronger is the tendency of the surfer to move to a node with a specific label, the higher are the scores associated to the nodes with that label. These assumptions reduce the number of free parameters to $2 \cdot (n^2 - n) + n = 2 \cdot n^2 - n$, being n the number of distinct labels.

3.1 Learning the parameters

We assume that a supervisor provides a set of examples $\mathcal{E}$ in the graph $\mathbf{G}$, where each example $s \in \mathcal{E}$ is a node associated to a target score x_s^d .

Consider the following cost function for the scores computed by a surfer model using the set of parameters $\mathcal{P}$,

$$E^{\mathcal{P}} = \frac{1}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} e_s^{\mathcal{P}} = \frac{1}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} \frac{1}{2} \cdot (x_s^{\mathcal{P}} - x_s^d)^2, \quad (3)$$

where $e_s^{\mathcal{P}}$ is called instant error for the node s . Such error takes into account the distance between the target score and the value returned by the ranking function.

We want to minimize the cost function (3) by gradient descent with respect to a generic parameter $w \in \mathcal{P}$, i.e.,

$$w' = w - \eta \frac{\partial E^{\mathcal{P}}}{\partial w} = w - \frac{\eta}{|\mathcal{E}|} \sum_{s \in \mathcal{E}} \frac{\partial e_s^{\mathcal{P}}}{\partial w}. \quad (4)$$

Using a procedure similar to the *back-propagation through time* algorithm (BPTT) [Haykin, 1994], commonly used in training recurrent neural networks, for each node s in the training set we unfold the graph starting from s at time T and moving backward in time through the path followed by the random surfer. The graph unfolding is basically a backward breadth-first visit of the graph, starting from the target node at time T , where each new unfolded level corresponds to moving backward of one time step. Each unfolded node is associated to the variable $x_q(T-t)$ where q is the corresponding graph node, t is the number of backward steps from the target node. Similarly, the parameters of the model are split into an independent set of parameters corresponding to the

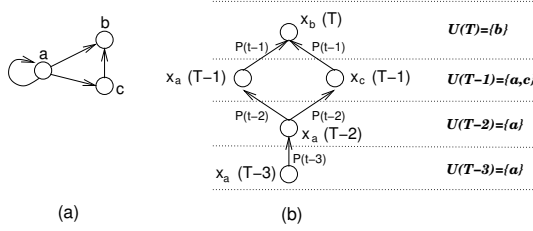


Figure 1: (a) A graph composed by three nodes. (b) The first four levels of the unfolding for the graph, starting from node b at time T . $\mathcal{U}(t)$ indicates the set of nodes at the level t of the unfolding (corresponding to the time step t). In the example, $\mathcal{U}(T) = \{b\}$, $\mathcal{U}(T-1) = \{a, c\}$, $\mathcal{U}(T-2) = \{a\}$ and $\mathcal{U}(T-3) = \{a\}$.

different time steps. We indicate with $\mathcal{P}(t)$ the model parameters at time t and with $w(t) \in \mathcal{P}(t)$ the value of any given parameter at time t . An infinite unfolding is perfectly equivalent to the original graph with respect to the computation of the score $x_s^{\mathcal{P}}(T)$ for a target node s . However, the unfolding takes into account only the score propagation due to the forward links, whereas the contribution of jumps (coming from any node in the graph) is neglected.

In the following, we indicate as $\mathcal{U}_s(t)$ the set of nodes contained in the unfolding centered on node s for the time step t . See figure 1 for an example of graph unfolding.

Thanks to the parameter decoupling in the unfolding, a parameter $w(t)$ influences only the scores at the following time step. Thus, the derivative of the instant error with respect to a parameter $w(t)$ can be written as,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial w(t)} = \sum_{p \in \mathcal{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}. \quad (5)$$

Assuming that the parameters of the surfer are *stationary* (i.e. they do not depend on time: $w(t) = w \forall t$), the derivative with respect to a single parameter can be obtained summing all the contributions for the single time steps,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial w} = \sum_{t=-\infty}^{T-1} \sum_{p \in \mathcal{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}. \quad (6)$$

The term $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t+1)}$ in equation (6) can not be directly computed, unless when $t = T-1, s = p$. In this latter case, it holds that,

$$\frac{\partial e_s^{\mathcal{P}}}{\partial x_s^{\mathcal{P}}(T)} = x_s^{\mathcal{P}}(T) - x_s^d. \quad (7)$$

Let $\delta_p(t)$ indicate $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$. If p is not a target node, then $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$ can be rewritten as,

$$\begin{aligned} \frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)} &= \delta_p(t) = \sum_{z \in \mathcal{G}} \frac{\partial e_s^{\mathcal{P}}}{\partial x_z^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_z^{\mathcal{P}}(t+1)}{\partial x_p^{\mathcal{P}}(t)} \approx \\ &\approx \sum_{z \in ch(p)} \frac{\partial e_s^{\mathcal{P}}}{\partial x_z^{\mathcal{P}}(t+1)} \cdot \frac{\partial x_z^{\mathcal{P}}(t+1)}{\partial x_p^{\mathcal{P}}(t)} = \end{aligned} \quad (8)$$

$$\begin{aligned} &= \sum_{z \in ch(p)} \delta_z(t+1) \cdot \frac{x(c^z | c^p, l)(t)}{\sum_{k \in ch(p)} x(c^k | c^p, l)(t)} \cdot \\ &\quad \cdot x(l | c^p)(t) = \\ &= \sum_{z \in ch(p)} \delta_z(t+1) \cdot P(p, t \rightarrow z, t+1) = \\ &= (x_s^{\mathcal{P}} - x_s^d) \cdot P(p, t \rightarrow s, T), \end{aligned}$$

where $ch(p)$ is the set of children of node p in the unfolding and $P(p, t_1 \rightarrow z, t_2)$ indicates the probability of the surfer to follow a path in the unfolding connecting the node p at time t_1 to the node z at time t_2 . Only the influence of transition parameters on the score of node p was taken into account in the propagation of the delta values.

Neglecting the influence of the jump parameters (i.e. taking into account only the influence due to link following) is an accurate approximation when the probability of following a link is much higher than the probability of jumping (like it is commonly assumed in PageRank). However, the experimental results show that such an approximation provides good results in the general case.

A generic node p at time t has influence over a supervised node s at time T proportional to the probability $P(p, t \rightarrow s, T)$ that the surfer will follow the path from the node p at time t to the central node s at time T . An implication of this result is that farther nodes will have less influence, given the exponential decrease of the probability of following a path with its length. In particular, a node p influences $\frac{\partial e_s^{\mathcal{P}}}{\partial x_p^{\mathcal{P}}(t)}$ only if $P(p, t \rightarrow s, T) \neq 0$. Since $\mathcal{U}_s(T-t)$ is the set of nodes from which it is possible to reach node s at time T starting at time t , only for the nodes in this set it holds that $P(p, t \rightarrow s, T) \neq 0$. Thus, in equation (8) not all the score variables at time $t+1$ must be considered in the application of the chain rule. Inserting equation (8) into (6), and limiting the chain rule only to the nodes with a non-zero influence, yields,

$$\begin{aligned} \frac{\partial e_s^{\mathcal{P}}}{\partial w} &= (x_s^{\mathcal{P}} - x_s^d) \sum_{t=-\infty}^{T-1} \sum_{p \in \mathcal{U}_s(t+1)} P(p, t \rightarrow s, T) \cdot \\ &\quad \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}. \end{aligned}$$

$$\begin{aligned} \frac{\partial e_s^{\mathcal{P}}}{\partial w} &= (x_s^{\mathcal{P}} - x_s^d) \sum_{t=1}^{T-1} \sum_{p \in \mathcal{U}_s(t+1)} P(p, t \rightarrow s, T) \cdot \\ &\quad \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}. \end{aligned} \quad (9)$$

Since the stable point of the system does not depend on the initial state, without loss of generality, the unfolding can be interrupted at time 1 (the graph is unfolded T times) assuming that T is large enough so that the state before time 1 has no effect on the final state (this is also expressed by the fact that the probability of following a path in the unfolding $P(p, t \rightarrow s, T)$ converges to 0 when the path length $T-t$ increases),

$$\begin{aligned} \frac{\partial e_s^{\mathcal{P}}}{\partial w} &= (x_s^{\mathcal{P}} - x_s^d) \sum_{t=1}^{T-1} \sum_{p \in \mathcal{U}_s(t+1)} P(p, t \rightarrow s, T) \cdot \\ &\quad \cdot \frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}. \end{aligned} \quad (9)$$

In the following sections, we compute the derivative $\frac{\partial x_p^{\mathcal{P}}(t+1)}{\partial w(t)}$ to be inserted into equation (9) for a transition, jump and behavior parameter.

3.2 Learning the transition parameters

For a transition parameter $w = x(c_i|c_j, l)$, equation (2) shows that $\frac{\partial x_p^P(t+1)}{\partial x(c_i|c_j, l)(t)} = 0$ if $c^P \neq c_i$. On the other hand if $c^P = c_i$, it holds that,

$$\frac{\partial x_p^P(t+1)}{\partial x(c_i|c_j, l)(t)} = \sum_{q \in pa(p)} x(l|c^q)(t) x_q^P(t) \cdot \frac{\partial \left[\frac{x(c^P|c^q, l)(t)}{\sum_{k \in ch(q)} x(c^k|c^q, l)(t)} \right]}{\partial x(c_i|c_j, l)(t)}$$

The application of the derivative rule for a fraction of functions, yields that when $c^P = c_i$,

$$\frac{\partial x_p^P(t+1)}{\partial x(c_i|c_j, l)(t)} = \sum_{q \in pa(p)} x(l|c^q)(t) \cdot x_q^P(t) \cdot \frac{\sum_{k \in ch(q)} x(c^k|c^q, l)(t) - x(c^P|c^q, l)(t) \cdot |k \in ch(q) : c^k = c_i|}{\left[\sum_{k \in ch(q)} x(c^k|c^q, l)(t) \right]^2} \quad (10)$$

where $|k \in ch(q) : c^k = c_i|$ is the number of children nodes of q having label c_i .

Without loss of generality, it can be assumed that the system has already converged at time 0 (i.e. $x_q = x_q(t)$, $t = 0, \dots, T$, $\forall q \in \mathbf{G}$). Inserting equation (10) into (9) and, after removing all temporal dependencies from the parameters and from the node scores under the assumption of model stationarity and convergence at time 0, the derivative of the instant error for the considered transition parameter is obtained.

Averaging over all the examples, the model parameters can be updated by gradient descent according to equation (4). The new parameters can be used to compute a new score distribution over the nodes x_p^P . Function optimization and score estimation can be iterated through multiple epochs, till a termination criterion is satisfied.

3.3 Learning the jump parameters

For a jump parameter $w = x(c_i|c_j, J)$, equation (2) shows that it holds that,

$$\frac{\partial x_p^P(t+1)}{\partial x(c_i|c_j, J)(t)} = \begin{cases} 0 & \text{if } c^P \neq c_i \\ \frac{x(J|c_j)(t)}{|c_i|} \cdot \sum_{q \in \mathbf{G} : c^q = c_j} x_q^P(t) & \text{if } c^P = c_i \end{cases}$$

where $|c_i|$ indicates the number of nodes in the graph $\mathbf{G}$ with label c_i . The previous equation can be inserted into (9) and, removing all temporal dependencies under the assumption of model convergence and stationarity, yields the derivative of the instant error with respect to a jump parameter.

3.4 Learning the surfer action bias

Following a procedure similar to that shown for the transition and jump parameters, we start computing the derivative

$\frac{\partial x_p^P(t+1)}{\partial x(l|c_i)(t)}$ for a single parameter.

In particular, equation (2) shows that,

$$\begin{aligned} \frac{\partial x_p^P(t+1)}{\partial x(l|c_i)(t)} &= \sum_{q \in pa(p) : c^q = c_i} \frac{x(c^P|c_i, l)(t)}{\sum_{k \in ch(q)} x(c^k|c_i, l)(t)} \cdot x_q^P - \\ &- \sum_{q \in \mathbf{G} : c^q = c_i} \frac{x(c^P|c_i, J)(t)}{|c^P|} \cdot x_q^P(t), \quad (11) \end{aligned}$$

where the surfer model is assumed to respect the probabilistic constraints at every time step, i.e. $x(J|c_i)(t) = 1 - x(l|c_i)(t)$.

Merging equation (9) and (11) under the assumption of model stationarity and convergence at time 0, yields the derivative of the instant error with respect to a behavior parameter. All parameters can be simultaneously updated as described in section 3.2, 3.3 and 3.4 to train the model. After the updating, the jump and behavior parameters must be normalized to respect the probabilistic constraints.

Finally, using an EM-style algorithm [Dempster *et al.*, 1977] the adapted surfer model can re-surf the Web graph yielding a new estimate of the scores. The procedure can be repeated till a stop criterion is satisfied. Since the parameters are estimated from a few levels deep unfoldings, only a small set of nodes must be considered at each iteration. The main overhead of the proposed learning algorithm is the full PageRank computation that must be completed at each step. Our experiments show that usually only very few iterations are needed to converge.

4 Experimental results

The experiments were performed on two datasets, which were collected by focus crawling the Web [Diligenti *et al.*, 2000], on the topics “wine” and “Playstation”, respectively. The “wine” dataset contains 1.000.000 documents, while the “Playstation” one 500.000. The documents were clustered using a hierarchical version of the k-means algorithm [Duda and Hart, 1973]. In particular, in two different runs, we clustered the pages from the dataset on topic “Playstation” into 25 sets, whereas the dataset on topic “wine” was clustered into 25 and 100 sets.

Two topic-specific search engines were built containing the downloaded documents. The rank of each page in the dataset was computed using a random walker producing the standard PageRank scores. We tested the quality of the ranking function by submitting a set of queries to the search engine whose results were sorted by the scores assigned by the random surfer. By browsing the results, we found pages which were authorities on the specific topic but were (incorrectly) not inserted in the top positions of the result list. Such pages were selected as examples of pages which should get a higher rank (positive examples). A target score equal to one (the maximum reachable score according to the model) was assigned to the positive examples. On the other hand, we found pages having a high rank which were not authorities for the topic. Such pages were selected as negative examples and a target score equal to zero was assigned to those pages. The learning algorithm, as previously described was applied to these datasets, using the selected examples. Only a small set of examples (3-15) was typically labeled as positive or negative in any single experiment.

4.1 Analysis of the effects of learning

This first group of experiments aims at demonstrating that the surfer model as learned by the proposed training algorithm, could not be generated by a non-adaptive schema as the focused versions of PageRank proposed in the literature.

In figure 2, we show the values of the transition, jump and behavior parameters for each cluster, resulting from the

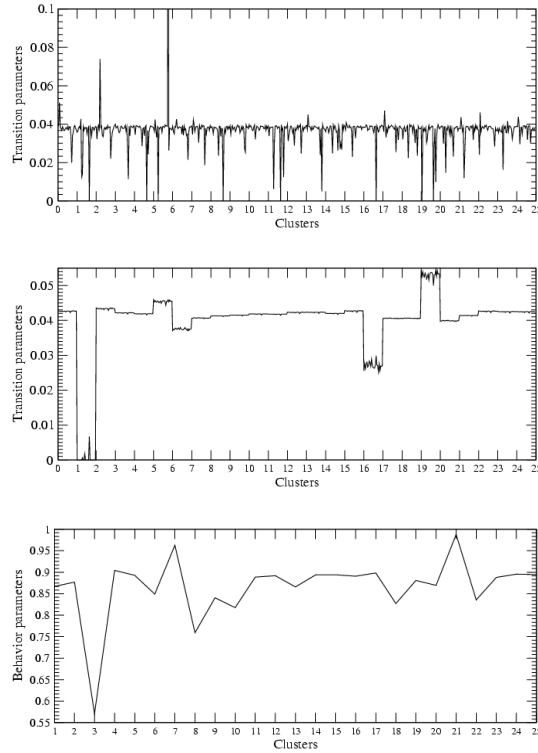
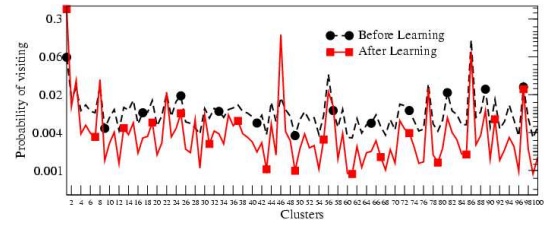


Figure 2: Values of the transition, jump and behavior parameters after learning for the “Playstation” dataset. For each cluster (x axis), all the n parameters to the other clusters are sequentially plotted.

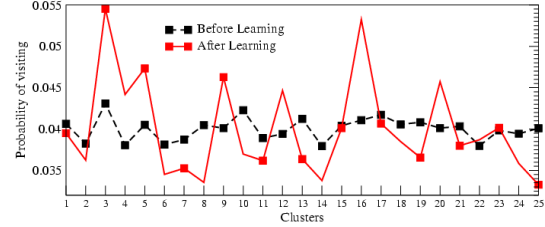
training sessions for a surfer on the topic “Playstation”. The learned parameters represent a complex interaction of the resulting surfer with the page clusters where a few clusters are strongly boosted and a few other strongly demoted. This behavior (expecially targeted demotions) can not be expressed by the focused versions of PageRank [Richardson and Domingos, 2002; Diligenti *et al.*, 2002], which simply boost all the parameters leading to the clusters of relevant pages (according to the simplistic assumption that relevant pages are more likely to point to other relevant pages). This demonstrates that the trained surfer is able to exploit hierarchies of topics to model the complex path leading to good/bad pages.

In figure 3-(a) and 3-(b), there are plotted the probabilities that the surfer is located in a page of a specific cluster. In particular we show the results for the dataset on topic “wine” clustered into 100 groups and a random graph with 200.000 nodes clustered into 25 groups. The learning algorithm is able to increase the likelihood of the random surfer to visit pages belonging to a set of clusters, while decreasing its probability of visiting pages belonging to other clusters.

Figure 4 shows the value of the cost function E^P at each iteration. Each plot reports the cost function when a specific set of parameters is optimized during training. The behavior parameters are less effective than the jump parameters in optimizing the surfer model. However, neither the behavior or the jump parameters are as effective as the transition parameters. As expected, when all parameters considered during the



(a)



(b)

Figure 3: Plots of the normalized probability of the surfer of being located in a page of a given cluster before and after learning. (a) results on the crawl for the topic “wine” with 100 clusters. (b) results on a random graph with 200.000 nodes and 25 clusters.

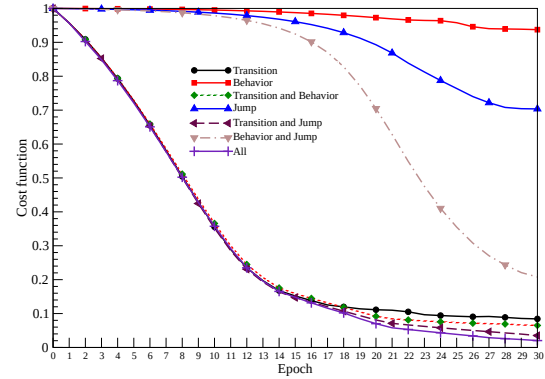


Figure 4: Values of cost function as a function of the iteration number. Each plot reports the cost function when a specific set of parameters is considered during training.

learning process, the cost function reaches its minimum.

In figure 5, there are shown the scores of the pages of cluster 8 of the dataset “Playstation” before and after learning. This cluster was manually detected as strongly on the topic “Playstation” and 3 pages belonging this cluster were explicitly inserted in learning set as pages that should get an higher score. As expected the rank of the other pages in this cluster was boosted up thanks to the capability of the algorithm to generalize from a small number of training examples.

4.2 Qualitative results

This experiment compares the quality of the page ranks before and after training. In particular, some authoritative pages on the topic “wine” were provided as positive examples, while some authoritative pages on other topics were provided as negative examples. Figure 6 reports the pages which obtained the largest variation in their rank. The pages that ob-

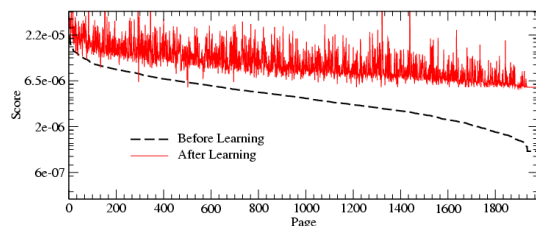


Figure 5: The scores of the pages of cluster 8 of the dataset “Playstation” before and after learning. In the learning process only 3 pages belonging to the cluster 8 were explicitly inserted in learning set as pages that should get an higher score.

Most descending pages on topic “wine”

<http://www.macromedia.com/go/getflashplayer>
<http://www.netscape.com>
<http://www.macromedia.com/downloads>
<http://www.yahoo.com>
<http://www.adobe.com/products/acrobat/readstep.html>
<http://help.yahoo.com>
<http://www.nl.placestostay.com/index.html>
<http://home.netscape.com>
<http://www.forgottensoldier.com>
<http://www.inntravels.com>

Most ascending pages on topic “wine”

<http://www.winecountry.com>
<http://webwinery.com>
<http://www.ny-wine.com/wineinfo.asp>
<http://www.wine-searcher.com>
<http://www.insidewine.com>
<http://webwinery.com>
<http://www.viniesaporidipuglia.com/en/index.html>
http://www.vinitaly.com/home_en.asp
<http://www.wineinstitute.org>
<http://www.winespectator.com>

Figure 6: URLs of the pages that yielded either the largest negative or positive rank changes among the pages that were initially in the top 1000 scoring documents.

tained a negative rank variation were either topic-generic authoritative pages (e.g. www.netscape.com) or pages relevant for different topics than “wine” (e.g. “www.forgottensoldier.org”). Among the pages that got a negative rank variation, only the page “www.yahoo.com” was specified as a negative example. On the other hand, the pages that obtained a positive rank variation were effectively authoritative pages on the topic “wine” (e.g. “www.wine-searcher.com” or “webwinery.com”). Among the most ascending documents, only the page “www.winespectator.com” was explicitly provided to the system as a positive example, whereas other pages were pushed up thanks to the capability of the system to generalize from a small set of training examples.

5 Conclusions

In this paper we introduced a novel algorithm to learn a score distribution over the nodes of a labeled graph. The learning algorithm adapts the parameters of a random surfer in order to match the target scores assigned on a small subset of the nodes. Since the parameters are estimated from the unfoldings of the graph centered on the example nodes, the learning algorithm is efficient and can be applied to graphs

with many nodes. The learning algorithm was applied to the problem of learning a generic ranking function on the portion of the Web graph.

References

- [Chang *et al.*, 2000] H. Chang, D. Cohn, and A. K. McCallum. Learning to create customized authority lists. In *Proceedings of the 17th International Conference on Machine Learning*, pages 127–134. Morgan Kaufmann, 2000.
- [Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:185–197, 1977.
- [Diligenti *et al.*, 2000] M. Diligenti, F. Coetzee, S. Lawrence, L. Giles, and M. Gori. Focus crawling by context graphs. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)*, pages 527–534, 2000.
- [Diligenti *et al.*, 2002] M. Diligenti, M. Gori, and M. Maggini. Web page scoring systems for horizontal and vertical search. In *Proceedings of the World Wide Web Conference (WWW2002)*, pages 508–516, 2002.
- [Diligenti *et al.*, 2003] M. Diligenti, M. Gori, and M. Maggini. A learning algorithm for web page scoring systems. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI2003)*, pages 575–580, 2003.
- [Duda and Hart, 1973] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [Haykin, 1994] S. Haykin. *Neural networks*. Macmillan College Publishing Company, 1994.
- [Kondor and Lafferty, 2002] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the International Conference on Machine Learning (ICML2002)*, pages 315–322, 2002.
- [Page *et al.*, 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Computer Science Department, Stanford University, 1998.
- [Richardson and Domingos, 2002] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems 14*, pages 1441–1448, Cambridge, MA, 2002.
- [Tsoi *et al.*, 2003] A. C. Tsoi, G. Morini, F. Scarselli, M. Hagenbuchner, and M. Maggini. Adaptive ranking of web pages. In *Proceedings of the 12th International World Wide Web Conference (WWW12)*, pages 356–365, 2003.

Reinforcement Learning in POMDPs Without Resets

Eyal Even-Dar

School of Computer Science
Tel-Aviv University
Tel-Aviv, Israel 69978
evend@post.tau.ac.il

Sham M. Kakade

Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
skakade@linc.cis.upenn.edu

Yishay Mansour

School Computer Science
Tel-Aviv University
Tel-Aviv, Israel 69978
mansour@post.tau.ac.il

Abstract

We consider the most realistic reinforcement learning setting in which an agent starts in an unknown environment (the POMDP) and must follow one continuous and uninterrupted chain of experience with no access to “resets” or “offline” simulation. We provide algorithms for general connected POMDPs that obtain near optimal average reward. One algorithm we present has a convergence rate which depends *exponentially* on a certain horizon time of an optimal policy, but has *no dependence* on the number of (unobservable) states. The main building block of our algorithms is an implementation of an *approximate* reset strategy, which we show always exists in every POMDP. An interesting aspect of our algorithms is how they use this strategy when balancing exploration and exploitation.

1 Introduction

We address the problem of lifelong learning in a partially observable Markov decision process (a POMDP). We consider the most general setting where an agent begins in an unknown POMDP and desires to obtain near optimal reward. In this setting, the agent is forced to obey the dynamics of the environment, which, in general, do not permit resets.

The problem of lifelong learning has been well studied for observable MDPs. Kearns and Singh (1998) provide the E^3 algorithm, which has finite (polynomial) time guarantees until the agent obtains near optimal reward. Unfortunately, such an algorithm is not applicable in the more challenging POMDP setting. In fact, none of the guarantees in the literature for learning in the limit for MDPs apply to POMDPs, for reasons which are essentially due to the partially observability.

For POMDPs, the problem of balancing exploitation with exploration has received rather little attention in the literature — typically most results in POMDPs are on planning (see for example Sondik (1971); Lovejoy (1991a, 1991b); Hauskrecht (1997); Cassandra (1998)). Most of the existing learning algorithms such as Parr and Russell (1995); Peshkin et al. (2000) either assume a goal state or assume a reset button. In fact, to the best of our knowledge, the literature does not

contain even asymptotic results for general POMDPs which guarantee that the average reward of an agent will be near optimal in the limit.

Part of the technical difficulty is that there are currently no general results for belief state tracking with an approximate model showing that divergence in the belief state does not eventually occur. Crudely, the issue is that if a belief state is being tracked in an approximate manner, then it is important to show that this approximation quality does not continually degrade with time — otherwise the agent will eventually lose track of the belief state in the infinite horizon (this of course is not an issue in an MDP where the current state is observable). Boyen and Koller (1998) address the issue of approximate belief state tracking, but in their setting the model is known perfectly and their goal is to keep a compact representation of the belief state. Note that approximate belief state tracking is much simpler if the agent is only acting over a fixed finite horizon, since then one can bound the error accumulation as a function of the horizon.

We present new algorithms for learning in POMDPs which guarantee that the agent will obtain the optimal average reward in the limit. Furthermore, we provide a finite time convergence rates for one of our algorithms which has an *exponential* dependence on a certain horizon time (of an optimal strategy) but has *no dependence* on the number of states in the POMDP. This result is reminiscent of the trajectory tree algorithm of Kearns et al. (1999) which has similar dependencies (though there they assumed access to a generative model, which allowed simulation of the POMDP). Given the plethora of complexity results in the literature on planning in POMDPs (see Lusena et al. (2001)), we feel these dependencies are the best one could hope for in the most general setting.

Central to our algorithms is the implementation of an approximate reset strategy or a *homing strategy*. The idea of a reset strategy is not new to the literature — homing sequences were used in the learning of deterministic finite automata (see Rivest and Schapire (1993), though there the sequence provided *exact* resets). Here, the agent follows a homing strategy in order to move *approximately* towards a reset. We show that such a strategy always exists, and our finite convergence rates also depend on a characteristic time it takes to approximately reset. However, note that existence of such a strategy alone does not imply that such a strategy will be useful.

The reason is that the agent must take actions to reset, which might otherwise be better spent exploring or exploiting. It turns out that our algorithms use the homing strategy while *both* exploring and exploiting. In fact, they use the homing strategies infinitely often, which, unfortunately, detracts from exploiting. However, we are able to show that the ratio of the time these homing strategies are used compared to the time spent exploiting is decreasing sufficiently rapidly, such that near optimal average reward can be obtained.

2 Preliminaries

A Partially Observable Markov Decision Process (POMDP) is defined by a finite set of states S , an initial state s_0 , a set of actions A , a set of observations O , with an output model Q , where $Q(o, r|s, a)$ is the probability of observing o and reward r after performing action a at state s (we assume that $r \in [0, 1]$), and a set of transitions probabilities P , where $P(s'|a, s)$ is the transition probability from s to s' after performing action a . We define $r(s, a)$ as the expected reward under $Q(\cdot|s, a)$ after performing action a in state s .

A *history* h is a sequence of actions, rewards and observations of some finite length, i.e. $h = \{(a_1, r_1, o_1), \dots, (a_t, r_t, o_t)\}$. A *strategy* or *policy* in a POMDP is defined as a mapping from histories to actions. We define a *belief state* B to be a distribution over states. Given an initial belief state B_0 let $\Pr[h|B_0] = \Pr[r_1, o_1, \dots, r_t, o_t|a_1, \dots, a_t, B_0]$ be the probability of observing the sequence of reward-observations $(r_1, o_1, \dots, r_t, o_t)$ after performing the actions $a_1 \dots a_t$.

For each strategy π we define its t -horizon expected reward from a belief state B as $R_t^\pi(B) = (1/t)E_{h \sim \pi}[\sum_{i=1}^t r(s_i, a_i)|B_0 = B]$. A t -Markov strategy is a strategy that depends only on the last t observations. The optimal t -Markov strategy's expected return from initial belief state B is defined as $R_t^*(B)$.

The only assumption we make is that the POMDP is *connected*, i.e. for all states s, s' , there exists a strategy π which can reach s' with positive probability starting from s . (We do not make any ergodicity assumptions, since strategies are by definition non-stationary). Note that if the POMDP is disconnected, then the best statement we could hope for is to obtain the optimal average reward for one of its connected components.

Connectivity implies that there exists a strategy π^* that maximizes the average reward. More formally, there exists a π^* such that: i) for every B , $\lim_{t \rightarrow \infty} R_t^{\pi^*}(B)$ exists and does not depend on B , which we denote by R^* , and ii) for all π and B , $R^* \geq \lim_{t \rightarrow \infty} \sup R_t^\pi(B)$. Hence, for all $\epsilon > 0$ there exist a τ , such that for all B and $t \geq \tau$:

$$|R^* - R_t^{\pi^*}(B)| \leq \epsilon$$

and we refer to τ as the ϵ -horizon time of the optimal strategy. Essentially, τ is the timescale in which the optimal strategy achieves close to its average reward.

When we say that we *restart* a t -Markov strategy π from a belief state B we mean specifically that we reset the history, i.e., $h = \emptyset$, and run π starting from a state s distributed according to B .

3 Homing Strategies

Clearly, having an action which resets the agent to some designated state would be useful, as it would allow us to test and compare the performance of various policies, starting at the same start state. However, in general, such an action is not at our disposal.

Instead our algorithms utilize an approximate reset, which we show always exists. There are a few subtle points when designing such a reset. First, we must select actions to achieve the approximate reset, i.e. the approximate reset is done through the use of a *homing strategy*. Hence, while homing, the agent is neither exploring nor exploiting. Second, rather than moving to a fixed state, the homing strategy can only hope to move toward a fixed (unknown) belief state. Third, as we shall see, since the POMDP might be periodic the stopping time must be a random variable¹. To implement this randomized stopping time, we introduce *fictional* 'stay' actions, in which the agent does not take an action that period. By this, we mean that if the homing strategy decides to take a 'stay' action at some time — which may not be possible if the true POMDP does not permit 'stay' actions — then the agent just ignores this 'stay' action and obtains another action from the homing strategy to execute. Hence, after the agent has taken t homing actions (which are either real or 'stay' actions), the agent has taken $t - m$ real actions in the POMDP and m stay actions. We now define an approximate reset strategy.

Definition 3.1 H is an (ϵ, k) -approximate reset (or homing) strategy if for every two belief states B_1 and B_2 , we have $\|H_E(B_1) - H_E(B_2)\|_1 \leq \epsilon$, where $H_E(B)$ is the expected belief state reached from B after k homing actions of H (so at most k real actions have been taken) and $H(B)$ is a random variable such that $H_E(B) = E_{H(B) \sim H, B}[H(B)]$.

The above definition only states that H will approximately reset, but this approximation quality could be poor. We now show how to amplify the accuracy of an approximate homing strategy, and then we show that an approximate homing strategy always exists.

Lemma 3.1 Suppose that H is an (ϵ, k) approximate reset then H^ℓ is an $(\epsilon^\ell, k\ell)$ approximate reset, where H^ℓ consecutively implements H for ℓ times. Furthermore, this implies there exists a unique belief state B_H such that $H_E(B_H) = B_H$.

Proof: The proof is a standard contraction argument, and we use induction. For $\ell = 1$, the claim follows by definition. Assume now that $\|H_E^{\ell-1}(B_1) - H_E^{\ell-1}(B_2)\|_1 \leq \epsilon^{\ell-1}$. Let $H_E^{\ell-1}(B_1) = Q_1$ and $H_E^{\ell-1}(B_2) = Q_2$, so $\sum_s |Q_1(s) - Q_2(s)| \leq \epsilon^{\ell-1}$. For an arbitrary state s' , and using the fact

¹Without randomizing over the stopping times (i.e. allowing 'stay' actions), the state transition matrix may be periodic and no stationary distribution may exist, e.g. if the states deterministically alternate between states 1 and 2.

that H is a linear operator, we have

$$\begin{aligned}
& \|H_E^\ell(B_1) - H_E^\ell(B_2)\|_1 \\
&= \|H_E(Q_1) - H_E(Q_2)\|_1 \\
&= \left\| \sum_s (Q_1(s) - Q_2(s)) H_E(s) \right\|_1 \\
&= \left\| \sum_s (Q_1(s) - Q_2(s)) H_E(s') \right\|_1 \\
&\quad + \left\| \sum_s (Q_1(s) - Q_2(s)) (H_E(s) - H_E(s')) \right\|_1 \\
&= 0 + \sum_s |Q_1(s) - Q_2(s)| \epsilon \\
&\leq \epsilon^\ell
\end{aligned}$$

where the first term is 0 since for any two distributions $\sum_s Q_1(s) - Q_2(s) = 1 - 1 = 0$ (and the vector $H(s')$ is constant in this sum), and we have used the fact that $\|H(s) - H(s')\|_1 \leq \epsilon$ (by definition of H). $\square$

We now show that the random walk strategy (including 'stay' actions) is an approximate reset strategy in every POMDP (including periodic ones), though with prior knowledge we might have better approximate reset strategies at our disposal.

Lemma 3.2 *For all POMDPs, the random walk strategy (using 'stay' actions) constitutes an (ϵ, k) approximate reset strategy for some $k \geq 1$ and $0 < \epsilon < \frac{1}{2}$.*

Proof: By our connectivity assumption, for all states s and s' , there exists some strategy that reaches s' from s with positive probability. This implies that under H (the random walk strategy), there is positive probability of moving from one state to another, i.e. the Markov chain is irreducible. Furthermore, since H performs 'stay' actions, then the Markov chain is aperiodic. Thus, there exists a unique stationary distribution. We choose k to be the time at which the error in convergence is less than $1/2$, from all starting states. Hence, by linearity of expectation, the error is less than $1/2$ from all belief states in k steps. $\square$

4 Reinforcement Learning with Homing

We now provide two algorithms which demonstrate how near-optimal average reward can be obtained, with different rates of convergence. The key to the success of these algorithms is their use of homing sequences in *both* exploration and in exploitation. For exploration, the idea is that each time we attempt some exploration trajectory we do it after implementing our reset strategy — hence our information is (approximately) grounded with respect to the belief state B_H (recall $H_E(B_H) = B_H$). The idea of exploration is to find a good t -Markov strategy $\hat{\pi}_t^*$ from B_H . During exploitation, the goal is to use this t -Markov strategy. Unfortunately, we have only guaranteed that $\hat{\pi}_t^*$ performs well *starting from* B_H and only for t steps. Hence, after each time we exploit with $\hat{\pi}_t^*$, we run our homing sequence to get back close to B_H (and then we rerun $\hat{\pi}_t^*$). We gradually increase t in the process.

The problem is that while homing, we are wasting time and neither exploiting nor exploring. Furthermore, since we use

```

Input :  $H$  /*a  $(1/2, K_H)$  approximate reset strategy */
for  $t = 1$  to  $\infty$  do
  /*Exploration in Phase  $t$  */;
   $k_1^t = O\left(\frac{1}{\epsilon_t} \log(t^2 |\Pi_t|)\right)$ ;
  foreach Policy  $\pi$  in  $\Pi_t$  do
    for  $i = 1$  to  $k_1^t$  do
      Run  $\pi$  for  $t$  steps;
      Repeatedly run  $H$  for  $\log(1/\epsilon_t)$  times;
    end
    Let  $v^\pi$  be the average return of  $\pi$  in from these
     $k_1^t$  trials;
  end
  /*Exploitation in Phase  $t$  */;
  Let  $\hat{\pi}_t^* = \arg \max_{\pi \in \Pi_t} v^\pi$ ;
   $k_2^t = O\left(\frac{1}{\epsilon_t} ([\text{current time } T] + [\text{time in } t + 1\text{-th exploration phase}])\right)$ ;
  for  $i = 1$  to  $k_2^t$  do
    Run  $\hat{\pi}_t^*$  for  $t$  steps;
    Repeatedly run  $H$  for  $\log(1/\epsilon_t)$  times;
  end
end

```

Algorithm 1: Policy Search

the homing sequence between *every* run of $\hat{\pi}_t^*$, asymptotically we never stop homing. Nonetheless, we are able to show that there exists an algorithm which obtains near optimal reward in a POMDP, since the ratio of the time spent exploiting vs. homing decreases sufficiently fast.

Theorem 4.1 *There exists an algorithm $\mathcal{A}$, such that in any connected POMDP, $\mathcal{A}$ obtains the optimal average reward in the limit, with probability 1.*

We later provide an algorithm with a better convergence rate (see Theorem 4.5). However, we start with a simpler policy search algorithm which establishes the above Theorem.

4.1 Policy Search

Algorithm 1 takes as input a $(1/2, K_H)$ -approximate reset strategy, which could be the random walk strategy with a very crude reset. The algorithm works in phases, interleaving exploration phases with exploitation phases. Let us start by describing the exploration phases. Let Π_t be the set of all t -Markov strategies. An estimate of the value of a policy $\pi \in \Pi_t$ can be found by first resetting and then running π for t steps. The exploration phase consists of obtaining an estimate v^π of the return of each policy $\pi \in \Pi_t$, where each estimate consists of an average of k_1 trials (followed by approximate resets).

These estimates have both bias and variance. The variance is just due to the stochastic nature of the POMDP. The bias is due to the fact that we never can exactly reset to B_H . However, if we run H for $\log 1/\epsilon_t$ times (where ϵ_t is an error parameter in the t -th phase, which will be fixed later) then, by lemma 3.1, all expected belief states we could approach will be $(1/2)^{\log 1/\epsilon_t} = \epsilon_t$ close to B_H . The following lemma shows that accurate estimates can be obtained.

Lemma 4.2 *In phase t , if $k_1^t = O\left(\frac{1}{\epsilon_t^2} \log(t^2 |\Pi_t|)\right)$ and each reset consists of using the homing sequence $\log 1/\epsilon_t$ times, then for all policies $\pi \in \Pi_t$, the estimated t -horizon reward, v^π , satisfies $|R_t^\pi(B_H) - v^\pi| \leq 2\epsilon_t$ with probability greater than $1 - \frac{1}{2t^2}$.*

Proof: First, let us deal with the bias. Any expected belief states b that results from using the homing sequence $\log 1/\epsilon_t$ times must satisfy $\|b - B_H\| \leq (1/2)^{\log 1/\epsilon_t} \leq \epsilon_t$. Now it is straightforward to see that if b and B_H are belief states such that $\|b - B_H\|_1 \leq \epsilon$, then for every strategy π , $|R_t^\pi(b) - R_t^\pi(B_H)| \leq \epsilon$. To see this, let the belief states at time t be b^t and B_H^t , which result from following either π starting from b or B_H , respectively. By linearity of expectation, it follows that $\|b^t - B_H^t\|_1 \leq \epsilon$. This directly implies that $|R_t^\pi(b) - R_t^\pi(B_H)| \leq \epsilon$.

For the variance, the Hoeffding bound and our choice of k_1^t imply that the average return of each policy is ϵ_t close to its expectation (the expectation is both over the initial state and on the policy trajectory) with probability $1 - 1/(2t^2)$. $\square$

Now during exploitation, the algorithm uses the policy $\hat{\pi}_t^*$ which had the highest return in the exploration phase (and by the previous lemma this is close to the policy with largest return). Note that we have only guaranteed a large return for executing $\hat{\pi}_t^*$ from B_H for t steps. However, we would like to exploit for a longer period of time than t . The key is that we again reset $\log(1/\epsilon_t)$ times after each time we run $\hat{\pi}_t^*$, which resets us close ϵ_t close to B_H . Unfortunately, this means we spend $K_H \log 1/\epsilon_t$ steps between each run of $\hat{\pi}_t^*$. Hence, our average return could be $O(\frac{1}{t} K_H \log 1/\epsilon_t)$ less than we would like, since $O(\frac{1}{t} K_H \log 1/\epsilon_t)$ is the fraction of time we spend resetting. Note that this fraction could be large if we desire that ϵ_t be very small (though this would guarantee very accurate resets).

Now, when we do exploit (and reset), we run the exploitation phase long enough (for k_2^t time) such that our overall average reward is comparable to the average reward in the last exploitation phase.

Lemma 4.3 *At any time T after phase t , the average reward from time 1 to time T satisfies: $\frac{1}{T} \sum_{i=1}^T r_i \geq R_t^*(B_H) - O(\epsilon_t + \frac{1}{t} K_H \log(1/\epsilon_t))$ with probability at least $1 - \frac{1}{t^2}$.*

Before proving this lemma, let us state a corollary from which Theorem 4.1 follows.

Corollary 4.4 *Let $\epsilon_t = 1/t$. Then $\frac{1}{T} \sum_{i=1}^T r_i \geq R_t^*(B_H) - O(\frac{K_H \log(t)}{t})$ with probability at least $1 - \frac{1}{t^2}$.*

Importantly, note the loss term goes to 0 as t goes to infinity. Furthermore, for a large enough phase t , we know that $R_t^*(B_H)$ will approach the optimal average reward R^* (since R^* is independent of the starting B). Theorem 4.1 follows. Essentially, although we have to home infinitely often, the ratio of time spent homing to the time spent using our t -step exploitation policies is going as $O(\frac{\log t}{t})$, which goes to 0.

Proof: First, let us show that the average reward, $\frac{1}{T} \sum_{i=1}^T r_i \geq R_t^*(B_H)$, is no less than ϵ_t from the average reward obtained in the t -th exploitation phase. To do this, we set the time of exploitation phase, k_2^t , to be $1/\epsilon_t$ times greater

than previous amount of time spent in the MDP time plus the amount of time that will be spent in the next exploration phase (this latter factor accounts for the case in which time T lies in the exploration phase immediately after t).

Now we bound the average reward obtained in the exploitation phase. First, let us show that the t -average reward of the policy used, $R_t^{\hat{\pi}_t^*}$, satisfies $R_t^{\hat{\pi}_t^*}(B_H) - R_t^{\pi_t^*}(B_H) \leq 4\epsilon_t$ with probability at least $1 - \frac{1}{2t^2}$. By Lemma 4.2 for each policy $\pi \in \Pi_t$, we have $|R_t^\pi(B_H) - v^\pi| \leq 2\epsilon_t$ with probability at least $1 - \frac{1}{2|\Pi_t|t^2}$. Therefore, $\hat{\pi}_t^*$ is $4\epsilon_t$ -optimal with probability $1 - 1/(2t^2)$. Now the observed average return of $\hat{\pi}_t^*$ in the exploitation period is $2\epsilon_t$ close to $R_t^{\hat{\pi}_t^*}(B_H)$ with probability at least $1 - \frac{1}{2t^2}$, since our observed average return in exploitation is at least as good as those used to find $v^{\hat{\pi}_t^*}$ (since $k_2^t > k_1^t$).

However, the average return during the exploitation phase is not the observed average return of $\hat{\pi}_t^*$, since we reset after each t exploitation steps for a number of steps that is $K_H \log(1/\epsilon_t)$. The resets in the exploitation period can change the average reward by at most a fraction $\frac{1}{t} K_H \log(1/\epsilon_t)$. $\square$

4.2 A Model Based Algorithm

The previous algorithm was the simplest way to demonstrate Theorem 4.1. However, it is very inefficient, since it is testing all t -Markov policies — there are doubly exponential, in t , such policies². Here, we provide a more efficient model based algorithm, which resembles the algorithms given in Kearns et al. (1999); McAllester and Singh (1999), and is exponential in the horizon time, yet it still has no dependence on the number of states in the POMDP.

We now state a convergence rate in terms of τ , the ϵ -horizon time of an optimal policy (see Section 2) and in terms of the homing time K_H (recall, such a time exists for every POMDP using a random walk policy).

Theorem 4.5 *There exists an algorithm $\mathcal{A}$, such that in any connected POMDP and with probability greater than $1 - \delta$, $\mathcal{A}$ achieves an average reward that is 2ϵ close to the optimal average reward in a number of steps in the POMDP which is polynomial in $|A|, |O|, K_H$ and $\log(1/\delta)$ and exponential in τ . Furthermore the computational runtime of this algorithm is polynomial in $|A|, |O|$, and $\log(1/\delta)$ and exponential τ .*

We provide such an algorithm in the next page. In exploration phase, the algorithm builds an approximate model of the transition probabilities after some history has occurred starting from B_H . In the t -th phase, it builds a model with respect to the set of all t -length histories, which we denote by $\mathcal{H}_t$. In the exploitation phase, it uses the best t Markov strategy with respect to this model. The use of homing strategies is similar to that in the previous algorithm.

Let $L = |A||O|$, and note that $2L^t \geq |\mathcal{H}_t|$. In the exploration phase, the algorithm takes actions uniformly at random for t steps and then resets (running the homing strategy

²The number of histories of length t is exponential in t , and the number of t -Markov policies is exponential in the number of t -length histories

```

Input :  $H$  /*an  $(1/2, K_H)$  approximate reset strategy */
Let  $L = |A| \cdot |O|$ ;
for  $t = 1$  to  $\infty$  do
   $k_1^t = O\left(\frac{L^{4t}}{\epsilon_t^2} \log(t^2 |\mathcal{H}_t|)\right)$ ;
  for  $k_1^t$  times do
    Run RANDOM for  $t$  steps;
    Run  $H$  for  $K_H \log(L^t/\epsilon_t)$  steps.
  end
  for  $h \in \mathcal{H}_t, a \in A$  and  $o \in O$  do
     $\hat{\Pr}[o|h, B_0, a] = 0$ ;
    if  $\hat{\Pr}[h(a, o)|B_0] \geq \frac{\epsilon_t}{L^t}$  then
       $\hat{\Pr}[o|h, B_0, a] = \frac{\hat{\Pr}[h(a, o)|B_0]}{\hat{\Pr}[h|B_0]\hat{\Pr}[a]}$ 
    end
  end
  Compute  $\hat{\pi}_t^*$  using  $\hat{\Pr}[o|B_0, h, a]$ ;
   $k_2^t = O\left(\frac{1}{\epsilon_t}([\text{current time } T] + [\text{time in } t + 1\text{-th exploration phase}])\right)$ ;
  for  $k_2^t$  times do
    Run  $\hat{\pi}_t^*$  for  $t$  steps;
    Run  $H$  for  $K_H \log(1/\epsilon_t)$  steps;
  end
end

```

Algorithm 2: Model based

for $\log(L^t/\epsilon_t)$ times). This is done k_1^t times.³ Then using the empirical frequencies in these trajectories the algorithm forms estimates $\hat{\Pr}[o|h, B_H, a]$, which is just the empirical probability of observing o conditioned on history h followed by taking action a . For histories h which are unlikely, these empirical estimates could be very bad, though, as we shall see, we do not need accurate estimates of $\hat{\Pr}[o|h, B_H, a]$ for such histories. Let $h(a, o)$ be a history with h followed by (a, o) .

Lemma 4.6 *In phase t , if $k_1^t = O\left(\frac{L^{4t}}{\epsilon_t^2} \log(t^2 |\mathcal{H}_t|)\right)$ and each reset consists of using the homing sequence $\log(L^t/\epsilon_t)$ times, then: (1) $|\hat{\Pr}[h|B_0] - \Pr[h|B_0]| \leq \frac{\epsilon_t}{L^{2t}}$, and (2) for every $h(a, o) \in \mathcal{H}_t$ such that $\Pr[h(a, o)|B_0] \geq \frac{\epsilon_t}{L^t}$, we have $|\hat{\Pr}[o|h, B_0, a] - \Pr[o|h, B_0, a]| \leq \frac{2|A|}{L^t}$, with probability at least $1 - \frac{1}{t^2}$.*

Proof: We first note that, with probability $1 - 1/t^2$, for every history $h \in \mathcal{H}_t$ we have $|\hat{\Pr}[h|B_0] - \Pr[h|B_0]| \leq \frac{\epsilon_t}{L^{2t}}$ (using the Hoeffding bound). The error, $|\Pr[o|h, B_0, a] -$

³We can use in the algorithm any approximate homing strategy H . However, if H is simply the random policy, then the reset and exploration would both use the same policy, and the algorithm would slightly simplify.

$\hat{\Pr}[o|h, B_0, a]$, is then

$$\begin{aligned}
& \left| \frac{\Pr[h(a, o)|B_0]}{\Pr[h|B_0]\hat{\Pr}[a]} - \frac{\hat{\Pr}[h(a, o)|B_0]}{\hat{\Pr}[h|B_0]\hat{\Pr}[a]} \right| \\
& \leq \frac{1}{\hat{\Pr}[a]} \left| \frac{\Pr[h(a, o)|B_0] + \frac{\epsilon_t}{L^{2t}}}{\Pr[h|B_0] - \frac{\epsilon_t}{L^{2t}}} - \frac{\Pr[h(a, o)|B_0]}{\Pr[h|B_0]} \right| \\
& = \frac{1}{\hat{\Pr}[a]} \left| \frac{\frac{\epsilon_t}{L^{2t}}}{\Pr[h|B_0] - \frac{\epsilon_t}{L^{2t}}} + \frac{2\Pr[h(a, o)|B_0]\frac{\epsilon_t}{L^{2t}}}{\Pr[h|B_0](\Pr[h|B_0] - \frac{\epsilon_t}{L^{2t}})} \right| \\
& \leq \frac{2|A|}{L^t},
\end{aligned}$$

where the first inequality holds with probability $1 - \frac{1}{t^2}$, and in the last inequality we used the fact that $\Pr[h|B_0] \geq \frac{\epsilon_t}{L^t}$. $\square$

The exploitation policy can be found using dynamic programming with the model. Note that the POMDP is equivalent to an MDP where the histories are states. In the exploitation phase, the algorithm uses the best t -Markov policy, $\hat{\pi}_t^*$, (with respect to the approximate model) interleaving it with $K_H \log(1/\epsilon_t)$ homing steps.

Lemma 4.7 *In phase t , the exploitation policy $\hat{\pi}_t^*$, satisfies $|R_t^*(B_H) - R_t^{\hat{\pi}_t^*}(B_H)| \leq t(\epsilon_t + \frac{2|A|}{L^t}) + (2\epsilon_t + \frac{2\epsilon_t}{L^t})$ with probability at least $1 - \frac{1}{t^2}$.*

Proof: (sketch) We observe that by ignoring all histories (which we view as nodes in a tree) such that $\hat{\Pr}(h|B_0) \leq \frac{\epsilon_t}{L^t}$, the return of an optimal strategy in this empirical model is decreased by at most $2t(\epsilon_t + \frac{\epsilon_t}{L^t})$, due to the fact that the true history probability is bounded by $\frac{\epsilon_t}{L^t} + \frac{\epsilon_t}{L^{2t}}$, the return from each node is bounded by t and the total number of such nodes is bounded by $2L^t$. Next we prove that the return of the optimal policy in the empirical model loses at most $t^2(\epsilon_t + \frac{2|A|}{L^t})$ due to the tree approximation on the other nodes (the other histories). Using backward induction, we show that the policy $\hat{\pi}_t^*$ has return not less than $k^2(\epsilon_t + \frac{2|A|}{L^t})$ in comparison to the true optimal value, starting from $(t - k + 1)$ -length histories. The base case, for the leaves (the t -length histories), holds since the reward (which is encoded through the observations) is within $\epsilon_t + \frac{2|A|}{L^t}$, where the first error is due to the imperfect reset and the second is due to the marginal distribution error that is bounded by $\frac{2|A|}{L^t}$ by Lemma 4.6. Assume the induction assumption holds for $k - 1$. There are two sources of error, the first is due to the current estimation error (of both the marginal distribution and the immediate reward) which is bounded by $(\epsilon_t + \frac{2|A|}{L^t})k$ and the second is due to errors from the previous levels and is bounded by $(k - 1)^2(\epsilon_t + \frac{2|A|}{L^t})$ by the induction assumption. Summing the terms completes the induction step. $\square$

Similarly to Subsection 4.1, we exploit long enough such that the overall average reward is essentially the average reward in the last exploitation period.

Lemma 4.8 *At any time T after phase t , the average reward from time 1 to time T satisfies: $\frac{1}{T} \sum_{i=1}^T r_i \geq R_t^*(B_H) - O(t\epsilon_t + \frac{|A|t}{L^t} + (1/t)K_H \log(1/\epsilon_t))$ with probability at least $1 - \frac{1}{t^2}$.*

Using the above lemma, Theorem 4.5 follows immediately if we set $\epsilon_t = 1/t^2$.

Proof: (sketch) We first note that all exploitations and explorations from phases 1 to T and from the next, $(T+1)$ -th, exploration phase can effect the average reward by at most ϵ_t . By Lemma 4.7, the exploitation policy is near optimal and satisfies, $|R_t^*(B_H) - R_t^{\pi_t^*}(B_H)| \leq t(\epsilon_t + \frac{2|A|}{L^t}) + (\epsilon_t + \frac{\epsilon_t}{L^t})$ with probability $1 - 1/(2t^2)$. As in Lemma 4.2, we observe that the bias of the exploitation policy is ϵ_t and the variance due to Hoeffding's bound and the large exploitation time is bounded by ϵ_t with probability $1 - 1/(2t^2)$. The last source for loss is the resets in the exploitation period and its effect can be bounded by $\frac{K_H \log(1/\epsilon_t)}{t}$. $\square$

A direct and simple corollary from which Theorem 4.1 follows as well.

Corollary 4.9 *Let $\epsilon_t = 1/t$. Then $\frac{1}{T} \sum_{i=1}^T r_i \geq R_t^*(B_H) - O(\frac{K_H \log(t)}{t})$ with probability at least $1 - \frac{1}{t^2}$.*

Acknowledgements

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, and by a grant from the Israel Science Foundation and an IBM faculty award. This publication only reflects the authors' views.

References

- X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *UAI*, 1998.
- A. Cassandra. *Exact and approximate algorithms for partially observable markov decision processes*. PhD thesis, Brown University, 1998.
- M. Hauskrecht. A heuristic variable-grid solution method for pomdps. In *AAAI-97*, pages 734–739, 1997.
- M. Kearns, Y. Mansour, and A. Ng. Approximate planning in large pomdps via reusable trajectories. In *NIPS 12*, 1999.
- M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Proceedings of ICML*, 1998.
- W. S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. *Operations Research*, 39(1):162–175, 1991a.
- W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28:47–66, 1991b.
- C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 14: 83–103, 2001.
- D. McAllester and S. Singh. Approximate planning for factored pomdps using belief state simplification. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 409–416, 1999.
- R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.
- L. Peshkin, K. Kim, N. Meuleau, and L.P. Kaelbling. Learning to cooperate via policy search. In *16th Proceedings of UAI*, pages 307–314, 2000.
- R. Rivest and R. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2): 299–347, 1993.
- E. Sondik. *The optimal control of partially observable processes over a finite horizon*. PhD thesis, Stanford University, Stanford, California, 1971.

A Simple-Transition Model for Relational Sequences

Alan Fern

School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331
afern@cs.orst.edu

Abstract

We use “nearly sound” logical constraints to infer hidden states of relational processes. We introduce a simple-transition cost model, which is parameterized by weighted constraints and a state-transition cost. Inference for this model, i.e. finding a minimum-cost state sequence, reduces to a single-state minimization (SSM) problem. For relational Horn constraints, we give a practical approach to SSM based on logical reasoning and bounded search. We present a learning method that discovers relational constraints using CLAUDIEN [De Raedt and Dehaspe, 1997] and then tunes their weights using perceptron updates. Experiments in relational video interpretation show that our learned models improve on a variety of competitors.

1 Introduction

We consider hidden-state inference from the observations of relational processes, i.e. processes where states and observations are described by properties of and relations among objects (e.g. people and blocks). To deal with the enormous state and observation spaces, we utilize “nearly-sound” (i.e. rarely violated) logical constraints on the states and observations. Such constraints can often be acquired via machine learning and/or a human expert, and we give a framework for combining them for relational sequential inference.

We introduce the *simple-transition cost model* (in Section 3), which is parameterized by a set of weighted logical constraints and a state-transition cost. The cost of a state sequence given an observation sequence is the weight of unsatisfied constraints plus a transition cost for each state change. Sequential inference corresponds to computing the minimum-cost state sequence. Here we study both learning and inference for this model.

First, in Section 4, we (efficiently) reduce sequential inference to a single-state minimization (SSM) problem. We then show, in Sections 5 and 6, how to leverage nearly-sound relational Horn constraints in order to compute SSM with practical efficiency. We also study the possibility of exploiting efficient SSM in richer models. We show that for a simple enrichment to our model, there is no efficient reduction unless $P=NP$. In Section 7, we discuss learning a relational simple-transition model using the logical discovery engine CLAUDIEN [De Raedt and Dehaspe, 1997] and a variant of Collins’

generalized perceptron algorithm [Collins, 2002]. Finally, in Section 8, we evaluate our approach in a relational video-interpretation domain. Our learned models improve on the accuracy of an existing trainable system and an approach based on more mainstream approximate inference.

2 Problem Setup

For simplicity, we describe our problem setup and approach for propositional (rather than relational) processes, where in the spirit of dynamic Bayesian networks (DBNs) [Murphy, 2002], states are described by a fixed set of variables. In Section 6, we extend to the relational setting.

A sequential process is a triple $(\mathcal{X}, \mathcal{Y}, \mathcal{P})$, where the *observation space* $\mathcal{X}$ and *state space* $\mathcal{Y}$ are sets that contain all possible observations and states. $\mathcal{P}$ is a probability distribution over the space of finite sequences constructed from members of $\mathcal{X} \times \mathcal{Y}$. Each such sequence contains a pair of an *observation sequence* (*o-sequence*) and a *state sequence* (*s-sequence*). For a sequence $Q = (q_1, \dots, q_T)$, we let $Q_{i:j} = (q_i, \dots, q_j)$ for $i \leq j$. We will use uppercase for sequences and lowercase for single states and observations. A process is *propositional* when its states are represented using a set of n *state variables* over the finite domain D_s , yielding $\mathcal{Y} = (D_s)^n$. The value of the i ’th variable in state s is denoted by $s^{(i)}$. We will not need to assume a representation for observations until Section 6

We assume that the utility of an inferred s-sequence S primarily derives from the sequence of distinct states, rather than identifying the exact state transition points. Let $\text{COMPRESS}(S)$ denote the sequence obtained by removing consecutive repetitions in S . For example, $\text{COMPRESS}(a, a, a, b, b, a, c, c) = a, b, a, c$. Our empirical goal is to map an o-sequences O to an s-sequence S such that $\text{COMPRESS}(S)$ is the distinct sequence of states that generated O . Indeed, in our video domain, the exact locations of state transitions are often ambiguous (as judged by a human) and unimportant for recognizing activity—e.g. in Figure 1 it is unimportant to know exactly where the transition occurs.

Example 1. *The process in our video-interpretation domain corresponds to a hand playing with blocks. Figure 1 shows key frames where a hand picks up a red block from a green block. The goal is to observe the video and then infer the underlying force-dynamic states, describing the support relations among objects. States are represented as sets of force-dynamic facts, such as $\text{ATTACHED}(\text{HAND}, \text{RED})$. Observations are represented as sets of low-level numeric facts, such*

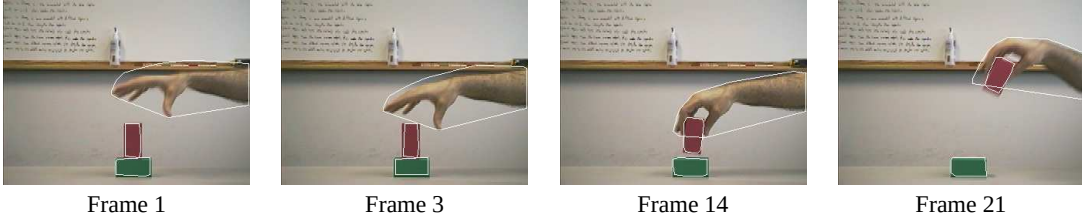


Figure 1: Key frames in a video segment showing a hand picking up a red block from a green block. The object tracker’s output is shown by the polygons. The video segment has two distinct force-dynamic states given by: $\{\text{GROUNDED}(\text{HAND}), \text{GROUNDED}(\text{GREEN}), \text{CONTACTS}(\text{GREEN}, \text{RED})\}$ (frames 1 and 3) and $\{\text{GROUNDED}(\text{HAND}), \text{GROUNDED}(\text{GREEN}), \text{ATTACHED}(\text{HAND}, \text{RED})\}$ (frames 14 and 20). The transition occurs between frames 3 and 14. See Example 2 regarding the predicates GROUNDED, CONTACTS, and ATTACHED.

as $\text{DISTANCE}(\text{GREEN}, \text{RED}) = 3$, that are derived from an object tracker’s noisy output. For a fixed set of objects, the process can be represented propositionally with a variable for each possible fact. However, our system must handle any number of objects, requiring a relational process representation described in Section 6.

3 The Simple-Transition Cost Model

Our framework utilizes an additive conditional cost model $C(S|O)$ to represent the cost of labeling o-sequence $O_{1:T}$ by s-sequence $S_{1:T}$

$$C(S_{1:T}|O_{1:T}) = \sum_{1 \leq i \leq T} C_a(s_i|o_i) + \sum_{1 < i \leq T} C_t(s_i, s_{i-1}) \quad (1)$$

with real-valued *atemporal-cost* and *transition-cost* functions C_a and C_t . This is the cost model implicit in (conditional) hidden Markov models (HMMs). $C_a(s_i|o_i)$ is the cost of labeling o_i by state s_i , and $C_t(s_i, s_{i-1})$ is the cost of transitioning from state s_{i-1} to s_i . As in [Lafferty et al., 2001], our work extends to C_a ’s with “non-local” observation dependencies. Sequential inference involves computing $\arg \min_{S_{1:T}} C(S_{1:T}|O_{1:T})$ for a given $O_{1:T}$.

$C(S_{1:T}|O_{1:T})$ is a *simple-transition model (STM)* with parameters $\langle C_a, K \rangle$ if the atemporal-cost function is C_a and $C_t(s_i, s_{i-1}) = K \cdot \delta(s_i \neq s_{i-1})$, where K is a real, and $\delta(p) = 1$ if p is true else 0. This model charges an equal cost for all state transitions, even though generally some transitions types are more likely than others. If this likelihood information is critical for inference, then an STM alone will not suffice. However, we believe that there are interesting processes where accurate inference does not require exploiting non-simple transition structure. For example, in our video domain, states persist for many observations and can be reliably inferred by integrating just those observations, without considering transition type. For such processes, it is important to study simple but sufficient models, as there can be considerable computational advantages.

4 Inference for Simple-Transition Models

Viewing an STM as an HMM we can apply the Viterbi algorithm [Forney, 1973] to compute a minimum-cost s-sequence $S_{1:T}$ in $O(T \cdot |\mathcal{Y}|^2)$ time. In fact, by leveraging the simple-transition structure, Viterbi can be improved to $O(T \cdot 2 \cdot |\mathcal{Y}|)$ time. However, these algorithms are impractical for us since they are exponential in the number of state variables n (i.e. $|\mathcal{Y}| = |D_s|^n$), which will typically be large for our relational processes. Likewise, for STMs, general-purpose

graphical-modeling techniques such as variable elimination and junction-tree algorithms are exponential in n (i.e. the induced tree width [Dechter, 1999] is linear in n).

Reduction to SSM. Sequential inference for an STM $\langle K, C_a \rangle$ reduces to the *single-state minimization (SSM)* problem, which is defined as computing the SSM function $\sigma(O_{1:j}) = \min_s \sum_{1 \leq i \leq j} C_a(s|o_i)$. The SSM function gives the minimum cost of labeling $O_{1:j}$ by a single state (i.e. no state transitions). For an o-sequence $O_{1:T}$ we denote the minimum cost over all s-sequences as $C^*(O_{1:T}) = \min_{S_{1:T}} C(S_{1:T}|O_{1:T})$. For STMs, $C^*(O_{1:T})$ can be expressed in terms of σ as follows (for proof see Fern [2004]).

$$C^*(O_{1:T}) = \min_{0 \leq j < T} [C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})] \quad (2)$$

Equation 2 minimizes over j , where $j + 1$ is viewed as the final state-transition point. The minimum cost over s-sequences with final transition $j + 1$ equals the minimum-cost sequence up to j , plus a transition cost K (unless $j = 0$), plus the SSM cost for the remaining suffix (no transitions occur after j). This decomposition is possible because STMs weigh all transition types equally, decoupling the minimization problems before and after j .

Equation 2 yields an efficient reduction to SSM based on dynamic programming, which we call the *SSM-DP algorithm*. Simply compute $C^*(O_{1:j})$ in the order $j = 1, 2, \dots, T$ for a total of T^2 SSM computations. It is straightforward to store information for extracting the minimum-cost s-sequence, which we denote by $\text{SSM-DP}(O_{1:T}, \sigma, K)$. Thus, efficient SSM computation implies efficient sequential inference for STMs. The complexity of SSM depends on the particular representation used for C_a . In Section 5, we leverage nearly-sound logical constraints to provide practically efficient SSM.

Though SSM-DP provides the potential for handling large state spaces, a naive implementation requires T^2 SSM computations, which is often unacceptable. However, significant and sound pruning of the computation is possible—e.g. in Equation 2, ignore any j when a k exists such that $C(S_{1:k}|O_{1:k}) \leq C(S_{1:j}|O_{1:j}) + \sigma(O_{j+1:k})$. In our application, pruning reduces the empirical number of SSM computations to $O(T)$. See Fern [2004] for details.

Sufficient SSM Computation. In practice we do not need exact SSM for all o-sequences. Rather we need only to compute a sufficient SSM approximation defined below. $O_{1:j}$ is a *critical o-sequence* of process $\mathcal{P}$ if for some o-sequence drawn from $\mathcal{P}$, $O_{1:j}$ is a maximal subsequence generated by a single state. Let σ be the SSM function for an STM

that accurately predicts s-sequences of $\mathcal{P}$. We say σ' is a *sufficient SSM approximation* to σ for $\mathcal{P}$ if both 1) for all o-sequences $O_{1:T}$, $\sigma(O_{1:T}) \leq \sigma'(O_{1:T})$, and 2) $\sigma'(O') = \sigma(O')$ for any critical o-sequence O' of $\mathcal{P}$. It can be shown that for o-sequences drawn from $\mathcal{P}$, $\text{SSM-DP}(O_{1:T}, \sigma, K) = \text{SSM-DP}(O_{1:T}, \sigma', K)$. Thus we can achieve accurate inference using σ' . In Section 5 we show how to efficiently compute such an approximation.

Non-Simple Models. Given our focus on SSM it is natural to consider efficient SSM reductions for non-simple models. Intuitively, if a model distinguishes between different transition types, we may need to consider states other than just SSM solutions (like Viterbi but unlike SSM-DP), possibly resulting in exponential behavior in n even given efficient SSM. Below we show that an efficient reduction is unlikely for a modest extension to STMs, giving a boundary between efficient and inefficient models under efficient SSM.

We extend STMs by allowing the model to assign higher costs to transitions where more state variables change, unlike STMs which can only detect whether a change occurred. A cost model C , with atemporal component C_a , is a *counting-transition model* if $C_t(s_i, s_{i-1}) = K \cdot \sum_{1 \leq j \leq n} \delta(s_i^{(j)} \neq s_{i-1}^{(j)})$, i.e. transition cost is linear in the count of propositions that change. We say C *allows efficient SSM* if the SSM function for C_a is computable in time polynomial in its input o-sequence size and number of state variables.

Theorem 1. *Given as input a counting-transition model C , an observation sequence O , and a cost bound τ , the problem of deciding whether $C^*(O_{1:T}) < \tau$ is NP-complete, even under the assumption that C allows efficient SSM.*

Proof: (sketch) See Fern [2004] for full proof. Inference in 2-d grid Potts models, a class of Markov random fields, is NP-hard [Veksler, 1999]. We first give a non-trivial extension of this result to the smaller class of 2-d grid Potts models with equally weighted “horizontal edges”. Next, we reduce this problem to counting-transition model inference. Intuitively, the state variables at time i represent the Potts model nodes in column i . Finally, we show that the constructed model allows efficient SSM via variable elimination. $\square$

5 Constraint-Based SSM

We now give a constraint-based representation for atemporal-cost functions and study the corresponding SSM problem. For simplicity, we assume that states have n binary state variables, i.e. $D_s = \{\text{true}, \text{false}\}$. We also assume a set of m binary *observation tests*, each one mapping observations to $\{\text{true}, \text{false}\}$. Non-binary extensions are straightforward.

Propositional Horn Constraints. A *propositional Horn constraint* ϕ is a logical implication (*body* $\rightarrow$ *head*), where *body* is a conjunction of state variables and observation tests, and *head* is a state variable or **false**. Given an observation o and state s , we let $\phi[o]$ ($\phi[s]$) denote the result of substituting observation tests (state variables) in ϕ with the truth values under o (under s). If a constraint has no variables, then it is interpreted as the truth value of the variable-free expression. Thus, $\phi[o][s]$ is the truth value of ϕ under o and s , and we say that o and s *satisfy* ϕ iff $\phi[o][s]$ is **true**. A set of Horn constraints is *satisfiable* iff there exists a state and observation that jointly satisfy each constraint. For horn constraints, testing satisfiability and finding satisfying assignments are polynomial-time computable [Papadimitriou, 1995].

Constraint-Based Cost Functions. We parameterize an atemporal-cost function using a set of weighted Horn constraints $\Phi = \{\langle \phi_1, c_1 \rangle, \dots, \langle \phi_v, c_v \rangle\}$, with Horn constraints ϕ_i and non-negative weights c_i representing the cost of violating ϕ_i . The non-negativity requirement will be important for inference. The sum of costs in Φ is denoted by $\text{COST}(\Phi)$ and we say Φ is *satisfiable* if its set of constraints is satisfiable. Atemporal cost is defined as $C_a(s|o, \Phi) = \sum_{\langle \phi, c \rangle \in \Phi} c \cdot \delta(\neg \phi[o][s])$, i.e. the total cost of unsatisfied constraints. In this work, we will assume that Φ contains “nearly sound” constraints, meaning that each constraint is usually satisfied by the state/observation pairs generated by our process. Our primarily non-theoretical goals do not require a formal notion of nearly sound (e.g. PAC).

Given Φ and an o-sequence $O_{1:j}$, we define the *combined constraint set* as $\Gamma(O_{1:j}, \Phi) = \bigcup_{1 \leq i \leq j} \bigcup_{\langle \phi, c \rangle \in \Phi} \langle \phi[o_i], c \rangle$, which involves only state variables and captures all of the state constraints “implied” by Φ and $O_{1:j}$. The SSM function for C_a is now given by

$$\sigma(O_{1:j}|\Phi) = \min_s \sum_{1 \leq i \leq j} C_a(s|o_i, \Phi) \quad (3)$$

$$= \min_s \sum_{\langle \phi, c \rangle \in \Gamma(O_{1:j}, \Phi)} c \cdot \delta(\neg \phi[s]) \quad (4)$$

which is equivalent to solving *maximum satisfiability* (MAX-SAT) [Jiang et al., 1995] for $\Gamma(O_{1:j}, \Phi)$, where MAX-SAT asks for an s such that the weight of satisfied constraints is maximum. While the number of SSM variables is fixed, the constraint set grows with sequence length. Fortunately, in practice we can significantly reduce this set by pruning and merging. $\text{PRUNE}(\Phi)$ contains members of Φ that do not have **false** in the constraint’s body. $\text{MERGE}(\Phi)$ combines logically equivalent members of Φ into one constraint by summing weights. MAX-SAT solutions are invariant under both operators. Thus, we solve SSM via MAX-SAT for the smaller constraint $\Gamma^*(O_{1:j}, \Phi) = \text{MERGE}(\text{PRUNE}(\Gamma(O_{1:j}, \Phi)))$.

A Dual MAX-SAT Approach. MAX-SAT is NP-hard even for Horn constraints [Jaumard and Simeone, 1987]. Rather than use general-purpose approximate techniques, we give a MAX-SAT approach that leverages our setting of nearly-sound Horn constraints. Let $\pi = \Gamma^*(O_{1:j}, \Phi)$ and Π be the set containing all satisfiable subsets of π . An equivalent dual form of Equation 4 is $\sigma(O_{1:j}|\Phi) = \text{COST}(\pi) - \max_{\pi' \in \Pi} \text{COST}(\pi')$, which asks for a maximum-cost member of Π . This motivates the following MAX-SAT approach that searches through constraint subsets rather than variable assignments. Conduct a cost-sensitive breadth-first search through subsets of π for a satisfiable subset—i.e. starting at π consider π subsets in order of non-increasing cost until finding a satisfiable one. Any satisfying assignment for this set is a MAX-SAT solution provided all weights are non-negative. For negative weights a satisfying assignment for a consistent constraint set may not be a MAX-SAT solution. Although we can always replace a negatively weighted constraint $\langle \phi, c \rangle$ by $\langle \neg \phi, -c \rangle$, $\neg \phi$ may not be horn, possibly making SAT hard. Hence we require non-negative weights.

Since testing satisfiability is efficient for Horn constraints, the time required by the dual approach primarily depends on the number of search nodes we consider. This number is

Table 1: Force-dynamic state predicates R_s (top) and observation predicates R_o (bottom) for our application.

ATTACHED(x, y)	x supports y by attachment
GROUNDED(x)	support of x is unknown
CONTACTS(x, y)	x supports y by contact
DIRECTION(x, d)	x is moving in direction d
SPEED(x, s)	x 's speed is s
ELEVATION(x, e)	x 's elevation is e
MORPH(x, c)	x 's shape-change factor is c
DISTANCE(x, y, d)	distance between x and y is d
Δ DIST(x, y, dd)	change in distance is dd
COMPASS(x, y, c)	compass direction of y to x is c
ANGLE(x, y, a)	angle between x and y is a

bounded by the number of subsets of π that have a cost greater than $\sigma(O_{1:j}|\Phi)$, which can be exponentially large. Thus, we compute an approximation σ^τ to σ by first searching through subsets of π for a maximum of τ steps. We return a solution if one is found and otherwise return an upper-bound to $\sigma(O_{1:j}|\Phi)$ resulting from a greedy hill-climbing search.

Though σ^τ will not be correct for all inputs, we know from Section 4, that it need only be a sufficient approximation to guarantee correct sequential inference. When our constraints are nearly sound, σ^τ will tend to be sufficient even for small τ . That is, σ^τ will equal σ for critical o-sequences. Recall that when $O_{1:j}$ is a critical o-sequence it must be generated by a single state s . Thus, s will satisfy most constraints in $\Gamma^*(O_{1:j}, \Phi)$ and our search need only remove a small number of constraints (the unsatisfied ones) to find a satisfiable subset. In addition, as described in the full paper, the MERGE operation tends to place high weight on the satisfied constraints, which often leads in removing unsatisfied constraints first.

6 Extending to Relational Processes

In the spirit of knowledge-base model construction [Wellman *et al.*, 1992], we extend to relational processes by compiling “relational schemas” to propositional models and use the ideas from previous sections.

Relational Processes. A process $(\mathcal{X}, \mathcal{Y}, \mathcal{P})$ is *relational* when the observation and state spaces $\mathcal{X}$ and $\mathcal{Y}$ are given by specifying a domain set of objects D , a state-predicate set R_s , and an observation-feature set F_o , each having a specified number of arguments. An *observation fact* has the form “ $f = v$ ”, where f is an observation feature applied to objects and v is a number. A *state fact* is a predicate from R_s applied to objects. See Example 1 for example facts from our video domain. Observations (states) are finite sets of observation (state) facts, representing all the facts that are true, and $\mathcal{X}$ ($\mathcal{Y}$) contains all such sets. States are restricted to only involve objects that appear in the corresponding observation. We often view relational states as propositional. Given a finite $D' \subseteq D$, denote by $\mathcal{Y}[D']$ the propositional state space over n binary variables, one variable for each of the $n = O(|D'|^q)$ state facts involving only objects in D' , where q is the maximum state-predicate arity.

Example 2. In our video domain we infer force-dynamic state sequences from videos of a hand playing with blocks. D contains all hands and blocks we might encounter. There are three force-dynamic state predicates and eight observation features, shown in Table 1. Figure 1 depicts two distinct force-dynamic states. Observations are sets of observation facts calculated for the objects and object pairs based on the

PERCEPTRON($\text{TRN}, \{\phi_1, \dots, \phi_v\}, \tau, M$)

$K \leftarrow 0; \vec{C} = \vec{0};$

repeat M times,

for-each $\langle O, S \rangle \in \text{TRN}$

$\Phi \leftarrow \{\langle \phi_1, c_1 \rangle, \dots, \langle \phi_v, c_v \rangle\}$

$\hat{S} \leftarrow \text{SSM-DP}(O, \sigma_r^\tau(\cdot|\Phi), K)$

if $\hat{S} \neq S,$

$\vec{C} \leftarrow [\vec{C} + \vec{V}(\hat{S}, O) - \vec{V}(S, O)]^+$

$K \leftarrow [K + \text{TRANS}(\hat{S}) - \text{TRANS}(S)]^+$

return $\langle \vec{C}, K \rangle$

Figure 2: Generalized Perceptron Pseudocode. $[\vec{C}]^+ = \vec{C}'$ s.t. $c'_i = \max(0, c_i)$.

object tracker output.

Relational Horn Constraints. A *state atom* is a state predicate or the relation “ $\neq$ ” applied to variables. An *observation atom* has the form “ $(f_1 \ r \ f_2)$ ”, where $r \in \{=, \leq\}$, and f_i is a number or an observation feature applied variables. A *relational Horn constraint* has the form $(\text{body} \rightarrow \text{head})$, where *body* is a conjunction of state and/or observation atoms, and *head* is a state atom or **false** and may only contain variables that appear in *body*. For example, $(\text{DISTANCE}(x, y) \leq 5) \wedge (6 \leq \text{Speed}(y)) \rightarrow \text{ATTACHED}(x, y)$ is a relational Horn constraint for predicting object attachment based on an observation. A relational Horn constraint ϕ is a schema for propositional constraints. Any way of (consistently) replacing variables in ϕ with objects gives a (propositional) *ground instance* of ϕ . Given a set of objects D' , $\text{GROUND}(\phi, D')$ contains all ground instances with only objects in D' .

Relational Cost Models. A *relational simple-transition model* is a pair $\langle \Phi, K \rangle$, with transition-cost K and non-negatively weighted relational Horn constraints $\Phi = \{\langle \phi_1, c_1 \rangle, \dots, \langle \phi_v, c_v \rangle\}$. Given a relational o-sequence $O_{1:T}$ with objects D' , we know that states may only involve facts constructed from D' , and thus we need only consider the propositional state space $\mathcal{Y}[D']$. To infer an s-sequence over $\mathcal{Y}[D']$ we use the set of propositional constraints $\Phi_p = \bigcup_{(\phi, c) \in \Phi} \bigcup_{\phi' \in \text{GROUND}(\phi, D')} \langle \phi', c \rangle$ to define an atemporal cost function $C_a(s|o, \Phi_p)$ as in Section 5. We then return the lowest-cost $S_{1:T}$ given by $\text{SSM-DP}(O_{1:T}, \sigma_r^\tau(\cdot|\Phi), K)$, where $\sigma_r^\tau(O_{1:j}|\Phi) = \sigma^\tau(O_{1:j}|\Phi_p)$ is a relational SSM function with search bound τ . That is σ_r^τ is computed by compilation to a propositional SSM function σ^τ and then using our bounded-search dual MAX-SAT approach.

A naive implementation of this approach can be expensive since Φ_p can be large. Fortunately, in practice, our relational representation allows us to avoid constructing most of the set. We use efficient forward-chaining logical inference to construct $\Gamma^*(O_{1:T}, \Phi_p)$, the input to MAX-SAT, without explicitly constructing Φ_p . Space constraints preclude details.

7 Learning a Relational STM

We learn relational STMs by first running the relational learner CLAUDIEN [De Raedt and Dehaspe, 1997] on a training set of labeled o-sequences, acquiring “nearly sound” relational Horn constraints that are satisfied by the training data (producing 300-400 constraints for our domain). Next, using a new training set we jointly tune the constraint weights and transition-cost K using a variant of Collins’ voted percep-

tron algorithm [Collins, 2002]. The algorithm extends Rosenblatt’s perceptron for binary labels [Rosenblatt, 1958] to handle structured labels such as sequences, and has convergence and generalization properties similar to Rosenblatt’s.

The algorithm requires representing cost using a linear combination of n features, which we do as follows. Given $O_{1:T}$ and $S_{1:T}$ involving just objects in the finite D' , and a relational STM $\{\langle\phi_1, c_1\rangle, \dots, \langle\phi_v, c_v\rangle\}, K$, the *violation-count feature* of ϕ_i is $V_i(O_{1:T}, S_{1:T}) = \sum_{1 \leq i \leq T} \sum_{\phi' \in \text{GROUND}(\phi_i, D')} \delta(-\phi'[o_i][s_i])$, i.e. the number of unsatisfied instances of ϕ_i . We let $\vec{V}(O_{1:T}, S_{1:T})$ be the v -dimensional vector of violation-count features and $\vec{C} = [c_1, \dots, c_v]$ is the weight vector. The *transition count feature* $\text{TRANS}(S_{1:T})$ is equal to the number of state transitions in $S_{1:T}$. These $v + 1$ features represent the STM cost as $C(S_{1:T}|O_{1:T}) = \vec{V}(O_{1:T}, S_{1:T}) \cdot \vec{C} + \text{TRANS}(S_{1:T}) \cdot K$.

The algorithm (see Figure 2) cycles through the training data and when an incorrect s-sequence $\hat{S}$ is inferred in place of S , the weights are adjusted to increase cost for $\hat{S}$ and decrease cost for S . The input is a training set TRN, relational Horn constraints $\{\phi_1, \dots, \phi_v\}$, an SSM search bound τ , and the number M of iterations. The output is the learned weights $\vec{C}$ and transition cost K . Ideally we want this STM to allow for accurate sequential inference when using the search-bounded relational SSM function σ_τ^τ . Unlike Collins we require non-negative weights for inference. Thus, we set a weight to zero if a normal perceptron update would result in a negative value. This variant has not yet been shown to possess the convergence and generalization properties of the unconstrained version. However, this variant of Rosenblatt’s perceptron has been shown to converge [Amit *et al.*, 1989].

8 Experimental Results

We apply our technique to force-dynamic state inference from real video. The LEONARD system [Siskind, 2001] uses these states to recognize events, such as “a hand picked up a block” (see Figure 1). Recently [Fern and Givan, 2004] developed a trainable system for this problem using the *forward greedy merge (FGM)* algorithm, which outperformed prior techniques. FGM also utilizes Horn constraints, but assumes that they are sound, rather than nearly sound. Since this is not true for CLAUDIEN-learned constraints, which were also used by FGM, that work utilized two steps to improve performance: 1) *Constraint pruning* yields a much smaller but (hopefully) “sufficient” constraint set, reducing the chance of constraint violations. 2) *Sequence-cleaning preprocessing* is an ad-hoc step that removes observations where constraint violations are “detected”. See [Fern and Givan, 2004] for details. While these steps allowed for good performance, the soundness assumption limits the approach’s applicability, as such preprocessing will not always be effective. The motivation for the work in this paper was to develop a “softened” more robust framework for utilizing nearly-sound constraints.

Procedure. Our corpus of 210 videos from [Siskind, 2003] contains 7 event types (30 movies each) involving a hand playing with up to three blocks (e.g. assembling a tower). We use the hand-labeled training sets of size 14 (TRN-14) and 21 (TRN-21) from [Fern and Givan, 2004]. The remaining videos are labeled with their compressed s-sequences (the output of COMPRESS), which is the label we wish to predict.

For each training set, we learn relational STMs for three SSM search bounds $\tau = 0, 100, 1000$. Seven training instances are used by CLAUDIEN to acquire constraints and the perceptron algorithm is run for 100 iterations on the remaining instances. We evaluate each iteration’s model according to the % of test videos for which SSM-DP’s compressed inferred s-sequence is correct (using the same τ as for learning).

We compare against FGM used with and without sequence cleaning and always with pruning. We also compare against a system that is closer to mainstream graphical modeling techniques, which uses the counting-transition model of Section 4, and WALKMAXSAT [Jiang *et al.*, 1995] for approximate inference. The model is identical to our relational STMs, using the same CLAUDIEN-learned constraints, except that it has a non-simple transition structure. For sequential inference we construct a single large MAX-SAT problem corresponding to the transition-counting model and use WALKMAXSAT to find an approximate solution. We tune the model’s weights using the perceptron algorithm with WALKMAXSAT (rather than SSM-DP) for inference.

Performance Across Iterations. Table 2 shows, for each training set, the testing error of our learned STMs for $\tau = 0, 100, 1000$ (shown as SSM-DP(τ)), over the first eight perceptron iterations. There is always a rapid improvement after iteration one, followed by a period of fluctuating or constant performance. The fluctuation continues out to iteration 100 (not shown), but never improves over the best performance shown. In practice, one could use cross-validation to select a good model, or consider “weight averaging” [Collins, 2002]. Note that the best performance on the smaller training set TRN-14 is superior to TRN-21. Our explanation for this is that by using relatively small training sets, the results can be significantly affected by a small number of particularly noisy movies (presumably in TRN-21). Experiments not shown, provide evidence for this by training on subsets of TRN-21.¹

Bounding Search. The search bound $\tau = 0$ means that SSM is (approximately) solved via hill-climbing. Surprisingly we can learn weights for which hill-climbing performs well. Increasing the search bound improves performance with respect to the best model across iterations, particularly for TRN-14. The last column shows the frames processed per second when inferring states for our corpus. Inference time apparently does not increase linearly with τ , indicating that the SSM search typically end much before reaching the bound. Increasing τ to 10,000, doubles inference time, but neither improves nor hurts results.

Other Techniques. We significantly outperform WALKMAXSAT (further iterations did not help), which we allowed a generous search time, using search cutoff 10^6 with 10^3 random restarts (other settings did not improve). Further experiments suggest that WALKMAXSAT does not scale well for our problems. Using learned STM weights for the counting-transition model, WALKMAXSAT reliably infers correct s-sequences for short videos, but is very poor for long videos. WALKMAXSAT gave equally poor results when used to learn

¹Runtime of CLAUDIEN (interpreted Prolog) on TRN-21 was about six days. Thus, it was not feasible to average results across many training sets. In another full experiment (21 movies), we observed similar (slightly better) performance for SSM-DP, and similar relative performance to the other systems.

Table 2: % error on test movies across training iterations. The last column gives frames processed per second (FPS) by the best model in each row on our video corpus.

	TRN-14 (Iteration)								TRN-21 (Iteration)								FPS
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	
SSM-DP(0)	94	24	3.2	2.1	2.1	2.6	2.1	2.6	94	6.3	6.9	5.8	4.8	12	17	3.7	15
SSM-DP(100)	93	32	2.6	2.6	2.1	2.1	2.1	2.1	93	3.2	3.7	3.7	3.2	10	21	8.5	14
SSM-DP(1000)	93	31	4.2	1.6	1.1	1.1	1.1	1.1	93	3.2	3.7	3.7	3.2	10	21	8.5	8
WALKMAXSAT	96	46	51	54	50	43	50	58	95	47	50	60	49	50	45	52	1.4
FGM / FGM+SC	17 / 6.3								15 / 3.2								29

STMs rather than counting-transition models. This suggests the poor performance is primarily due to the ineffectiveness of this general-purpose inference technique for our models. We conjecture that other approximate techniques such as (loopy) belief propagation and Gibbs sampling will also yield inferior performance. Indeed, Gibbs sampling is very similar to WALKMAXSAT. Evaluating this conjecture is future work.

FGM without sequence cleaning (shown as FGM) is an order of magnitude worse than STMs (which never use cleaning). We significantly improve on FGM with sequence cleaning (FGM+SC) on TRN-14 and yield equal performance on TRN-21. FGM is faster, close to frame rate. A reimplementa-tion of our LISP prototype will likely achieve frame rate.

9 Related Work

Segment models [Ostendorf *et al.*, 1996] subsume our STM formulation. We are not aware of prior work that has leveraged or noted the SSM reduction for large state spaces. Perhaps this is because prior segment modeling work typically utilizes non-simple transition structure (perhaps sometimes unnecessarily) and to our knowledge has not been applied to large state spaces.

Our “model schema” approach for handling relational data is now standard in probabilistic modeling [De Raedt and Kersting, 2004]. Most closely related are relational Markov networks (RMNs) [Taskar *et al.*, 2002]. A relational STM can be viewed as defining a log-linear conditional RMN, where the RMN “clique feature templates” correspond to relational Horn constraints and a transition template. RMNs are very general and thus the existing techniques do not fully exploit the structure of relational STMs. Instead RMN-like proposals rely on general-purpose approximate inference (e.g. belief propagation), with unclear practical implications. Likewise dynamic probabilistic relational models [Sanghai *et al.*, 2003] provide a generic schema-based approach specialized to relational sequence data. Again the generality precludes leveraging the STM structure.

Acknowledgments

This work was supported in part by NSF grants 9977981-IIS and 0093100-IIS.

References

[Amit *et al.*, 1989] D. Amit, K. Wong, and C. Campbell. Perceptron learning with sign-constrained weights. *Journal of Physics A: Mathematical and General*, 22:2039–2045, 1989.

[Collins, 2002] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm. In *Conf. on Empirical Methods in NLP*, 2002.

[De Raedt and Dehaspe, 1997] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

[De Raedt and Kersting, 2004] L. De Raedt and K. Kersting. Probabilistic logic learning. *ACM-SIGKDD Explor.*, 5, 2004.

[Dechter, 1999] R. Dechter. Bucket elimination: A unifying framework for reasoning. *AIJ*, 113(1-2), 1999.

[Fern and Givan, 2004] A. Fern and R. Givan. Relational sequential inference with reliable observations. In *ICML*, 2004.

[Fern, 2004] Alan Fern. *Learning Models and Formulas of a Temporal Event Logic*. PhD thesis, Purdue University, August 2004.

[Forney, 1973] G. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[Jaumard and Simeone, 1987] B. Jaumard and B. Simeone. On the complexity of the maximum satisfiability problem for Horn formulas. *Information Processing Letters*, 26:1–4, 1987.

[Jiang *et al.*, 1995] Y. Jiang, H. Kautz, and B. Selman. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *Joint Workshop on AI and OR*, 1995.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[Murphy, 2002] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science, July 2002.

[Ostendorf *et al.*, 1996] M. Ostendorf, V. Digalakis, and O. Kimball. From HMMs to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Trans. on Speech and Audio Proc.*, 1996.

[Papadimitriou, 1995] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing, 1995.

[Rosenblatt, 1958] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Sanghai *et al.*, 2003] S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *IJCAI*, 2003.

[Siskind, 2001] J. Siskind. Grounding lexical semantics of verbs in visual perception using force dynamics and event logic. *JAIR*, 15:31–90, 2001.

[Siskind, 2003] J. Siskind. Reconstructing force-dynamic models from video sequences. *AIJ*, 151(1-2):91–154, 2003.

[Taskar *et al.*, 2002] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.

[Veksler, 1999] O. Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, Computer Science, July 1999.

[Wellman *et al.*, 1992] M. Wellman, J. Breese, and R. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 5, 1992.

Repairing Concavities in ROC Curves

Peter A. Flach

Department of Computer Science
University of Bristol
Bristol, United Kingdom
Peter.Flach@bristol.ac.uk

Shaomin Wu

School of Construction Management
and Engineering, University of Reading
Reading, United Kingdom
Shaomin.Wu@reading.ac.uk

Abstract

In this paper we investigate methods to detect and repair concavities in ROC curves by manipulating model predictions. The basic idea is that, if a point or a set of points lies below the line spanned by two other points in ROC space, we can use this information to repair the concavity. This effectively builds a hybrid model combining the two better models with an inversion of the poorer models; in the case of ranking classifiers, it means that certain intervals of the scores are identified as unreliable and candidates for inversion. We report very encouraging results on 23 UCI data sets, particularly for naive Bayes where the use of two validation folds yielded significant improvements on more than half of them, with only one loss.

1 Introduction

There is an increasing amount of work on model selection and model combination in the machine learning and data mining literature: for instance, model selection based on ROC space [Provost and Fawcett, 2001], model combination by means of bagging [Breiman, 1996], boosting [Freund and Schapire, 1996], arcing [Breiman, 1998], the mixture of experts method [Jacobs *et al.*, 1991], to name just a few. A review on ensembles of learning machines can be found in [Valentini and Masulli, 2002].

Typically, these methods assume a set of given models with fixed performance, and the issue is how best to combine these models to obtain a better ensemble model. There is no attempt to analyse the performance of the given models to determine a region where performance is sub-standard. This paper investigates methods to improve given models using ROC analysis.

ROC (Receiver Operating Characteristic) analysis is usually associated with classifier selection when both class and misclassification cost distribution are unknown at training time. However, ROC analysis has a much broader scope that is not limited to cost-sensitive classification. A categorical classifier is mapped to a point in ROC space by means of its false positive rate on the X-axis and its true positive rate on the Y-axis. A probabilistic classifier results in a ROC

curve, which aggregates its behaviour for all possible decision thresholds. The quality of a probabilistic classifier can be measured by the Area Under the ROC Curve (AUC), which measures how well the classifier separates the two classes without reference to a decision threshold. A good classifier should have a large AUC, and $AUC=1$ means that there is a decision threshold such that the corresponding categorical classifier has 100% accuracy.

We use the term *model repair* to denote approaches that modify given models in order to obtain better models. In contrast, ensemble methods produce hybrid models that leave the original models intact. An approach to model construction using ROC space is given in [Blockeel and Struyf, 2002], where the authors identify and assemble parts of a decision tree that perform well in different areas of ROC space. Our approach in this paper is to identify ‘bad’ areas, or concavities, in a ROC curve and repair them by manipulating the corresponding low-quality predictions. The approach is experimentally validated using both naive Bayes and decision tree, but the approach has much wider scope as it can be applied to any classifier that computes class scores.

To illustrate the approach, we describe in Section 2 the **RepairPoint** algorithm that combines three models based on different thresholds of the same probabilistic classifier, and creates a new model which theoretically should improve upon the worst of the three models. In Section 3 we introduce the main algorithm **RepairSection**, that mirrors an entire concave region (a region of the curve that is below its convex hull). In Section 4 we present experimental results on 23 data sets from the UCI repository. Section 5 reviews some related work on model ensembles, gives the main conclusion and suggests further work.

2 Basics of repairing classifiers in ROC space

Assume that the confusion matrix of a classifier evaluated on a test set is as in Table 1. Then the true positive rate of the classifier is $a/(a+b)$ and the false positive rate of the classifier is $c/(c+d)$. The point $(c/(c+d), a/(a+b))$ in the XY plane (i.e., ROC space) will be used to represent the performance of this classifier.

If a model is under the ascending diagonal in ROC space, this means that it performs worse than random. Models A and B in Figure 1 are such worse-than-random models.

	predicted positive	predicted negative
actual positive	a	b
actual negative	c	d

Table 1: A confusion matrix.

However, there is a very useful trick to obtain better-than-random models: simply invert all predictions of the original model. This corresponds to exchanging the columns in the contingency table, leading to a new true positive rate of $b/(a+b) = 1 - a/(a+b)$, i.e. one minus the original true positive rate; similarly we obtain a new false positive rate of $d/(c+d) = 1 - c/(c+d)$. Geometrically, this corresponds to mirroring the original ROC point through the midpoint on the ascending diagonal.

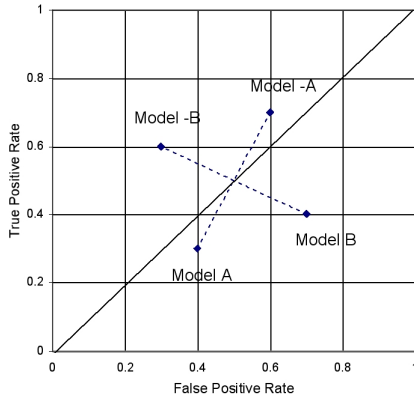


Figure 1: By inverting their predictions, worse-than-random models A and B below the diagonal can be transformed into better-than-random models -A and -B above the diagonal.

Notice that the ascending diagonal really connects two classifiers: the classifier which always predicts negative in (0,0), and the classifier which always predicts positive in (1,1). This suggests that the above repair procedure can be generalised to line segments connecting arbitrary classifiers. For instance, consider Figure 2. Denote the sets of true and false positives of Model i by TP_i and FP_i , then we can construct Model 4 under the condition that $TP_1 \subseteq TP_3 \subseteq TP_2$ and $FP_1 \subseteq FP_3 \subseteq FP_2$. In particular, these *inclusion constraints* are satisfied if Models 1, 2 and 3 are obtained by setting thresholds on the same probabilistic model, which is what we assume throughout the paper.

Model 4 operates as indicated in Table 2. The inclusion constraints guarantee that the geometric configuration of Figure 2 holds. This is formally stated in the following theorem.

Theorem 1 *Assuming that $TP_1 \subseteq TP_3 \subseteq TP_2$ and $FP_1 \subseteq FP_3 \subseteq FP_2$, the model produced by Algorithm RepairPoint has true and false positive rates $TPr_4 = TPr_1 + TPr_2 - TPr_3$ and $FPr_4 = FPr_1 + FPr_2 - FPr_3$, where TPr_i and FPr_i denote true and false positive rates of Model i .*

Proof. Under the inclusion constraints expressed in the theorem, there are four disjoint groups of examples: those classified positive by Models 1, 3 and 2 ($TP_1 \cup FP_1$); those clas-

Given three models Model 1, Model 2 and Model 3, output Model 4 that operates as follows:

1. If both Model 1 and Model 2 predict negative, then predict negative;
2. If both Model 1 and Model 2 predict positive, then predict positive;
3. If Model 1 predicts negative and Model 2 predicts positive, then predict the opposite of what Model 3 predicts;
4. Otherwise, predict what Model 3 predicts.

Table 2: Algorithm RepairPoint. The last clause does not apply under the inclusion constraints and is only added for completeness.

sified negative by Model 1 and positive by Models 3 and 2 ($(TP_3 - TP_1) \cup (FP_3 - FP_1)$); those classified negative by Models 1 and 3 and positive by Model 2 ($(TP_2 - TP_3) \cup (FP_2 - FP_3)$); and those classified negative by all three models ($(POS - TP_2) \cup (NEG - FP_2)$, where POS and NEG are the sets of all positive and all negative examples, respectively). By construction, Model 4 classifies the first group as positive, the second group as negative, the third group as positive, and the fourth group as negative. The true positives of Model 4 are thus $TP_1 \cup (TP_2 - TP_3)$; because of the inclusion constraints the result follows (analogous for false positives).

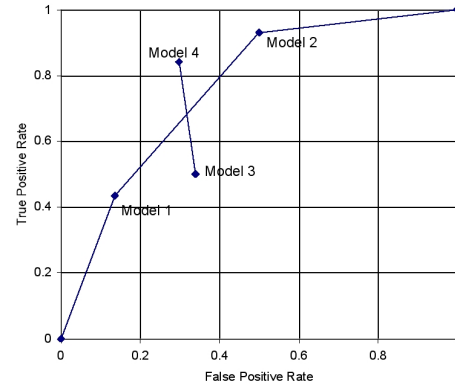


Figure 2: Model 3 is mirrored to Model 4 with the help of Models 1 and 2.

An equivalent construction is the following. Remove from the test set all instances classified positive by Model 1, and all instances classified negative by Model 2. We can imagine this as a smaller nested ROC space in which Model 1 represents (0,0) and Model 2 represents (1,1); because of the inclusion constraints the position of Model 3 remains unchanged. Note that Model 3 performs worse than a random model in this nested ROC space. We then construct Model 4 as in Figure 1, by inverting the predictions of Model 3 on the remaining test examples.

We end this section by noting that the true and false positive rates derived for Model 4 only hold for the same test set on which Models 1, 2 and 3 were evaluated. On a second, independent test set the ROC locations of the 4 classifiers may be different – in particular, Model 3 may not be below the

line connecting Models 1 and 2, in which case Model 4 will be evaluated worse than Model 3. In our experiments, we therefore use validation sets to decide whether concavities are stable across different samples. This will be further discussed in Section 4.

3 Identifying and repairing concavities in a ROC curve

The previous section outlined the main ideas underlying repairing concavities in ROC curves. However, preliminary experiments indicated that the three-point approach is too crude to work well in practice. In this section we introduce our main algorithm, which manipulates a whole section of a ROC curve. A ROC curve is obtained by evaluating a probabilistic classifier on a test set and varying the decision threshold, resulting in a step curve [Hand and Till, 2001]. An efficient way of constructing this curve is by ranking the instances corresponding to their predicted probability of being positive [Fawcett, 2003]. Figure 3 shows both the ROC curve for a probabilistic model evaluated on a small test set¹ and its convex hull [Provost and Fawcett, 2001].

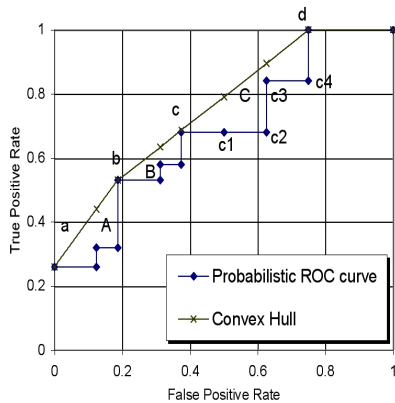


Figure 3: A probabilistic ROC curve and its convex hull.

Four points of the ROC curve (point a, b, c and d) are located on the convex hull. Each of these points corresponds to a probability threshold, and thus each segment of the convex hull corresponds to a probability interval. For instance, the convex hull in Figure 3 has three segments corresponding to three disjoint probability intervals. Three out of these five segments delineate concave regions of the ROC curve, indicated as A, B and C. For example, area A is delineated by line ab and the ROC curve between point a and point b. In general, a concave area means that the ranking obtained from the probabilistic model in this probability interval is worse than random. For instance, consider area C which is the largest concavity. One way to repair this concavity is by ignoring the

¹For illustration purposes, the test set contains only a few examples and the resolution of the ROC curve is low. In practical circumstances a step curve with much higher resolution is obtained.

score calculated by the probabilistic model in this interval, and output a constant score (e.g., the mid-point of the interval). Assuming that ties are broken by assigning a random rank, this would replace the concave region of the ROC curve with the line segment cd. It is interesting to note that if this procedure is followed for all concavities, this corresponds to constructing the convex hull by discretising the probability scores.

However, in theory we should be able to do better than that: we can invert the ranking of the instances in the probability interval, which can be seen as applying Algorithm Repair-Point to all thresholds in this interval. For instance, by applying this algorithm to the model corresponding to threshold c2, this point would be point-mirrored through the midpoint on line segment cd to the other side of the convex hull. The same can be done for the other thresholds. The resulting ROC curve is shown in Figure 4. We can note that the area C under the curve has been replaced by an equally large area C' above the curve. The AUC of the repaired curve is therefore larger than both the AUC of the original curve and the AUC of its convex hull.

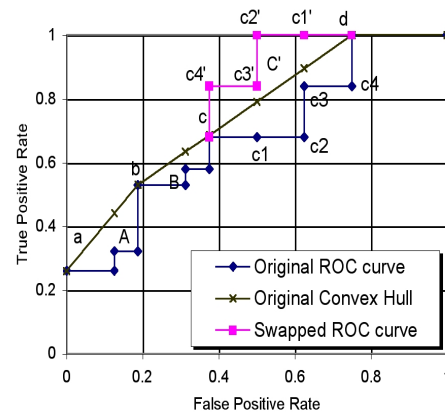


Figure 4: Mirroring a concave part of the ROC curve.

The algorithm to produce the model with the repaired ROC curve is given in Table 3. The procedure works for any model that calculates a score; a probabilistic model is a special case.

Given a scoring model M and two thresholds $T_1 > T_2$, construct a scoring model M' predicting scores as follows. Let S be the score predicted by M :

1. If $S > T_1$, then predict S ;
 2. If $S < T_2$, then predict S ;
 3. Otherwise, predict $T_1 + T_2 - S$.
-

Table 3: Algorithm RepairSection. The algorithm effectively inverts the ranking of all instances whose score as predicted by M falls in the interval $T_1 < S < T_2$.

4 Experimental evaluation

We describe a number of experiments to evaluate our approach. We used 23 two-class data sets from the UCI repository [Blake and Merz, 1998]. Table 4 shows their numbers of attributes, numbers of examples, and relative size of the majority class.

Dataset	#Attrs	#Exs	%MajClass
Australia	14	690	55.51
Sonar	60	208	51.92
Glass	9	214	67.29
German	20	1000	69.40
Car	6	1728	69.68
Anneal	38	798	74.44
Monk1	6	566	50.00
Monk2	6	601	65.72
Monk3	6	554	55.41
Hepatitis	19	155	78.71
House	16	435	62.07
Tic-tac-toe	9	958	64.20
Heart	13	270	55.56
Ionosphere	34	351	64.10
Breast Cancer	9	286	70.28
Lymphography	17	148	56.81
Primary Tumor	17	339	55.75
Soybean-large	35	683	55.51
Solar-Flare	12	323	56.35
Hayes-Roth	4	133	60.91
Credit	15	690	55.51
Balance	4	625	53.92
Bridges	12	108	66.67

Table 4: UCI data sets used in our experiments.

The experimental procedure for the first experiment was as follows. We split a data set into ten folds and use eight of them for training, one for validation and one for testing. We trained a naive Bayes model and a decision tree model² on the training data, and chose two thresholds that delineate a concavity. We then produced a new model by repairing the probabilities between the thresholds; we only use the new model if it improves AUC on the validation set. The detailed procedure is given in Table 5.

We ran Experiment RepairSection ten times and obtained m pairs of AUC values. Since we use a validation set, we are able to decide whether or not repair resulted in a better model – if not, we discard it and use the original, unrepaired model. For this reason, we only report results on the non-discarded models, so m may be smaller than 100. The average AUCs of the unrepaired curve and the repaired curve for each data set are given in Tables 6 and 7. We also performed a paired t-test with $m - 1$ degrees of freedom and level of confidence 0.1 to test the significance of the average difference in AUC. The results are favourable: the significance tests yield 10 wins and 3 losses for repaired naive Bayes models and 11 wins and 5 losses for repaired decision tree models.

Experiments without using a validation set yielded worse results, so the use of a validation set appears crucial. We

²Notice that decision trees can be viewed as scoring classifiers [Ferri *et al.*, 2002; Provost and Domingos, 2003].

1. Train a naive Bayes or decision tree model M on the training data; construct a ROC curve C and its convex hull H on the training data.
2. Find adjacent points on H such that in this interval the area between C and H is largest. Let T_1 and T_2 be the corresponding score thresholds.
3. Produce a new probabilistic model M' by calling RepairSection(T_1, T_2).
4. Evaluate M and M' on the validation set, construct their ROC curves and calculate their AUCs. If $AUC(M') \leq AUC(M)$ then go to 6.
5. Evaluate M and M' on the test set, construct their ROC curves and calculate their AUCs.
6. Go to 1. until each fold has been used as a test set.

Table 5: Experiment RepairSection.

Dataset	AUC (Original)	AUC (Repaired)	Better ?
Australia	90.9 ± 0.88	90.6 ± 0.94	×
Sonar	77.5 ± 1.30	76.8 ± 1.31	
Glass	76.6 ± 1.43	80.6 ± 1.36	✓
German	79.3 ± 0.88	78.9 ± 0.88	×
Car	99.0 ± 0.079	99.0 ± 0.08	
Anneal	86.7 ± 0.48	90.2 ± 0.47	✓
Monk1	75.4 ± 0.70	77.4 ± 0.70	✓
Monk2	64.6 ± 0.71	66.1 ± 0.92	✓
Monk3	96.2 ± 0.29	97.7 ± 0.24	✓
Hepatitis	86.3 ± 1.94	85.1 ± 1.22	
House	96.3 ± 0.35	96.3 ± 0.35	
Tic-Tac-Toe	74.1 ± 0.61	75.6 ± 0.58	✓
Heart	91.0 ± 1.04	90.2 ± 1.08	×
Ionosphere	93.2 ± 0.69	93.1 ± 0.690	
Breast Cancer	76.1 ± 1.63	77.7 ± 1.57	✓
Lymphography	90.3 ± 1.39	91.0 ± 1.49	✓
Primary Tumor	79.6 ± 1.02	80.6 ± 1.03	✓
Soybean-Large	91.9 ± 0.46	91.9 ± 0.46	
Solar-Flare	91.1 ± 0.71	91.2 ± 0.74	
Hayes-Roth	89.4 ± 1.53	91.3 ± 1.58	✓
Credit	90.8 ± 0.68	90.7 ± 0.65	
Balance	98.4 ± 0.27	98.4 ± 0.29	
Bridges*	92.7 ± 1.66	92.9 ± 1.89	
Average	86.41	87.1	

Table 6: Results of Experiment RepairSection with naive Bayes. *For the Bridges data set we used 5-fold cross-validation.

therefore conducted an experiment with two validation folds: we would only use the repaired model if its AUC was higher on both validation folds. The results for naive Bayes are shown in Table 8. We now obtain 12 significant wins and only 1 significant loss, and the average increase in accuracy is more than a percentage-point. Interestingly, two validation folds didn't work well for decision trees.

Finally, we conducted an experiment with naive Bayes whereby we selected all concavities, and repaired those that occurred on two validation sets. The results were similar to the results in the first experiment (11 significant wins and three losses), with some of the wins and losses occurring on

Dataset	AUC (Original)	AUC (Repaired)	Better ?
Australia	82.27 $\pm$ 0.50	82.28 $\pm$ 0.49	
Sonar	87.35 $\pm$ 1.20	88.92 $\pm$ 1.01	✓
Glass	76.09 $\pm$ 1.34	80.69 $\pm$ 1.15	✓
German	80.78 $\pm$ 0.55	81.01 $\pm$ 0.54	✓
Car	99.05 $\pm$ 0.074	99.07 $\pm$ 0.081	
Anneal	86.86 $\pm$ 0.41	90.64 $\pm$ 0.37	✓
Monk1	75.82 $\pm$ 0.74	78.05 $\pm$ 0.704	✓
Monk2	66.46 $\pm$ 0.66	67.95 $\pm$ 0.662	✓
Monk3	96.30 $\pm$ 0.32	98.14 $\pm$ 0.32	✓
Hepatitis	91.92 $\pm$ 2.84	91.92 $\pm$ 2.84	
House	96.18 $\pm$ 0.31	96.66 $\pm$ 0.29	✓
Tic-Tac-Toe	74.81 $\pm$ 0.54	75.97 $\pm$ 0.56	✓
Heart	91.72 $\pm$ 1.34	90.77 $\pm$ 1.21	×
Ionosphere	96.57 $\pm$ 0.91	95.88 $\pm$ 0.62	
Breast Cancer	74.82 $\pm$ 1.71	75.75 $\pm$ 1.51	
Lymphography	87.71 $\pm$ 1.86	87.88 $\pm$ 1.89	✓
Primary Tumor	76.63 $\pm$ 1.23	75.51 $\pm$ 1.36	×
Soybean-Large	90.93 $\pm$ 0.50	91.10 $\pm$ 0.52	×
Solar-Flare	86.61 $\pm$ 1.59	85.65 $\pm$ 1.65	×
Hayes-Roth	86.87 $\pm$ 2.24	88.33 $\pm$ 2.08	✓
Credit	91.09 $\pm$ 0.65	90.92 $\pm$ 0.64	
Balance	99.37 $\pm$ 0.18	99.42 $\pm$ 0.189	
Bridges	85.57 $\pm$ 4.36	81.71 $\pm$ 3.35	×
Average	86.16	86.71	

Table 7: Results of Experiment RepairSection with decision trees.

Dataset	AUC (Original)	AUC (Swapped)	Better ?
Australia	87.17 $\pm$ 2.28	86.86 $\pm$ 2.33	×
Sonar	81.34 $\pm$ 2.46	84.61 $\pm$ 2.66	✓
Glass	74.67 $\pm$ 1.61	74.75 $\pm$ 1.42	✓
German	82.33 $\pm$ 0.62	82.72 $\pm$ 0.60	✓
Car	99.25 $\pm$ 0.091	99.29 $\pm$ 0.098	
Anneal	87.06 $\pm$ 0.53	90.32 $\pm$ 0.48	✓
Monk1	75.68 $\pm$ 1.12	78.65 $\pm$ 0.92	✓
Monk2	65.74 $\pm$ 0.89	67.24 $\pm$ 0.95	✓
Monk3	96.75 $\pm$ 0.47	98.54 $\pm$ 0.25	✓
Hepatitis	92.67 $\pm$ 3.79	93.19 $\pm$ 3.37	
House	96.76 $\pm$ 0.37	96.76 $\pm$ 3.44	✓
Tic-Tac-Toe	75.62 $\pm$ 0.69	76.92 $\pm$ 0.71	✓
Heart	92.91 $\pm$ 1.29	92.98 $\pm$ 1.42	
Ionosphere	97.07 $\pm$ 0.77	97.53 $\pm$ 0.57	
Breast Cancer	78.20 $\pm$ 2.43	79.16 $\pm$ 2.12	
Lymphography	86.94 $\pm$ 2.41	90.49 $\pm$ 1.94	✓
Primary Tumor	78.14 $\pm$ 1.31	80.19 $\pm$ 1.27	✓
Soybean-Large	91.36 $\pm$ 0.45	91.63 $\pm$ 0.44	
Solar-Flare	89.69 $\pm$ 1.90	89.94 $\pm$ 1.88	
Hayes-Roth	84.89 $\pm$ 3.56	88.33 $\pm$ 3.09	✓
Credit	89.03 $\pm$ 1.56	88.67 $\pm$ 1.70	
Balance	99.53 $\pm$ 0.027	99.10 $\pm$ 0.078	
Average	86.49	87.63	

Table 8: Results with naive Bayes using two validation folds.

different data sets. It appears that repairing only the largest concavity and using two validation sets is the best strategy (at least for naive Bayes).

5 Discussion and conclusions

The work reported in this paper bears some similarity with ensemble methods. Bagging and boosting are two well-known ensemble approaches. Both approaches are implemented by re-sampling methods. In bagging [Breiman, 1996], the ensemble is formed by making bootstrap replicates of the training data sets and then multiple generated hypotheses are used to get an aggregated predictor. Boosting algorithms [Freund and Schapire, 1996] assign different weights to training instances depending on whether they are correctly classified. The approaches presented in this paper do not make use of re-sampling techniques.

Another relevant ensemble method is majority voting [Kimura and Shridar, 1991; Lam and Sue, 1997], in which the class predicted by the ensemble is the most predicted class among the base classifiers. Algorithm RepairPoint in this paper uses a kind of voting: when both Model 1 and Model 2 (see Figure 2) agree on the classification of an instance, then we choose that class. Otherwise, we choose the class not predicted by Model 3. In majority voting, on the other hand, we would choose the class predicted by Model 3. The difference is that majority voting does not take the quality of the different models into account, whereas our repair scheme knows that Model 3 is sub-optimal and therefore corrects its predictions in the relevant region.

ROC curves contain a wealth of information about the performance of one or more classifiers, which can be utilised to construct better models. They have been used to find optimal labelling of decision trees [Ferri *et al.*, 2002] and to find good decision thresholds for probabilistic classifiers [Lachiche and Flach, 2003]. In this paper we have proposed a novel approach to construct new models by repairing concavities in a ROC curve. The first method, RepairPoint, works on a probabilistic classifier with three probability thresholds, and tries to improve the poorest model with help of the other two. Preliminary experimental results (not reported) showed that this didn't work too well, but this may be due to the fact that the selection of the threshold for Model 3 is not easy. The threshold is a point chosen based on the ROC curve of the training data set; this point (see point c3 in Figure 3) has the farthest distance to the convex hull. If this threshold is not optimal on the test data (for instance, the position c3 in the ROC curve on the test data set unfortunately is located in the position c2), the AUC after repair becomes worse. Still, we believe that the idea of mirroring models around lines in ROC space will prove to be very useful. An interesting investigation for future work is whether a similar method can be made to work if the models are not obtained from a single scoring model (this would invalidate Theorem 1, i.e., the position of Model 4 may be different from the point obtained by point-mirroring).

The second method, RepairSection, locates and repairs an entire concave region of a ROC curve. Experimental results were very encouraging for both naive Bayes and decision trees. For naive Bayes we were able to improve results even further by using two validation folds, but this didn't work for decision trees. We are currently investigating why this is so. One possible explanation is that ROC curves ob-

tained from decision trees have lower resolution (because all instances in a leaf receive the same predicted probability), which may mean that concavities are less stable across samples. Pruning may be another factor, as it has been shown that pruning is detrimental for probability prediction [Provost and Domingos, 2003; Ferri *et al.*, 2003].

There are several other ways in which this work could be taken further. One is to investigate how much repair is possible, by concentrating on ROC curves with large concavities, possibly from artificial data sets. Another is to work with averaged ROC curves that are obtained by cross-validation (since each instance occurs in the test fold exactly once, an averaged ROC curve can be simply constructed by combining all instances with their predicted probabilities).

Acknowledgments

A preliminary version of this paper (without the experimental results with decision trees and two validation folds) appeared as [Flach and Wu, 2003]. We gratefully acknowledge the constructive comments made by the anonymous reviewers. We would also like to thank Rich Roberts for performing additional experiments.

References

- [Blake and Merz, 1998] C. Blake and C. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [Blockeel and Struyf, 2002] H. Blockeel and J. Struyf. Deriving biased classifiers for improved ROC performance. *Informatica*, 26(1):77–84, 2002.
- [Breiman, 1996] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Breiman, 1998] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26(3):801–849, 1998.
- [Fawcett, 2003] T. Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Laboratories, Palo Alto, CA, USA, 2003. www.purl.org/net/tfawcett/papers/HPL-2003-4.pdf.
- [Ferri *et al.*, 2002] C. Ferri, P. Flach, and J. Hernandez-Orallo. Decision tree learning using the area under the ROC curve. In C. Sammut and A. Hoffman, editors, *Proceedings of the 19th International Conference on Machine Learning*, pages 139–146. Morgan Kaufmann, 2002.
- [Ferri *et al.*, 2003] C. Ferri, P. Flach, and J. Hernandez-Orallo. Improving the AUC of probabilistic estimation trees. In N. Lavrač, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Proceedings of the 14th European Conference on Machine Learning*, volume 2837 of *Lecture Notes in Computer Science*, pages 121–132. Springer-Verlag, 2003.
- [Flach and Wu, 2003] P. Flach and S. Wu. Repairing concavities in ROC curves. In J. Rossiter and T. Martin, editors, *Proceedings of the 2003 UK Workshop on Computational Intelligence*, pages 38–44. University of Bristol, 2003.
- [Freund and Schapire, 1996] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.
- [Hand and Till, 2001] D. Hand and R. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.
- [Jacobs *et al.*, 1991] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
- [Kimura and Shridar, 1991] F. Kimura and M. Shridar. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, 24(10):969–983, 1991.
- [Lachiche and Flach, 2003] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning*, pages 416–423. AAAI Press, 2003.
- [Lam and Sue, 1997] L. Lam and C. Sue. Application of majority voting to pattern recognition: An analysis of its behaviour and performance. *IEEE Transactions on Systems, Man and Cybernetics*, 27(5):553–568, 1997.
- [Provost and Domingos, 2003] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.
- [Provost and Fawcett, 2001] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
- [Valentini and Masulli, 2002] G. Valentini and F. Masulli. Ensembles of learning machines. In M. Marinaro and R. Tagliaferri, editors, *13th Italian Workshop on Neural Nets*, volume 2486 of *Lecture Notes in Computer Science*, pages 3–22. Springer-Verlag, 2002.

Inferring Useful Heuristics from the Dynamics of Iterative Relational Classifiers

Aram Galstyan and Paul R. Cohen

USC Information Sciences Institute

4676 Admiralty Way, Suite 1001

Marina del Rey, California 90292

{galstyan,cohen}@isi.edu

Abstract

In this paper we consider dynamical properties of simple iterative relational classifiers. We conjecture that for a class of algorithms that use label-propagation the iterative procedure can lead to non-trivial dynamics in the number of newly classified instances. The underlying reason for this non-triviality is that in relational networks *true* class labels are likely to propagate faster than *false* ones. We suggest that this phenomenon, which we call *two-tiered dynamics* for binary classifiers, can be used for establishing a self-consistent classification threshold and a criterion for stopping iteration. We demonstrate this effect for two unrelated binary classification problems using a variation of an iterative relational neighbor classifier. We also study analytically the dynamical properties of the suggested classifier, and compare its results to the numerical experiments on synthetic data.

1 Introduction

Recently there has been a growing interest in relational learning and classification. While traditional learning approaches assume different data instances are independent and identically distributed, relational classification methods allow for the study of more complex data structures because it explicitly takes into account their existing links and relations. Recently developed algorithms for relational classification have been used for learning probabilistic relational models [Friedman *et al.*, 1999], hypertext classification [Getoor *et al.*, 2001], web page classification [Slattery and Craven, 2000; Macskassy and Provost, 2003], link prediction [Taskar *et al.*, 2004], discovery of authoritative information [Kleinberg, 1999], studying relational structures in scientific publications [McGovern *et al.*, 2003], etc.

Most of the existing methods for relational classification are iterative in nature [Neville and Jensen, 2000; Macskassy and Provost, 2003]. The intuitive assumption behind the iterative approach is that information inferred about one entity can be used to make inferences about other entities that are related to it. This allows class labels (or associated probabilities) to propagate throughout the system as newer instances are classified. Although iterative classifiers have been shown

to enhance prediction accuracy [Neville and Jensen, 2000; Macskassy and Provost, 2003], there is an associated risk: a few false inferences sometimes lead to a “snow-ball” effect [Neville and Jensen, 2000] cascading the number of misclassified entities as iteration proceeds. As a consequence, the final results will depend strongly on the accuracy of initially assigned class labels and on the classification criteria, i.e., model parameters. More generally, it is known that many classification algorithms are sensitive to parameter settings. For example, [Keogh *et al.*, 2004] found that few popular data mining algorithms performed well on multiple problems without parameter adjustments.

In this paper we argue that for a class of iterative classifiers the issue of parameter sensitivity can be addressed in a self-consistent and elegant manner by examining the dynamics of the iterative procedure more closely. We illustrate this point for a simple iterative classifier that uses a threshold-based criterion for labelling data-instances, and propose a meta-heuristic for setting the optimal threshold value. Our heuristic is based on the assumption that in relational networks true class labels are likely to propagate faster than false ones. We demonstrate that this phenomenon, which we call *two-tiered dynamics*, can serve as a natural criterion for both setting a threshold value and stopping the iteration. Hence, our method reduces the dependency of iterative classification on model parameters.

To illustrate our approach, we introduce a simple algorithm for relational binary classification. We consider the case when the relation between two data-instances is characterized by the weight of the link connecting them, so that the relational structure is fully determined by an adjacency matrix $\mathbf{M}$. Given an initial set of known instances of a class A , our algorithm then defines an iterative scheme with a simple threshold based rule: an instance will be classified as type A if it is connected to super-threshold number of classified or initially known instances of type A . The novelty of our approach is that we suggest a self-consistent way of setting the parameter of our model, the threshold: namely, we set it automatically to the value that produces the most pronounced two-tiered dynamics. We present empirical results that illustrate the use of this approach. We also develop an analytical framework for describing the dynamics of iterative classification of our algorithm, and compare its predictions with results obtained for synthetic, randomly generated networks.

The rest of the paper is organized as follows. In the next section we provide a more detailed description of our algorithm. In Section 3 we present results for two case studies that demonstrate the two-tier dynamics. In Section 4 we present an analytical framework for studying the dynamical properties for a binary iterative classifier. Discussion on our results and future developments are presented in Section 5.

2 Dynamics of for Iterative Classification

To understand the main idea behind our approach, we find it illustrative to frame the classification problem as an epidemic process. Specifically, let us consider a binary classification problem defined on a network where data-instances (from classes A and B) correspond to nodes and the relations between them are represented by (weighted) links. We are given the correct classification of a small subset of nodes from class A and want to classify other members of this class. Assume that we are using a simple, threshold based iterative relational classifier (such as one described below in Fig. 1), for assigning class labels and propagating them through the system. Now, if we treat the initially labelled data-instances as “infected”, then the iterative scheme defines an epidemic model where at each time step new instances will be infected if the super-threshold classification criterion is met. Clearly, the fixed point of this epidemic process will depend both on the value of the threshold and both inter- and intra-class link structure of the network. In the case where two classes are totally decoupled (i.e., there are no cross-links between two sub-classes) the epidemics will be contained on the set A , and one can relax the classifier threshold to guarantee that all the instances of A will be correctly classified. If there are links between data-instances in different sub-classes, then there is a chance that the nodes from class B will be infected too (i.e., misclassified). However, if the link patterns between two subclasses are sufficiently different, we can hope that the process of epidemic spreading in two systems will be separated in time. Moreover, by tuning the classification threshold, we can control the rate of epidemic spreading in sub-population A , hence affecting the epidemic spread in sub-population B . The main idea of our approach is to tune the threshold parameter in order to achieve maximum temporal separation of epidemic peaks in two classes.

We characterize the dynamics of an iterative classifier by the number of newly classified instances at each time step, e.g., if $N(t)$ is the total number of classified A -instances at time t , then the relevant variable is $\Delta N(t) = N(t) - N(t-1)$. As it will be clear later, two-tiered dynamics arises whenever $\Delta N(t)$ has two temporally separated peaks.

To proceed further, we now formally define our binary classification algorithm. Let S be the set of data-instances to be classified, and let assume that S is composed of two subsets $S_A \subset S$ and $S_{-A} = S \setminus S_A$. Initially, we know the correct class labels of a (small) subset of the instances of type A , S_A^0 , and the problem is to identify other members of class S_A given the characterizing relations between the entities across both types. We define M_{ij} as the weight of the link between the i -th and j -th entities.

We associate a state variable with each entity, $s_i = 0, 1$

so that the state value $s_i = 1$ corresponds to type A . Initially, only the entities with known class labels have $s_i = 1$. At each iteration step, for each non-classified instance we calculate the cumulative weight of the links of that instance with known instances of type A . If this cumulative weight is greater or equal than a preestablished threshold H , that instance will be classified as a type A itself. This is shown schematically in Figure 1. Note that our algorithm differs slightly from other simple relational classifiers (such as Relational Neighbor classifier [Macskassy and Provost, 2003]) in two aspects: First, it directly assigns class labels and not probabilities, and second, it is asymmetric in the sense that if an instance was classified as type A it will remain in that class for the remainder of the run. This later property implies that the total number of classified A -instance is a monotonically non-decreasing function of time. If the number of iterations

```

input adjacency matrix  $\mathbf{M}$ 
initialize  $s_i = 1$ , for initially known instances,  $s_i = 0$  for the rest
initialize a threshold  $H$ 
iterate  $t = 0 : T_{max}$ 
  for  $i$ -th node with  $s_i(t) = 0$ 
    calculate the weight  $w_i$  of adversary nodes connected to it:
     $w_i = \sum M_{ij} s_j(t)$ 
    if  $w_i \geq H \Rightarrow s_i(t+1) = 1$ 
  end for loop
end

```

Figure 1: Pseudo-code of the iterative procedure

T_{max} is large enough, then a steady state will be achieved, i.e., no instance will change its state upon further iterations. As we mentioned above, the final state of the system will depend on the threshold value and the adjacency matrix. If the threshold value is set sufficiently low then the system will evolve to a state where every instance has been classified as type A . On the other hand, if it is set too high, then no additional instances will be classified at all. As we will show below, for intermediary values of the threshold H the system will demonstrate two-tier dynamics.

3 Case Studies

In this section we test our hypothesis empirically on two distinct and unrelated data-sets: The first is a synthetic data generated by the Hats Simulator [Cohen and Morrison, 2004], and the second is Cora [McCallum *et al.*, 2000], a large collection of research papers in computer science.

3.1 Results for the Hats Simulator Data

The Hats simulator is a framework designed for developing and testing various intelligence analysis tools. It simulates a virtual world where a large number of agents are engaged in individual and collective activities. Each agent has a set of elementary capabilities which he can trade with other agents if desired. Most of the agents are benign while some are covert adversaries that intend to inflict harm by destroying certain landmarks called *beacons*. There also are agents known to be adversaries. Agents travel for meetings that are planned

by an activities generator. Each agent belongs to one or more organizations that can be of two types, benign or adversary. Each adversary agent belongs to at least one adversary organization, while each benign agent belongs to at least one benign organization and does not belong to any adversary organization. When a meeting is planned, the list of participants is drawn from the set of agents that belong to the same organization. Hence, a meeting planned by an adversary organization will consist of only adversary (either known or covert) agents, whereas a meeting planned by a benign organization might contain all three types of agents.

The type of data from the Hats simulator is a sequence of lists containing unique hat ID-s that have met with each other. Given this sequence, one can unequivocally construct a graph (adjacency matrix) $\mathbf{M}$ of hats' meeting activities, where each entry M_{ij} describes the number of meetings between the i -th and j -th agents (note that the graph is not directed so the matrix is symmetric). In the simulations presented here we used Hats data for $N = 1200$ agents, ($N_k = 50$ known adversaries, $N_b = 1000$ benign, and $N_c = 150$ covert) which was collected for the first 2500 ticks of simulations.

We tested our algorithm for small, large, and intermediate values of the threshold H . For small H most of the nodes in the network are infected after a short time, as expected (see Fig. 2). Similarly, for large values of H (not shown here) the epidemic spreads through only a very small subset of nodes. In both cases the epidemics are characterized by one-tier dynamics. The situation is drastically different for intermediate values of the threshold, as the behavior of epidemic spreading demonstrates two-tiered structure. Namely, after a sharp initial spread the epidemic seems to be saturated. However, upon further iterations, the number of infected nodes increases sharply, and all the nodes in the network are infected shortly thereafter. Clearly, this corresponds to some kind of threshold-phenomenon in the whole network, where infection of certain nodes causes an epidemic in the whole system. This is illustrated in Fig. 2(a) where we plot the number of infected actors vs time for $H = 4$ and $H = 6$. As we mentioned in the introduction, this behavior suggests a natural criterion for stopping the iteration. More precisely, in Fig. 2(b) we plot the number of newly infected nodes at each times step versus time, i.e., $\Delta N(t) = N(t) - N(t-1)$. For the threshold value $H = 4$ (not plotted) there is only a single peak in $\Delta N(t)$. However, for $H = 6$ one can distinguish two well-separated peaks, denoted P_a and P_b which are indicative of two-tier dynamics in epidemic spreading. If we assume that P_a corresponds to the epidemic spreading in the first (covert) sub-population, and P_b corresponds to the rest of the network, then the iteration should be stopped right before the infection starts to spread in the rest of the system (i.e., at $t = 5$). In this particular example, we established that if the above mentioned criterion is used, then the algorithm correctly identifies 118 out of 150 covert adversaries, while at the same time misidentifying only 8 of the 1000 benign nodes as adversaries. This is a surprising result especially taking into account the simplicity of the algorithm.

More generally, our experiments with the Hats data indicate that although the detection error rate of the algorithm varies depending on the particular structure of the network,

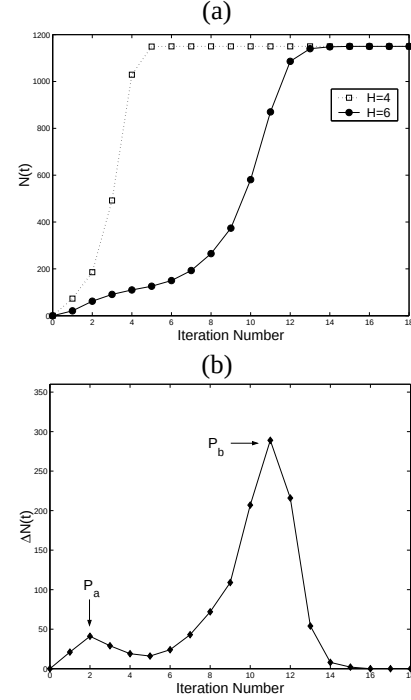


Figure 2: (a) Total number of infected nodes $N(t)$ for $H = 4$ and $H = 6$. (b) The number of newly infected instances vs time, $\Delta N(t)$ for $H = 6$. Two separated peaks are a clear indication of two-tier dynamics.

the amount of the available data, as well as the presence of noise, its performance is rather robust as long as the two-tier dynamics is observed.

3.2 Results for Cora Data

The Cora data [McCallum *et al.*, 2000] contains a set of computer science research papers that are hierarchically categorized into topics and subtopics. Each paper includes a label for a topic/subtopic/sub-subtopic, and a list of cited articles. Following the previous studies [Macskassy and Provost, 2003], we focused on the papers on Machine Learning category, that contained seven different subtopics: Case-Based, Theory, Genetic Algorithms, Probabilistic Methods, Neural Networks, Rule Learning and Reinforcement Learning. Two papers are linked together by using common author (or authors) and citation.

Since our algorithm is for binary classification, we constructed separate classification problem for each topic. We varied the fraction of known instances from as high as 50% to as low as 2%. For each classification task, we did up to 10 different runs using random subsets of classified hats. Note, that initially labelled set contained papers that belong to a class other than one we wanted to identify (the class label of all the papers in the initially labelled sets were fixed throughout iteration).

After pruning out the isolated papers from the data-set, we were left with 4025 unique titles. Since we observed a large dispersion in the node connectivity ranging from 1 to more

than 100, we revised our threshold-based rule a little so that the threshold condition was established not only for the total weight, but also on the fraction of that weight.

We observed that the structure of dynamics (i.e., two-tier vs single-tier) varied from topic to topic. From the seven subtopics, the data that demonstrated the best manifestation of two-tiered dynamics was Reinforcement Learning subtopic: it robustly demonstrated two separate peaks from run to run for various fraction of initially known data as shown in Fig 3(a). What is more striking, however, is that the accuracy of classification was remarkable even if the fraction of initially known instances were as low as 2% of the total number. Indeed, as illustrated in Fig 3(b), for 2% of initially known class-labels, and from which only 7 in the Reinforcement Learning Topic, the F -Measure at the iteration-stopping point $t = 8$ is $F_M \approx 0.66$. Moreover, our experiments also suggest that increasing the number of initially labelled data does not necessarily improve the performance. Although this seems counterintuitive, it makes sense from the perspective of our algorithm: Indeed, the more labelled instances we have at the start, the better the chances that the epidemics will leave the sub-network and start to infect nodes from other classes. One could think of increasing the threshold would help, but it did not, probably because of large dispersion in node connectivity. Of course, one can always sample from available known instances and choose to include only an appropriate number of them.

We observed two-tier structures in most of the other topics too. Although some of them were not as pronounce as for the previous case, they were robust in the sense that uprise of the second peak almost surely corresponded with the spread of label-propagation outside of that class. However, in some instances, notably for the Neural Networks subtopic, we did not observe any clear sign of this dynamics at all. Our explanation is that this subcategory was vastly larger than the RL -one (1269 compared to 344) so it was easier for the initial infection to propagate outside. This suggests that perhaps our method is best when one wants to identify a sub-class that is considerably smaller compared to the total number of instances.

4 Analysis

In this section we present an analysis of our classification algorithm. Specifically, we study the epidemics spreading as described by our algorithm. For the sake of simplicity, we consider a case of an unweighed graph when the entries in the adjacency matrix are either 0 or 1. Generalization to the case of the weighed networks is straightforward. Also, for clarity purposes we will retain the language of section 3.1 and refer to instances as agents.

Let us start by considering only one of the subnetworks in the system. Namely, we neglect benign agents for now and consider a network consisting of covert and known adversaries only. We want to examine how the epidemic spreads through the covert population.

We now consider the iterative procedure more closely. Initially, all of the agents except known adversaries are classified as not-infected, $s_i(t = 0) = 0, i = 1, 2, \dots, N$. We define a

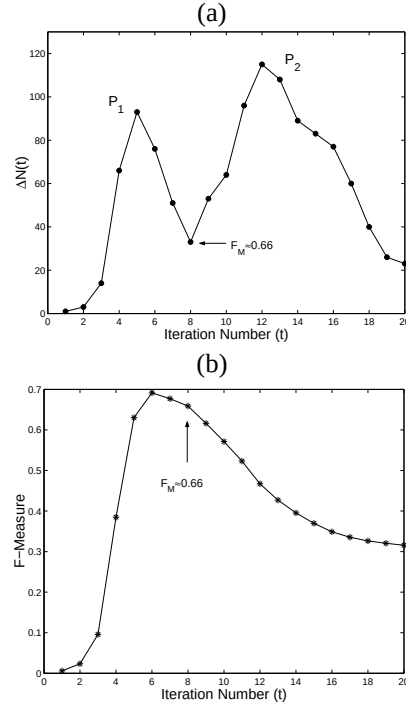


Figure 3: (a) $\Delta N(t)$ for $H = 4$, with 2% of initially classified instances. (b) F -Measure of predictive accuracy vs iteration step. At $t = 8$ (the iteration stopping point), $F_M \approx 0.66$.

local field h_i for the i -th agents as the number of its connections with known adversaries. We assume that h_i -s are uncorrelated random variables drawn from a probability density function $\mathcal{P}(h)$. Let $f(h) = \sum_{h' \geq h} \mathcal{P}(h')$ be the fraction of agents who are connected with at least h initially known adversary agents. Also, let k_i be the number of connections the i -th agent has with newly classified adversaries (note that k_i excludes the links to the initially known adversaries) and assume that k_i -s are described by a probability density function $P(k; t)$. In other words, $P(k; t)$ is the probability that a randomly chosen uninfected covert agent at time t is connected to exactly k infected covert agents. Since the number of infected agents changes in time, so does the distribution $P(k; t)$. Initially, one has $P(k; t = 0) = \delta_{k,0}$, where δ_{ij} is the Kronecker's symbol.¹

In the first step of the iteration, the agents who have a local field larger than or equal to the threshold value, $h_i \geq H$, will change their states to 1. Hence, the fraction of agents classified as adversaries at $t = 1$ is $n(t = 1) = f(H)$. Since these new adversary agents are connected to other non-classified agents, this will alter the local fields for non-classified agents. Let us define a variable for each agent $z_i = h_i + k_i$. Then the distribution of z_i is described by

$$P(z; t) = \sum_{k=0}^{\infty} \sum_{h=0}^{\infty} P(k; t) \mathcal{P}(h) \delta_{z, k+h}$$

¹Kronecker's symbol is defined as follows: $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0, i \neq j$.

$$= \sum_{k=0}^{\infty} P(k; t) \mathcal{P}(z - k) \quad (1)$$

Clearly, the criterion for infection is $z_i \geq H$. Hence, the fraction of agents that will be classified as covert at time $t + 1$ is

$$n(t + 1) = \sum_{z=H}^{\infty} P(z, t) = \sum_{k=0}^{\infty} P(k; t) f(H - k) \quad (2)$$

Note that the probability $P(k; t)$ of being connected to an infected agent depends on the fraction of infected agents, $P(k; t) = P(k; n(t))$. Hence, the Equation 2 is in general a highly non-linear map. Once the functions $P(k; t)$ and $f(h)$ are specified, Equation 2 can be solved (at least numerically) to study the dynamics of epidemic spreading in a homogeneous, single-population network. In particular, the final fraction of infected agents is given by its steady state $n(t \rightarrow \infty)$.

The above framework is easily generalized for the case when there are two sub-populations in the network. We denote two sub-populations by C (covert) and B (benign). Let $f_c(h)$ $f_b(h)$ be the fraction of C and B agents respectively that are connected to at least h known adversaries. Also, let $P_{cc}(k; t)$ and $P_{cb}(k; t)$ be the probability that a randomly chosen C agent is connected to exactly k infected C and infected B agents, respectively. Similarly, we define $P_{bb}(k; t)$ and $P_{bc}(k; t)$ as the probability that a randomly chosen B agent is connected to k infected B and infected C agents, respectively. Then the fraction of infected agents in each population $n_c(t)$ and $n_b(t)$ satisfy the following set of equations:

$$\begin{aligned} n_c(t + 1) &= \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} P_{cc}(k; t) P_{cb}(j; t) f_c(H - k - j) \\ n_b(t + 1) &= \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} P_{bb}(k; t) P_{bc}(j; t) f_b(H - k - j) \end{aligned} \quad (3)$$

To proceed further we need to make assumptions about the distribution functions $P_{cc}(k; t)$, $P_{bb}(k; t)$, $P_{cb}(k; t)$ and $P_{bc}(k; t)$ i.e., probability that a randomly chosen uninfected agent of type C (B) is connected to k infected agents of respective type. This is particularly easy to do when the graph is obtained by randomly establishing a link between any two agents/nodes with a fixed probability. Specifically, let us assume that each covert agent is connected to covert and benign agents with corresponding probabilities p_{cc} and p_{cb} , while each benign agent has similarly defined probabilities p_{bb} and p_{bc} (note that $p_{cb} = p_{bc}$). Hence, each covert agent in average is connected with $\gamma_{cc} = p_{cc}N_c$ covert and $\gamma_{cb} = p_{cb}N_b$ benign agents, and each benign agent is connected with $\gamma_{bb} = p_{bb}N_b$ benign and $\gamma_{bc} = p_{bc}N_b$ benign agents.

Consider now a randomly chosen uninfected agent of either type, say, covert, at a certain time t , and denote it as c_0 . There are $N_c n_c(t)$ infected covert agents at time t , and c_0 is connected with each of them with probability p_{cc} . Hence, the probability $P_{cc}(k; t)$ that c_0 is connected with exactly k infected covert agents at time t is given by a Poisson distribution with a mean $\gamma_{cc} n_c(t)$. Similar arguments hold also for

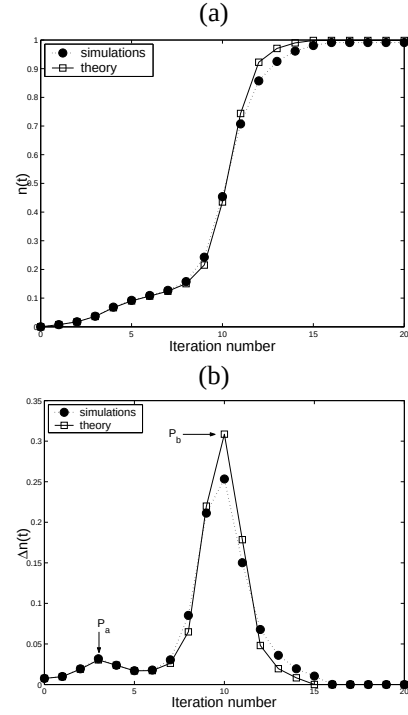


Figure 4: Analytical and simulation results for (a) $n(t)$ and (b) $\Delta n(t)$ for a random network. The results are averaged over 100 trials

$P_{bb}(k; t)$, $P_{cb}(k; t)$ and $P_{bc}(k; t)$. Hence, one obtains from the Equation 3

$$\begin{aligned} n_c(t + 1) &= \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} \frac{[\gamma_{cc} n_c(t)]^k}{k!} \frac{[\gamma_{cb} n_b(t)]^j}{j!} \\ &\times f_c(H - k - j) e^{-\gamma_{cc} n_c(t) - \gamma_{cb} n_b(t)} \end{aligned} \quad (4)$$

$$\begin{aligned} n_b(t + 1) &= \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} \frac{[\gamma_{bb} n_b(t)]^k}{k!} \frac{[\gamma_{bc} n_c(t)]^j}{j!} \\ &\times f_b(H - k - j) e^{-\gamma_{bb} n_b(t) - \gamma_{bc} n_c(t)} \end{aligned} \quad (5)$$

Equations 4 and 5 are a system of coupled maps that governs the evolution of fraction of infected individuals in both sub-populations. The coupling strength depends on γ_{cb} and γ_{bc} , or in other words, average number of interconnections between two sub-populations. Note that if one sets $\gamma_{cb} = \gamma_{bc} = 0$ the dynamics of two sub-populations are totally independent and one recovers the system Equation 2 with corresponding parameters for each sub-population. To validate the prediction of our analysis, we compared Equations 4 and 5 with experiments on randomly generated graphs. The results are shown in Fig. 4 where we plot the fraction of the infected nodes $n(t) = n_c(t) + n_b(t)$ as well as the difference $\Delta n(t) = n(t + 1) - n(t)$ versus iteration number for a randomly generated graph of $N_c = 100$ covert and $N_b = 1000$ benign nodes. The parameters of the graph are $p_{cc} = 0.2$, $p_{bb} = 0.04$ and $p_{cb} = 0.04$. Also, we chose the

value of the threshold field such that to ensure two-tier dynamics. The results of the simulations were averaged over 100 random realization of graphs. Clearly, the agreement between the analytical prediction given by equations 4 and 5 and the results of the simulations is quite good. In particular, these equations accurately predicts the two-tier dynamics observed in the simulations. We also note that the graphs are structurally very similar to the results from the Hats simulator data in Fig. 2. This suggests that despite the the explicit organizational structure in the Hats data, its infection dynamics is well captured by a simple random-graph analysis model. Note however, that this agreement might deteriorate for more complex organizational structure (e.g., large overlap between different organizations), hence more sophisticated analytical models might be needed.

5 Discussion and Future Work

We have presented a simple, threshold based iterative algorithm for binary classification of relational data. Our algorithm can be stated in terms of epidemic spreading in networks with two sub-populations of nodes (data-instances) where infected nodes correspond labelled data-instances. We also presented a meta-heuristics that utilizes the differences in the propagation of true and false class-labels for setting the right threshold value in our algorithm. Specifically, we demonstrated that if the threshold value is tuned appropriately, the dynamics of the number of newly classified instances will have a two-peak structure suggesting that the infection propagation in two sub-classes is time-separated. Consequently, we suggested that the iteration should be stopped at the point when the second peaks starts to develop.

Our empirical tests, especially with Cora, indicate that the two-tier dynamics is not an artifact, but is present in real world relational data. Although we did not observe this dynamics in all the classification tasks in Cora, our results nevertheless indicate that whenever the two-tier dynamics is present, it is indeed robust, and contains useful information that can be utilized by classification algorithm. In addition, our experiments, as well as qualitative arguments on epidemic spreading, suggest that the method presented in this paper should work best when the sub-class one wants to identify is a small fraction of the whole data-set, as there is a greater chance that the class-labels will propagate throughout the proper sub-population first before infecting instances of other classes.

We also developed an analytical framework for studying the properties of iterative classification. In particular, we obtained a coupled of set discrete-time maps that describe the evolution infected/labelled individuals in both sub-populations. We compared our analytical results with numerical experiments on synthetic data and obtained excellent agreement. We would like to mention that the assumption of a random graph we used in our analysis is clearly an over-simplification. Indeed, most of the real-world relational structures (e.g., social networks) demonstrate small-world phenomenon that is not captured by our random graph model. In our future work we intend to extend our framework to account for more general type of networks. Note

that in this scenario the probability of being infected will be strongly correlated with a degree of a node (i.e., more links will imply more chances of being infected).

6 Acknowledgements

We thank Joshua Moody for preparing the Hats data.

References

- [Cohen and Morrison, 2004] P. Cohen and C. T. Morrison. The hats simulator. In *Proceedings of the 2004 Winter Simulation Conference*, Washington, DC, 2004.
- [Friedman et al., 1999] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1300–1307, Stockholm, Sweden, 1999.
- [Getoor et al., 2001] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *Proceedings of IJCAI-01 Workshop on Text Learning: Beyond Supervision*, Seattle, WA, 2001.
- [Keogh et al., 2004] E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, WA, 2004.
- [Kleinberg, 1999] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [Macskassy and Provost, 2003] S. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of Workshop on Multi-Relational Data Mining in conjunction with KDD-2003*, Washington, DC, 2003.
- [McCallum et al., 2000] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.
- [McGovern et al., 2003] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen. Exploiting relational structure to understand publication patterns in. *SIGKDD Explorations*, 5(2):165–172, 2003.
- [Neville and Jensen, 2000] J. Neville and D. Jensen. Iterative classification in relational data. In *Proceedings of AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, Austin, TX, 2000.
- [Slattery and Craven, 2000] Seán Slattery and Mark Craven. Discovering test set regularities in relational domains. In *Proceedings of ICML-2000*, pages 895–902, Stanford, US, 2000.
- [Taskar et al., 2004] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Learning Coordination Classifiers

Yuhong Guo Russell Greiner Dale Schuurmans

Department of Computing Science

University of Alberta

{yuhong,greiner,dale}@cs.ualberta.ca

Abstract

We present a new approach to ensemble classification that requires learning only a single base classifier. The idea is to learn a classifier that simultaneously predicts *pairs* of test labels—as opposed to learning multiple predictors for single test labels—then coordinating the assignment of individual labels by propagating beliefs on a graph over the data. We argue that the approach is statistically well motivated, even for independent identically distributed (iid) data. In fact, we present experimental results that show improvements in classification accuracy over single-example classifiers, across a range of iid data sets and over a set of base classifiers. Like boosting, the technique increases representational capacity while controlling variance through a principled form of classifier combination.

1 Introduction

Supervised learning has been by far the most studied task in machine learning research. The problem is to take a finite set of observed training examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ and produce a classifier $f : X \rightarrow Y$ that achieves small misclassification error on subsequent test examples. Most research has tended to adopt a standard “iid” assumption that the training and test examples are independent and identically distributed. In fact, this assumption is fundamental to much of the theoretical research on the topic [Anthony and Bartlett, 1999] and also characterizes most standard learning methods—as exemplified by the fact that most machine learning methods classify each test pattern in isolation, independently of other test patterns.

Recently, however, increasing attention has been paid to problems where the training and test labels are not independent, but instead strongly related. For example, in domains such as part of speech tagging and webpage classification, each word-tag or webpage-label depends on the tag or label of proximal words or webpages, in addition to just the features of the immediate word or webpage. Various forms of “relational” learning models have been developed to handle these kinds of problems over the last few years. A notable example is work on probabilistic relational models (PRMs), where the correlation between the class labels of different instances is

explicitly represented in a directed graphical model [Getoor *et al.*, 2001; 2002]. Other approaches for learning multivariate classifiers include conditional random fields (CRFs) [Lafferty *et al.*, 2001], relational Markov networks (RMNs) [Taskar *et al.*, 2002], and maximum margin Markov networks (M3N) [Taskar *et al.*, 2003]. All of these methods have led to substantial progress on learning classifiers that make dependent predictions of test labels that are explicitly related.

Although learning multivariate predictors is an exciting problem, we nevertheless focus on the classical iid case in this paper. However, we demonstrate what we believe is a surprising and counterintuitive connection: that learning multivariate dependent predictions is a beneficial idea even in the iid setting. In particular, we develop a relational learning strategy that classifies test patterns by connecting their labels in a graphical model—hence correlating the subsequent predictions—even when it is explicitly assumed that all training and test examples are iid.

Before explaining the rationale behind our approach and explaining why dependent prediction still makes sense in the iid setting, we note that standard relational learning approaches, such as PRMs, CRFs, RMNs and M3Ns, do not naturally correlate predictions on iid data. That is, these techniques only consider label dependencies that are explicitly asserted to hold in the true underlying model of the domain. In the iid case, since no dependencies are asserted between test labels, these standard relational approaches reduce to single-label learning techniques, such as univariate logistic regression and support vector machines. However, what we are proposing is different: we intentionally *add* dependencies between test labels, even when all labels are explicitly assumed to be independent in the underlying data generation process. Surprisingly, we demonstrate that correlating predictions can still be advantageous.

After introducing our basic approach below, we then motivate and justify our technique in three separate ways. First, we show that predicting correlated test labels is statistically justified in the iid setting, even when the independence assumptions are explicitly taken into account. In fact, we show that it is incorrect to conclude that a learned predictor can sufficiently treat test cases as independent simply because they come from an iid source. Second, we show that our proposed relational learning technique can be viewed as a natural generalization of similarity-based learning techniques.

Moreover, it can also be viewed as a simple form of ensemble learning method that has some advantages over standard approaches. Third, we show empirically that our proposed method can achieve improvements in classification accuracy across a range of iid domains, for different base learning algorithms.

2 Learning Coordinated Label Predictors

We begin by simply introducing our learning method, and then attempt to motivate it more thoroughly below. Initially, we will focus on using probabilistic classifiers (although we briefly consider an extension to nonprobabilistic classifiers in Section 5 below).

In the iid setting, the standard approach to probabilistic classification is to learn a univariate model $P(y|\mathbf{x}, \theta)$ that asserts a conditional probability distribution over a single classification variable y given an input pattern $\mathbf{x}$, where θ represents the parameters of the model. Given such a representation, there are two key steps to building a univariate classifier: The first is to learn a specific predictive model $P(y|\mathbf{x}, \theta)$ given training data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, based on using a principle such as maximum (conditional) likelihood or maximum a posteriori estimation. Then, given a set of test patterns $\mathbf{x}_1^*, \dots, \mathbf{x}_m^*$, one classifies each $\mathbf{x}_i^*$ independently by computing the label $\hat{y}_i$ that maximizes the estimated conditional probability, $\hat{y}_i = \arg \max_y P(y|\mathbf{x}_i^*, \theta)$. Natural examples of this approach are learning naive Bayes classifiers [Friedman *et al.*, 1997], logistic regression classifiers [Hastie *et al.*, 2001], kernel logistic regression classifiers [Zhu and Hastie, 2001], sigmoid network classifiers [Neal, 1996], and Bayesian network classifiers [Greiner and Zhou, 2002].

Our approach is different. Instead of learning a univariate classifier that predicts only a single label, we instead propose to learn a *pairwise* label classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ that takes an arbitrary pair of input patterns, $\mathbf{x}_i$ and $\mathbf{x}_j$, and asserts a conditional *joint* distribution over the pair of labels y_i and y_j . For example, if there are two classes, say 0 and 1, then a pairwise classifier would assert the conditional probability of the four possible pair labelings $(y_i, y_j) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ given the two input patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. In general the pairwise predictor does *not* assume that y_i and y_j are independent given $\mathbf{x}_i$ and $\mathbf{x}_j$, even though they are indeed assumed to be independent in the true model (we present a justification of this in Section 3 below). We refer to a pairwise label classifier of this form as a “*coordination classifier*” to highlight the fact that it attempts to model any coordination that might appear between the labels y_i and y_j given the input patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. Given the alternative representation $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ we next describe the two main processes of, first, training a coordination classifier from data, and then using it to label test patterns.

2.1 Training a Coordination Classifier

A coordination classifier doubles the number of input features and squares the number of output classes from an original univariate classifier. Despite the increase in model complexity, training a coordination classifier remains conceptually straightforward. Assume a standard training sample

$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ is given. First we construct a set of pairs from the set $\{(\mathbf{x}_i \mathbf{x}_j, y_i y_j)\}$ and then supply these as a conventional training set for learning a predictive model $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ from data. (In our experiments below we ignore duplicate pairs but otherwise include both orderings of each distinct pair to ensure that the learned model is symmetric.) Once the training data is constructed, the parameters of the model, ϕ , can then be estimated in the same way as the univariate case, either by using a maximum (conditional) likelihood or maximum a posteriori estimation principle. For example, given a linear logistic representation for $P(y_i | \mathbf{x}_i, \theta)$, we use an analogous linear logistic representation for $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ but over the joint feature space $\mathbf{x}_i \mathbf{x}_j$; thus training the two models under the same estimation principle, but using different (although related) data sets.

A coordination model learned in this way will generally not make independent predictions of y_i and y_j , since the extended parameters ϕ are not constrained to enforce independence.¹ That is, we expect the model to learn to make dependent, coordinated predictions of the two labels from the corresponding input patterns. Interestingly, learning a coordination classifier has the advantage of potentially squaring the number of available training examples, even though this advantage is mitigated by subsampling and the increase in the complexity of the model being learned.

2.2 Classifying Test Data with Coordination

Given a coordination classifier, we require a principle for classifying individual test patterns $\mathbf{x}^*$. In fact, the problem of classifying test patterns becomes much more involved in this case. The approach we take is to consider the set of training examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ and test patterns $\mathbf{x}_1^*, \dots, \mathbf{x}_m^*$ as a whole. That is, rather than classify each test pattern $\mathbf{x}_i^*$ in isolation, we instead seek to classify test patterns in a dependent manner. To perform classification, we proceed in three stages reminiscent of conditional random fields: First, we construct a graph over the test and training labels. Once the graph has been constructed, we then use the learned coordination classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ to assign “potentials” to the possible labelings of each edge (y_i, y_j) . These potentials can then be used to define a Markov random field over test label assignments, thereby establishing a joint probability distribution over the labelings. Finally, we compute a joint labeling $y_1^*, \dots, y_m^*$ of the test examples that maximizes (or at least approximately maximizes) the joint label probability. We describe each of these three steps in more detail below.

Defining the graph First, to construct the graph, we only consider edges that connect a pair of *test* labels (y_i^*, y_j^*) , or a test label and a training label (y_i^*, y_j) . That is, after training we do not make any further use of edges between training labels.

To classify test patterns, the simplest approach, conceptually, is to consider the *complete* graph that connects each

¹Many readers are probably objecting at this point that, given the iid assumption, there can be no new information to be gained from the n^2 example pairs that was not already present in the original n examples. However, Section 3 argues that this conclusion is generally incorrect in a machine learning context.

test label y_i^* to all other test and training labels. However, we have found that it is usually impractical to consider all $m((m-1)/2 + n)$ test pairs (ignoring duplicate pairs). Therefore, we reduce the number of edges by adding a few restrictions. The natural alternatives we consider below are: (i) connecting each test label only to training labels (which, as we will see, is analogous to standard similarity and kernel based learning methods), (ii) connecting each test label only to other test labels (which, surprisingly, gives the best results in our experiments below), and (iii) connecting each test label to both training and test labels. To further reduce the overall number of edges, we then uniformly subsample edges, subject to these different restrictions.

Defining the potentials Once a graph has been constructed, we then assign potentials to the configurations of each edge. There are two cases depending on whether the edge connects two test labels, or a test label and a training label.

For an edge that connects only two test labels, (y_i^*, y_j^*) , we simply assign the potential $\psi(y_i^*, y_j^*) = P(y_i^* y_j^* | \mathbf{x}_i^* \mathbf{x}_j^*, \phi)$ given by the learned coordination classifier.

For an edge that connects a test and a training label, (y_i^*, y_j) , we assign a unit potential to the singleton node (y_i^*) given by the conditional probability of y_i^* given y_j . That is, we assign

$$\psi_{y_j}(y_i^*) = P(y_i^* | y_j, \mathbf{x}_i^* \mathbf{x}_j, \phi) = \frac{P(y_i^* y_j | \mathbf{x}_i^* \mathbf{x}_j, \phi)}{\sum_y P(y y_j | \mathbf{x}_i^* \mathbf{x}_j, \phi)}$$

Once a potential has been assigned to the singleton (y_i^*) we then remove the edge (y_i^*, y_j) from the graph. Thus, the resulting graph only has edges between test labels, and possibly a combination of singleton potentials on nodes (y_i^*) and pairwise potentials on edges (y_i^*, y_j^*) .

Once all of the potentials have been assigned, we then define a joint probability distribution over node labelings in the same manner as a Markov random field, by taking the product form

$$P(y_1^*, \dots, y_m^*) = \frac{1}{Z} \prod_{i=1}^m \left[\prod_{j'} \psi_{y_{j'}}(y_i^*) \right] \left[\prod_{j>i} \psi(y_i^*, y_j^*) \right]$$

and normalizing by an appropriate constant Z .

Computing the labeling Finally, given a joint probability distribution defined by the Markov random field, our goal is to compute the joint test pattern labeling that has maximum probability. (Since we are only interested in computing the maximum probability assignment, we can ignore the normalization constant above.) Depending on which edge model we use, there are different implications.

First, assuming model (i) (test labels only connect to training labels), there are no pairwise potentials and the Markov random field becomes completely factored. In this case, computing the maximum probability assignment is easy and can be determined independently for each test pattern. Essentially, removing test-test edges reduces our technique to a classical method in which each test pattern is classified independently. Here the learned coordination model

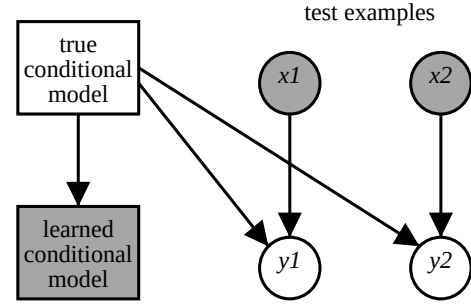


Figure 1: In an iid setting, the true test labels y_1 and y_2 are independent given the true conditional model. However, they are not independent given a learned estimate of the model.

$P(y_i^* y_j | \mathbf{x}_i^* \mathbf{x}_j, \phi)$ plays the role of a generalized similarity measure for classifying y_i^* in terms of $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$. The only difference is that the coordination model is learned in an earlier training phase, rather than being fixed beforehand.

The remaining cases (ii) and (iii) are more difficult because they introduce edges between test labels, which causes the labels to become dependent. Surprisingly, we have found that exploiting test label dependence can actually improve classification accuracy, even when the test data is known to be iid. (This is one of the main points of the paper.) For these models, computing the maximum probability assignment is hard, because the graph can contain many loops. To cope with the problem of performing probabilistic inference in a complex graphical model, we use loopy belief propagation to efficiently compute an approximate solution [Murphy *et al.*, 1999]. Below we find this gives adequate results.

3 Rationale and Discussion

Before presenting our experimental results, it is important to explain the rationale behind our technique and suggest why coordinated classification even makes sense in an iid setting.

Given the assumption that the training and test data are independent, we are proposing to predict test labels by building a graph, asserting joint potentials over pairs of labels (from a learned coordination classifier), and using belief propagation to make dependent predictions. Why does it make sense to make dependent predictions of iid labels? It turns out that this approach is justified even when taking the independence assumptions into account. Figure 1 illustrates the basic argument. In a standard machine learning setting, it is indeed true that, given the correct model for generating iid data, the label y_1 for an input pattern $\mathbf{x}_1$ is independent of the label y_2 for another pattern $\mathbf{x}_2$. However, note that this requires knowledge of the correct model (or at least its correct structure), which is rarely the case in classification learning. Instead, given only an *estimate* of the true model obtained from training data, y_1 and y_2 remain dependent, as Figure 1 clearly shows. That is, in the context of supervised learning it is generally the case that test labels are *dependent* given a learned model. In fact, it is obvious that supervised learning algorithms correlate the labels on the training data. Our observation is simply that the same principle can also be applied to test data.

Although using a relational technique for an iid problem might still appear awkward, it has a well known precedent in machine learning research: transductive learning [Vapnik, 1998; Zhu *et al.*, 2003]. In transduction, the learner knows the set of test patterns beforehand, and exploits this knowledge to make predictions that are ultimately dependent. In fact, this idea has been exploited in recent approaches to semi-supervised learning using Markov random fields [Zhu *et al.*, 2003]. What we are proposing is a general framework for extending standard probabilistic learning algorithms to be transductive in a similar fashion.

Our method can be further motivated by noting that it is a natural extension of standard ideas in supervised iid classification. As observed, learning a coordination classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ is a natural generalization of learning methods that use a similarity measure $k(\mathbf{x}_i, \mathbf{x}_j)$ to classify test examples $\mathbf{x}_i^*$ based on similarities $k(\mathbf{x}_i^*, \mathbf{x}_1), \dots, k(\mathbf{x}_i^*, \mathbf{x}_n)$ to the training patterns. In fact, this corresponds to our graph choice (i) above, which only connects test labels to training labels. Coordination classification extends the standard similarity based approach by first learning how patterns predict dependencies between the labels (using standard methods applied in a novel way), and then correlating test predictions over a graph. Although there has indeed been recent work on *learning* kernels for classification [Lanckriet *et al.*, 2004], as well as *transductive* learning with kernels [Xu *et al.*, 2004], thus far these formulations have remained hard to extend and apply in practice.

Another interesting view of coordination classification is that it is a novel form of ensemble method. That is, the label for a test pattern $\mathbf{x}_i^*$ is computed by a combination of votes from multiple predictors associated with different test (and training) patterns $x_j^{(*)}$. In fact, even remotely connected patterns can influence a classification via belief propagation.

As an ensemble method, coordination classification has some useful features. First, it only requires the training of a single base classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$, rather than multiple base classifiers trained from perturbed data. Second, as with boosting and bagging, coordination classification increases the representational capacity of an original (univariate) classifier. That is, given a classifier representation for a single label $P(y_i | \mathbf{x}_i, \theta)$, as mentioned previously, a coordination classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$ doubles the number of input features and squares the number of output classes. In addition, the prediction of a test label can, in principle, depend on all training and test patterns. Of course, simply increasing the representational capacity of a base classifier increases the risk of overfitting. However, the advantage of ensemble methods is that the resulting classifier, although more complex, is “smoothed” by a principled form of model combination, which helps avoid overfitting while exploiting added representational complexity. That is, the process of model combination can be used to reduce the variance of the learned predictor. In our case, we base our model combination principle on inference in a Markov random field. We will see below that, in fact, coordination classification is competitive as an ensemble technique.

The biggest drawback of coordination classification is the need to perform probabilistic inference (via loopy belief

propagation) in order to label test instances. However, we have still found the method to be robust to approximations, like running only a single iteration of loopy belief propagation, or even just taking local votes or products.

4 Experimental Results

We implemented the proposed coordination classification technique for a few different forms of probabilistic classifiers and using various standard iid data sets. Our intent was to determine whether the approach indeed had merit and was also robust to alterations in classifiers and data sets. Our experiments were conducted on standard two-class benchmark data sets from the UCI repository. The data sets used were: 1. australian, 2. breast, 3. chess, 4. cleve, 5. corral, 6. crx, 7. diabetes, 8. flare, 9. german, 10. glass2, 11. heart, 12. hepatitis, 13. mofn-3-7, 14. pima, and 15. vote. All of our experimental results were obtained by 5-fold cross validation, repeated 10 times for different randomizations of the graph structures. The tables and plots report averages of these results, with standard deviations included in the tables.

Table 1 and Figure 2 show the results of our first experiment. In this case, we implemented a standard logistic regression model, using unaltered input features, to learn a base coordination classifier $P(y_i y_j | \mathbf{x}_i \mathbf{x}_j, \phi)$. Classification was performed by running loopy belief propagation until the test labels stabilized (usually after 2 to 8 iterations). In the first experiment we used a graph over test labels only, to determine whether introducing label dependency would have any beneficial effect (see “edge” results). Here we subsampled test-test edges uniformly at random for an overall density of 18 edges per test example. Table 1 and Figure 2 show the resulting misclassification error obtained by coordination classification in comparison to learning a standard logistic regression model, $P(y_i | \mathbf{x}_i, \theta)$. Here we see a notable reduction in overall misclassification error (19%→16%), with a significant improvement on some data sets (Breast, -10%, Diabetes, -6%, MofN, -11%, Pima, -6%), and a minor increase on two data sets (Cleve, +1%, and Corral, +1%).

Table 1 also compares the error of coordination classification to *boosting* the base logistic regression model $P(y_i | \mathbf{x}_i, \theta)$. Here we used 18 rounds of Adaboost [Freund and Schapire, 1996; 1997], thereby combining approximately the same number of votes per test pattern as coordination classification. This experiment shows that as an ensemble method, coordination classification performs competitively in this case. An advantage of coordination classification is that it only needs to learn a single base classifier, as opposed to the multiple training episodes required by boosting. The need to run loopy belief propagation on the output labels is a disadvantage however.

To investigate the robustness of the method, we repeated the previous experiments using a different base classifier. Table 2 and Figure 3 show the results of an experiment using naive Bayes instead of logistic regression as the base classification method. Here the results are not as strong as the first case we tried, although they are still credible. Note that boosting obtains a few larger improvements, but also larger losses. Classification coordination appears to be fairly stable.

Table 1: A comparison of average misclassification error (%) on UCI data using **logistic regression** as a base model. Δ = average improvement over base.

	base	boosted	$\Delta \pm std$	edge	$\Delta \pm std$
australian	15.1	14.5	-0.6 ± 0.4	14.2	-0.9 ± 0.9
breast	14.5	14.9	0.4 ± 0.3	4.2	-10.3 ± 2.4
chess	7.6	8.3	0.7 ± 0.2	7.6	0 ± 0.1
cleve	15.3	14.9	-0.4 ± 0.6	16.6	1.3 ± 1.1
corral	8.8	8.8	0 ± 0.0	9.9	1.1 ± 0.9
crx	16.5	16.3	-0.2 ± 0.4	14.4	-2.1 ± 0.8
diabetes	31.2	31.3	0.1 ± 0.1	24.8	-6.4 ± 1.1
flare	17.4	17.5	0.1 ± 0.5	17.5	0.1 ± 1.3
german	25.8	25.9	0.1 ± 0.1	24.8	-1.0 ± 1.0
glass2	29.4	27.5	-1.9 ± 1.3	28.8	-0.6 ± 4.7
heart	16.3	15.9	-0.4 ± 0.4	15.9	-0.4 ± 1.9
hepatitis	17.5	17.5	0 ± 0.0	14.1	-3.4 ± 3.0
mofn-3-7	28.6	28.6	0 ± 0.0	17.3	-11.3 ± 0.8
pima	30.9	30.5	-0.4 ± 0.3	24.9	-6.0 ± 0.8
vote	6.7	6.7	0 ± 0.8	5.8	-0.9 ± 0.8
average	18.8	18.6	-0.2 ± 0.4	16.1	-2.7 ± 1.4

Table 2: A comparison of average misclassification error (%) on UCI data using **naive Bayes** as a base model. Δ = average improvement over base.

	base	boosted	$\Delta \pm std$	edge	$\Delta \pm std$
australian	13.8	16.2	2.4 ± 1.6	14.2	0.4 ± 0.4
breast	2.6	5.0	2.4 ± 1.1	2.7	0.1 ± 0.1
chess	20.1	8.2	-11.9 ± 3.1	19.1	-1.0 ± 0.6
cleve	16.3	17.0	0.7 ± 0.4	16.4	0.1 ± 0.2
corral	13.6	13.6	0 ± 5.5	14.2	0.6 ± 0.7
crx	14.6	16.6	2.0 ± 0.9	14.4	-0.2 ± 0.2
diabetes	22.6	22.6	0 ± 0.0	22.6	0 ± 0.2
flare	16.8	16.7	-0.1 ± 0.1	17.0	0.2 ± 0.4
german	25.5	26.3	0.8 ± 0.7	25.1	-0.4 ± 0.6
glass2	15.6	11.9	-3.7 ± 1.2	13.9	-1.7 ± 1.2
heart	15.9	17.0	1.1 ± 1.3	16.1	0.2 ± 0.2
hepatitis	13.8	15.0	1.2 ± 6.1	9.3	-4.5 ± 1.2
mofn-3-7	14.2	0.0	-14.2 ± 0.6	1.0	-13.2 ± 0.8
pima	21.8	21.7	-0.1 ± 0.1	22.2	0.4 ± 0.4
vote	9.9	5.5	-4.4 ± 1.8	9.8	-0.1 ± 0.2
average	15.8	14.2	-1.6 ± 1.6	14.5	-1.3 ± 0.5

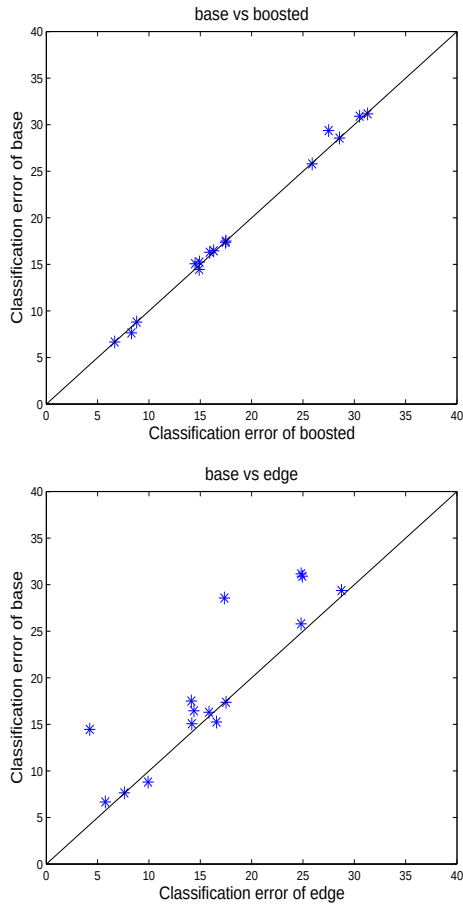


Figure 2: A comparison of average misclassification error on UCI data sets using logistic regression. **Top plot:** base model versus boosted logistic regression. **Bottom plot:** base model versus “edge”-based coordination classification.

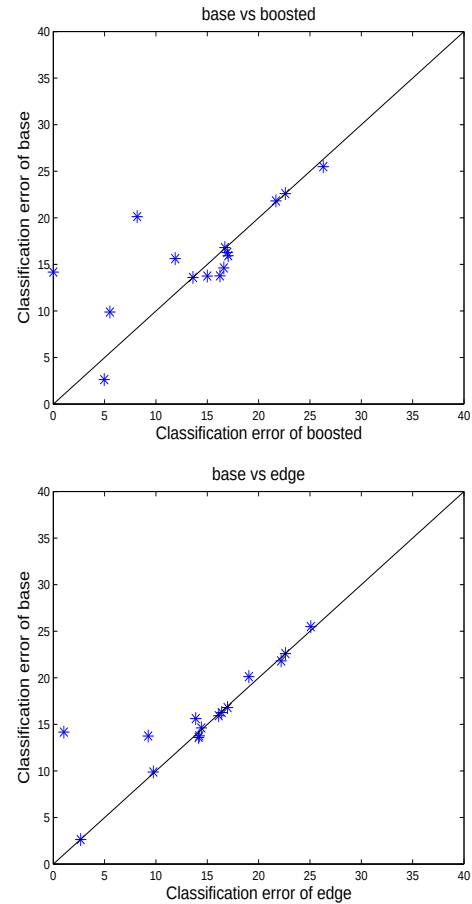


Figure 3: A comparison of average misclassification error on UCI data sets using naive Bayes. **Top plot:** base model versus boosted naive Bayes. **Bottom plot:** base model versus “edge”-based coordination classification.

The above experiments were run using only edges between test labels. To compare to standard approaches for iid data, we repeated the experiments using only edges between test and training labels, hence decoupling the test labels and eliminating the need for belief propagation (as discussed in Section 2.2). In this case, test labels are predicted independently. Table 3 and Figure 4 (top) still show, however, that this approach generally improves the base logistic regression classifier $P(y_i|x_i, \theta)$ (see “node” results). We conclude that correlating the test labels appears to be a beneficial idea, even in an iid setting. The marginal improvement of label dependence, although positive, might be secondary to the benefit of learning a similarity measure.

All of the previous results were obtained by subsampling edges at a density of 18 edges per test label. To test the sensitivity of our results to the edge density, we repeated the first experiment (test-test edges) using different edge densities. Figure 5 shows that the performance of coordination classification does not appear to be sensitive to edge density.

Finally, we experimented with the combined edge approach (iii) which randomly mixed test-test edges and test-train edges yielding the results of Table 3 and Figure 4 (bottom). The results in this case do not appear to surpass the performance of using test only edges (see “mix” results).

5 Extensions

All of our results so far have involved probabilistic classifiers whose output is a conditional distribution over the label y given the input pattern $\mathbf{x}$. Of course, many of the most important classification learning technologies, such as decision trees and support vector machines do not naturally produce probabilistic classifications over the class label (although they can be extended in various ways to do this [Platt, 2000]). This raises the obvious question of generalizing our technique to consider nonprobabilistic classifiers.

It is always possible to extend a classification learning method to learn to predict label pairs (y_i, y_j) given paired input patterns $(\mathbf{x}_i, \mathbf{x}_j)$. The difficulty is combining several paired predictions to render a sensible classification for a single test pattern. A convenient way to do this would be to convert the predicted outputs to nonnegative potentials over labelings. However, rather than do this, we tried the simpler alternative of combining pair classifications by a simple vote to classify a single test pattern $\mathbf{x}_i^*$. This is a less powerful combination method than loopy belief propagation, but requires fewer extensions to existing methods to test the coordination classifier idea in these cases.

To test this simple idea, we conducted an experiment on the same UCI data using a neural network classifier. Specifically we used a feedforward neural network with one hidden layer and logistic activation functions. The base neural network used one output unit, whereas the pairwise neural network used four output units (two units to indicate the class of each of the two input vectors) and double the number of input units. In each case the number of hidden units was set to 20, subject to the constraint that $(n_{in} + n_{out}) \times n_{hidden} \leq \text{train size}/2$. We trained the networks to minimize cross entropy error using the quasi-Newton method from Netlab

Table 3: *Alternative comparison of average error (%) on UCI data using logistic regression as a base model. Δ = average improvement over base.*

	base	node	$\Delta \pm std$	mix	$\Delta \pm std$
australian	15.1	14.1	-1.0 ± 0.9	14.2	-0.9 ± 0.9
breast	14.5	3.8	-10.7 ± 2.4	3.8	-10.7 ± 2.5
chess	7.6	8.1	0.5 ± 0.3	7.9	0.3 ± 0.2
cleveland	15.3	16.6	1.3 ± 1.3	16.8	1.5 ± 1.4
corral	8.8	11.2	2.4 ± 1.2	10.3	1.5 ± 0.8
crx	16.5	14.9	-1.6 ± 0.8	14.7	-1.8 ± 0.9
diabetes	31.2	24.7	-6.5 ± 1.3	24.9	-6.3 ± 1.3
flare	17.4	17.6	0.2 ± 1.2	17.8	0.4 ± 1.1
german	25.8	24.7	-1.1 ± 0.9	24.8	-1.0 ± 0.9
glass2	29.4	29.7	0.3 ± 5.0	29.4	0 ± 4.6
heart	16.3	15.7	-0.6 ± 2.0	15.9	-0.4 ± 1.8
hepatitis	17.5	15.0	-2.5 ± 3.4	14.0	-3.5 ± 2.8
mofn-3-7	28.6	25.1	-3.5 ± 0.2	23.6	-5.0 ± 0.4
pima	30.9	25.1	-5.8 ± 0.8	24.5	-6.4 ± 1.0
vote	6.7	6.0	-0.7 ± 0.8	5.9	-0.8 ± 0.7
average	18.8	16.8	-2.0 ± 1.5	16.6	-2.2 ± 1.4

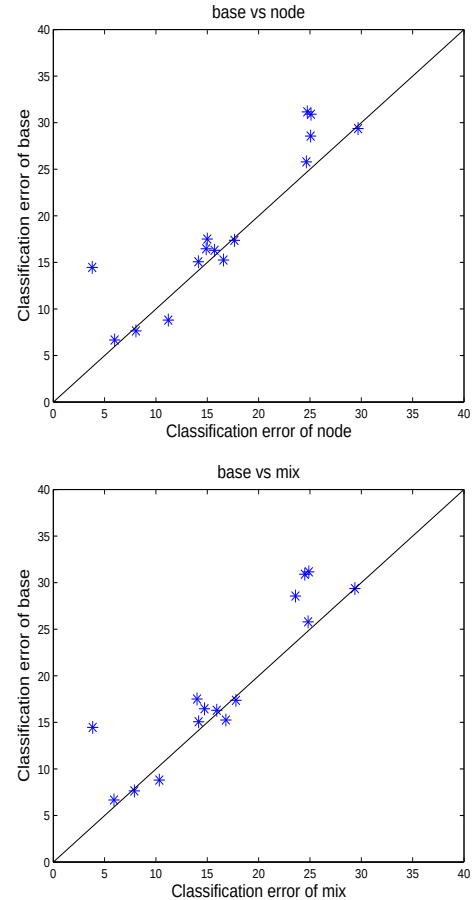


Figure 4: *Alternative comparison on UCI data using logistic regression. **Top plot:** base model versus “node”-based coordination classification. **Bottom plot:** base model versus a mix of “edge” and “node” based coordination classification.*

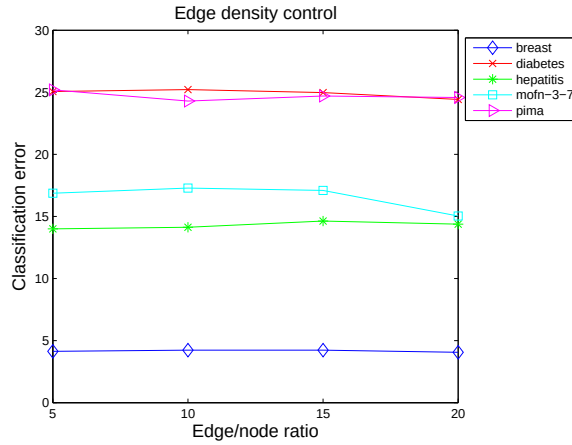


Figure 5: Average misclassification error of “edge”-based coordination classification using logistic regression, comparing different ratios of edges to the number of test patterns.

[Nabney, 2001] (www.ncrg.aston.ac.uk/netlab).

Once a pairwise neural network classifier was learned, we classified test examples according to the previous “edge” model, again by building a random graph between test labels (using an average of 18 edges per test label as before), using the learned coordination neural network to make hard predictions for each edge, and then combining the edge predictions using a simple vote to classify each test example. Table 4 (“edge”) and Figure 6 show the results of this experiment. Surprisingly, we still obtain a slight overall reduction in misclassification error over the base level neural network classifier, while again competing well against boosting.

Although encouraging, our results are not as positive in every case. We also conducted a simple experiment with decision trees [Quinlan, 1993] as the base coordination classifier, once again combining these predictions with a simple vote to label specific test patterns. Unfortunately, we did not observe an overall improvement over the base decision tree classifier in this case; see Figure 7. This result suggests that a more powerful model combination idea might be required to achieve robust improvements more generally.

6 Conclusion

We have proposed a novel classification learning strategy that coordinates the prediction of test labels on a graph over the data. The coordination classification idea can be used to extend any probabilistic classification approach quite naturally, and even seems to be applicable to nonprobabilistic techniques as well (although more research needs to be done).

One insight behind the technique is that making correlated predictions of test labels is justified, even advantageous, in the standard iid setting. This fact has often been overlooked in classification learning, therefore we believe it to be a point worth emphasizing. The ability to learn and predict coordinated test labels, combining them with probabilistic inference, provides a flexible new tool for improving classification accuracy on iid data.

Table 4: A comparison of average misclassification error (%) on UCI data using a **neural network** as a base model. Δ = average improvement over base.

	base	boosted $\Delta \pm std$	edge $\Delta \pm std$
australian	19.3	16.5 -2.8 ± 0.7	15.4 -3.9 ± 0.4
breast	4.0	4.4 0.4 ± 0.6	4.1 0.1 ± 0.4
chess	3.5	3.6 0.1 ± 0.2	6.7 3.2 ± 0.9
cleveland	22.9	20.0 -2.9 ± 1.4	18.8 -4.1 ± 1.3
corral	0	0 0 ± 0.0	0 0 ± 0.0
crx	20.6	18.5 -2.1 ± 0.9	15.9 -4.7 ± 1.9
diabetes	27.6	28.1 0.5 ± 0.8	25.0 -2.6 ± 0.9
flare	19.4	21.4 2.0 ± 0.5	19.7 0.3 ± 0.4
german	28.9	26.4 -2.5 ± 0.6	26.4 -2.5 ± 1.0
glass2	19.3	16.3 -3.0 ± 1.5	19.8 0.5 ± 1.2
heart	22.0	21.1 -0.9 ± 2.1	18.8 -3.2 ± 1.0
hepatitis	15.0	13.8 -1.2 ± 3.5	13.5 -1.5 ± 2.0
mofn-3-7	0	0 0 ± 0.0	0 0 ± 0.0
pima	26.7	29.0 2.3 ± 0.8	25.0 -1.7 ± 0.6
vote	6.0	6.7 0.7 ± 0.9	6.2 0.2 ± 0.4
average	15.7	15.0 -0.7 ± 1.0	14.4 -1.3 ± 0.8

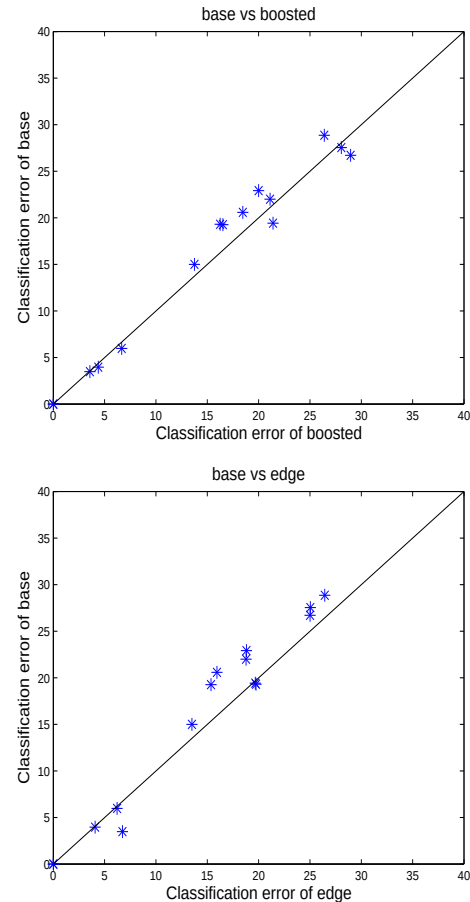


Figure 6: A comparison of average misclassification error on UCI data sets using a neural network. **Top plot:** base model versus boosted neural network. **Bottom plot:** base model versus “edge”-based coordination classification.

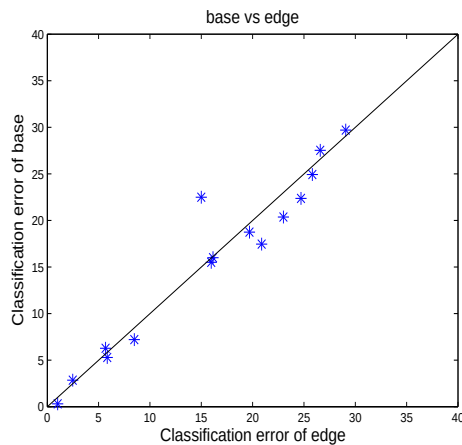


Figure 7: A comparison of average misclassification error on UCI data sets using C4.5 as the base classifier.

One idea for future work that we are considering is to extend the notion of a pairwise edge classifier to a more general clique classifier. We are also investigating alternative principles for defining potentials on label pairs to see perhaps if there are other combination ideas that work more effectively. Finally, we are also investigating whether combining standard “single node” classifiers with our generalized “edge” predictors might lead to further accuracy improvements.

Acknowledgments

Research supported by the Alberta Ingenuity Centre for Machine Learning, NSERC, MITACS, and the Canada Research Chairs program.

References

- [Anthony and Bartlett, 1999] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge, 1999.
- [Freund and Schapire, 1996] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 148–156, 1996.
- [Freund and Schapire, 1997] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.
- [Friedman et al., 1997] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:121–163, 1997.
- [Getoor et al., 2001] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI01 Workshop on Text Learning*, 2001.
- [Getoor et al., 2002] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. In *Journal of Machine Learning Research*, volume 3, pages 679–707, 2002.
- [Greiner and Zhou, 2002] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *Proceedings of the 18th Annual National Conference on Artificial Intelligence (AAAI)*, pages 167–173, 2002.
- [Hastie et al., 2001] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
- [Lafferty et al., 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 282–289, 2001.
- [Lanckriet et al., 2004] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
- [Murphy et al., 1999] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, 1999.
- [Nabney, 2001] I. Nabney. *NETLAB: Algorithms for Pattern Recognition*. Springer, New York, 2001. <http://www.ncrg.aston.ac.uk/netlab>.
- [Neal, 1996] R. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.
- [Platt, 2000] J. Platt. Probabilities for SV machines. In A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.
- [Quinlan, 1993] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.
- [Taskar et al., 2002] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2002.
- [Taskar et al., 2003] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS)*, pages 25–32, 2003.
- [Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- [Xu et al., 2004] L. Xu, B. Larson, J. Neufeld, and D. Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 1537–1544, 2004.
- [Zhu and Hastie, 2001] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems 14 (NIPS)*, pages 1081–1088, 2001.
- [Zhu et al., 2003] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 912–919, 2003.

Generalization Bounds for Weighted Binary Classification with Applications to Statistical Verification

Vu Ha Tariq Samad

Honeywell Labs

3660 Technology Dr.

Minneapolis, MN 55418

{vu.ha,tariq.samad}@honeywell.com

Abstract

We describe an approach to statistically verifying complex controllers. This approach is based on deriving practical Vapnik-Chervonenkis-style (VC) generalization bounds for binary classifiers with *weighted* loss. An important case is deriving bounds on the probability of false positive. We show how existing methods to derive bounds on classification error can be extended to derive similar bounds on the probability of false positive, as well as bounds in a decision-theoretic setting that allows tradeoffs between false negatives and false positives. We describe experiments using these bounds in statistically verifying computational properties of an iterative controller for an Organic Air Vehicle (OAV).

1 Introduction

The computational requirements for high-performance complex control algorithms can vary considerably. The variation arises because the computation depends on a number of factors such as the sensed/estimated state of the system under control, environmental disturbances, and the operational mode of the system. Furthermore, the variation in general can not be determined analytically. In hard real-time systems where the computation must complete in time for a command to be issued to the actuator at the next sample instant, these uncertainties pose a significant challenge. This is the reason that PID (Proportional-Integral-Derivative) controllers, with their deterministic execution time, are still the preferred choice in many applications despite their lesser performance.

In order to bring practical acceptance to high-performance complex control algorithms, we propose a compromise that makes use of both types of controller algorithms. High-performance algorithms will be used within a *safe operational envelope* (SOE) where they are guaranteed to complete within the allocated time. Outside of the SOE, lower performance and computationally simpler algorithms will be used. The SOE is determined based on simulation data, and hence is only *safe with some statistical guarantees*.

The problem of identifying the SOE is a binary classification problem where a false negative merely means a conservative use of a low performance controller and a false positive

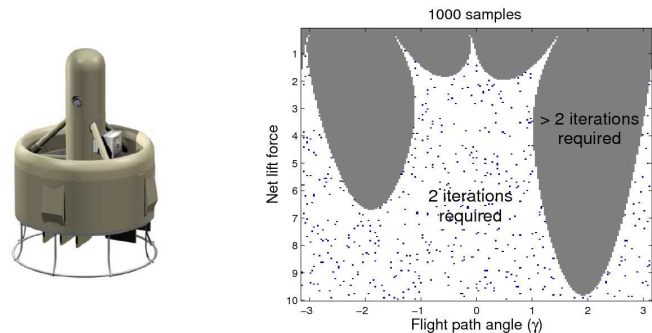


Figure 1: The Organic Air Vehicle (left), and the complexity of an iterative equilibrium angle-of-attack computation as determined by two factors: The flight path angle γ and the net lift force $\frac{qS}{mg}$ (right). The computational complexity here is equated with the number of iterations. The safe operating envelope consists of those inputs that require at most two iterations. The decision boundary is empirically drawn using 1000 random samples.

may have drastic consequences such as loss of the vehicle. Thus we would like to obtain an SOE that has some statistical guarantee that it is indeed safe, i.e., a classifier with provably small *probability of false positive*. This however is not the only criterion, for otherwise the trivial classifier that classifies everything as negative, namely the current state of the art, would be the top candidate. Instead, the goal is to push the boundaries of the SOE as far as possible while keeping a cap on the probability of false positive.

As an example, let us consider the computational property of a high-performance controller for an OAV (Figure 1). The OAV has a ducted fan propulsion unit, with control provided by movable vanes in the propwash. The vanes are situated in the propulsion airflow and consequently the interactions between the propulsion and the control surfaces are highly non-linear. The trim calculation for the OAV is an iterative algorithm whose computational time depends on several factors [Elgersma and Morton, 2000]. We are interested in conditions under which this calculation can be reliably used.

Our approach is based on statistical learning theory (SLT). Specifically, we derive statistical guarantees for SOEs using Vapnik-Chervonenkis-style generalization bounds for classification problems with *weighted* loss, of which the classification problem with false positive loss is a special case. We are

interested *practical* bounds—bounds that are asymptotically competitive *and* have small pre-constants. While the SLT literature contains a vast collection of VC-style bounds, the majority of these bounds are stated and proved only for the *probability of misclassification* and only a few are directly applicable to our problem. Furthermore, SLT bounds are often derived with little emphasis on obtaining optimal pre-constants. In addition to the emphasis on finding small pre-constants, our analysis has two unique aspects. First, we assume that it is possible to achieve small empirical loss (which is true for the case of false positive loss). Second, our analysis is upper-tail-oriented, as we are only interested in deriving *upper* bounds for the expected loss.

2 Preliminaries

Notations: We use the symbols $\mathbb{P}$, $\mathbb{E}$, and $\mathbb{V}$ to denote the probability, expectation, and variance, respectively. $\sigma = \{\sigma_i\}_{i=1}^n$ denotes a Rademacher sequence—a sequence of independent, symmetric $-1/1$ -valued random variables.

Let X and Y be non-empty sets and $Z = X \times Y$. A pair of $(x, y) \in X \times Y$ is denoted as z . Let (Z, μ) be a fixed probability measure. A *training set* is a finite sample $S_n = \{z_i\}_{i=1}^n = \{(x_i, y_i)\}_{i=1}^n \in Z^n$ drawn independently according to μ . The probability and expectation with respect to S_n are written as $\mathbb{P}_n$ and $\mathbb{E}_n$. A *hypothesis space* is a set H of functions from X to Y . A *loss function* is a function $l : Y \times Y \rightarrow \mathbb{R}$. The loss of a hypothesis $h \in H$ on z is $l(h, z) = l(h(x), y)$. The *expected loss* of h is $l(h) = \mathbb{E}l(h, z)$. The *empirical loss* of h on the training set S_n is $l_n(h) = l(h, S_n) = \frac{1}{n} \sum_{i=1}^n l(h, z_i)$. We assume that all loss functions have range $[0, 1]$. When Y is finite, we have a *classification problem*. When $|Y| = 2$, the classification is *binary*, and let $Y = \{-1, 1\}$. Here we only deal with binary classification.

If we assume that correct classifications incur zero loss, i.e. $l(-1, -1) = l(1, 1) = 0$, then what emerges is a loss function that we refer to as *weighted classification error*, defined as:

$$l^{(\rho)}(y, y') = \begin{cases} 0 & \text{if } y = y' \\ \rho & \text{if } y = -1, y' = 1 \text{ (false negative)} \\ 1 & \text{if } y' = -1, y = 1 \text{ (false positive)} \end{cases}$$

Here ρ is a number between 0 and 1. The idea is that false positives (bad errors) are more costly than false negatives (good errors). When $\rho = 1$, i.e. when no difference is made between the two types of errors, the loss is called the *misclassification error*. When $\rho = 0$, $l^{(0)}(h)$ is the probability of the classifier h making a false positive error.

Given a training set S_n , a hypothesis $h \in H$, and a loss function l , how can we bound the expected loss $l(h)$? Statistical learning theory (SLT) provides a probabilistic answer to this question. A typical SLT result of the form $\mathbb{P}_n(l(h) > \epsilon) < \delta$ or, succinctly, $l(h) <_\delta \epsilon$ provides an upper bound ϵ on $l(h)$ with confidence at least $1 - \delta$, where ϵ and δ are positive, reasonably small numbers. The upper bound ϵ typically depends on δ , the sample size n , the empirical loss $l_n(h)$, and a complexity measure of the hypothesis space H . The most important complexity measure in SLT is *VC dimension*. Let $\mathcal{C} \subseteq 2^X$ be a set of subsets of X . Note that H , as a set of binary classifiers on X , is an example of such sets. For any

$A \subseteq X$, define $\mathcal{C} \cap A = \{C \cap A : C \in \mathcal{C}\}$. The *growth function* of $\mathcal{C}$ is defined as

$$\Delta_k(\mathcal{C}) = \max\{|\mathcal{C} \cap A| : A \subseteq X, |A| = k\}.$$

Clearly, $\Delta_k(\mathcal{C}) \leq 2^k$. The *VC Dimension* of $\mathcal{C}$ is defined as

$$d(\mathcal{C}) = \sup\{k \in \mathbb{Z} : \Delta_k(\mathcal{C}) = 2^k\}.$$

Let $d = d(H)$. We assume that $d < \infty$.

Theorem 2.1. [Vapnik, 2000, Equations 3.26]

$$l(h) <_\delta l_n(h) + \sqrt{\frac{1}{n} \left(d \ln \frac{2en}{d} + \ln \frac{4}{\delta} \right)}. \quad (2.1)$$

The bound (2.1) is obtained by applying a very general result of Vapnik to the special setting of learning binary classifiers with a specific loss function $l^{(\rho)}$. It is thus natural to ask if this bound can be improved. While many SLT results have been obtained that address this issue, the majority of are formulated and proved for $l = l^{(1)}$ only. One of the goals of this paper is to examine if these results can be extended to the loss functions $l^{(\rho)}$, $\rho \in [0, 1)$. It turns out that these results fall into two categories: those that are applicable for all $\rho \in [0, 1]$, and those that are applicable for $\rho = 0, 1$ only.

We conclude the preliminaries with the statement of the often-used Sauer's lemma.

Sauer's Lemma. [Sauer, 1972] $\Delta_n(H) \leq (en/d)^d$.

3 VC Dimension-Based Bounds

Suppose that h is a hypothesis with “small” empirical loss: $l_n(h) \leq \epsilon_1$. We would like to bound the probability that $l(h)$ is “large”: $l(h) > \epsilon$ for some $\epsilon > \epsilon_1$. This amounts to bounding $\mathbb{P}_n(Q)$ where Q is defined as

$$Q = \{S_n : \exists h : l_n(h) \leq \epsilon_1, l(h) > \epsilon\}.$$

There are two major approaches to do this: The classical approach of Vapnik and Chervonenkis [1971], and the approach based on abstract concentration inequalities developed by Talagrand and others.

3.1 The Classical Approach

The classical VC analysis begins with the observation that $\mathbb{P}_n(Q) \leq \mathbb{P}(R)/\mathbb{P}(R|Q)$ for any R that satisfies $\mathbb{P}(R|Q) > 0$. We then define R based on S_n and an additional independent sample whose size may or may not be equal to n . In this analysis we take the former approach: Let $S'_n = \{z_i\}_{i=n+1}^{2n}$ be an independent sample of size n , commonly referred to as the *ghost sample* and let $l'_n(h) = l(S'_n, h)$, the empirical loss on the ghost sample. Denote $S_{2n} = (S_n, S'_n)$. Let $0 \leq \epsilon_1 < \epsilon_2 \leq \epsilon$. Define R as

$$R = \{S_{2n} : \exists h : l_n(h) \leq \epsilon_1, l'_n(h) > \epsilon_2\}. \quad (3.1)$$

Upper bounding $\mathbb{P}_n(Q)$ now reduces to upper bounding $\mathbb{P}_{2n}(R)$ (the covering step) and lower bounding $\mathbb{P}_{2n}(R|Q)$ (the symmetrization step).

The Covering Step: Upper Bounding $\mathbb{P}_{2n}(R)$. Intuitively, $\mathbb{P}_{2n}(R)$ is small because if h has small empirical loss on a sample, its empirical loss on a ghost sample should also be small. We can change the definition of R in (3.1) as

$$R = \{S_{2n} : \exists h : l'_n(h) - l_n(h) > \eta, \text{ where } \eta = \epsilon_2 - \epsilon_1\}.$$

The next step uses the so-called *permutation technique*.

$$\begin{aligned}\mathbb{P}_{2n}(R) &= \mathbb{E}_{2n}(\exists h : l'_n(h) - l_n(h) > \eta) \\ &= \mathbb{E}_{2n} \mathbb{E}_\sigma \left(\exists h : \sum_{i=1}^n \sigma_i (l(h, z_{i+n}) - l(h, z_i)) > n\eta \right).\end{aligned}$$

Next, we fix $\mathcal{S}_{2n}$ and bound the inner expectation. Let

$$H_{2n} = \{h_{2n} = (h(x_1), \dots, h(x_{2n})) \mid h \in H\} \subseteq \{-1, 1\}^{2n}.$$

The mapping $h \mapsto h_{2n}$ is many-to-one. Denote $l_i(h_{2n}) = l(h, z_i)$, $1 \leq i \leq 2n$. The inner expectation can be written as

$$\mathbb{E}_\sigma \left(\exists h_{2n} \in H_{2n} : \sum_{i=1}^n \sigma_i (l_{i+n}(h_{2n}) - l_i(h_{2n})) > n\eta \right)$$

which, by the union bound, is bounded by

$$\sum_{h_{2n} \in H_{2n}} \mathbb{E}_\sigma \left(\sum_{i=1}^n \sigma_i (l_{i+n}(h_{2n}) - l_i(h_{2n})) > n\eta \right). \quad (3.2)$$

The cardinality of H_{2n} is at most $\Delta_{2n}(H)$, which is at most $(2en/d)^d$ by Sauer's lemma. For a fix $h_{2n} \in H_{2n}$, the summand in (3.2) can be written as

$$\mathbb{P}_\sigma \left(\sum_{i=1}^n \sigma_i (l_{i+n}(h_{2n}) - l_i(h_{2n})) > n\eta \right)$$

which, by Höeffding's right-tail inequality, is bounded by

$$\exp \left(\frac{-2n^2\eta^2}{4 \sum_i (l_{i+n}(h_{2n}) - l_i(h_{2n}))^2} \right) \leq \exp \left(-\frac{n\eta^2}{2} \right).$$

Thus we have arrived at the following result.

Lemma 3.1. [Vapnik and Chervonenkis, 1971]

$$\mathbb{P}_{2n}(l_n(h) \leq \epsilon_1, l'_n(h) > \epsilon_2) \leq \left(\frac{2en}{d} \right)^d \exp(-.5n(\epsilon_2 - \epsilon_1)^2).$$

When $\epsilon_1 \ll \epsilon_2$, it is possible to improve upon Lemma 3.1. The idea is, instead of bounding the probability that the *absolute discrepancy* $l'_n(h) - l_n(h)$ is large, we bound the probability that the *relative discrepancy* $\frac{l'_n(h) - l_n(h)}{\sqrt{l'_n(h) + l_n(h)}}$ is large. We

“weaken” the definition of R in (3.1) as

$$\begin{aligned}R &= \{ \mathcal{S}_{2n} : \exists h : l'_n(h) - l_n(h) > \eta' \}, \\ \eta' &= (\epsilon_2 - \epsilon_1) \sqrt{\frac{l'_n(h) + l_n(h)}{\epsilon_2 + \epsilon_1}} = \eta \sqrt{\frac{l'_n(h) + l_n(h)}{\epsilon_2 + \epsilon_1}}.\end{aligned}$$

Now, proceed identically as before, except that η is now replaced with η' , we arrive at the following results.

Lemma 3.2. [Vapnik and Chervonenkis, 1971]

$$\mathbb{P}_{2n}(l_n(h) \leq \epsilon_1, l'_n(h) > \epsilon_2) \leq \left(\frac{2en}{d} \right)^d \exp \left(\frac{-n(\epsilon_2 - \epsilon_1)^2}{2(\epsilon_2 + \epsilon_1)} \right).$$

Corollary 3.3. [Vapnik and Chervonenkis, 1971]

$$\mathbb{P}_{2n}(l_n(h) = 0, l'_n(h) > \epsilon_2) \leq \left(\frac{2en}{d} \right)^d \exp(-.5n\epsilon_2).$$

By considering the relative discrepancy, we have managed to insert the term $\epsilon_2 + \epsilon_1$, resulting in a tighter bound in Lemma (3.2). Further tightening is possible when $\epsilon_1 = 0$. Instead of using Hoeffding's inequality, Blumer *et al.* [1989] use a combinatorial argument that leads to the following improvement of Corollary 3.3.

Lemma 3.4. [Blumer *et al.*, 1989]

$$\mathbb{P}_{2n}(l_n(h) = 0, l'_n(h) > \epsilon_2) \leq \left(\frac{2en}{d} \right)^d \exp(-n\epsilon_2 \ln 2).$$

The Symmetrization Step: Lower Bounding $\mathbb{P}_{2n}(R|Q)$. To lower bound $\mathbb{P}_{2n}(R|Q)$, we can fix $\mathcal{S}_n$, ignore the condition $l_n(h) \leq \epsilon_1$, and bound the following conditional probability:

$$\mathbb{P}_{\mathcal{S}'_n}(\{\mathcal{S}'_n : \exists h : l'_n(h) > \epsilon_2\} | \exists h : l(h) > \epsilon),$$

or, equivalently, $\mathbb{P}_n(\{\exists h : l_n(h) > \epsilon_2\} | \exists h : l(h) > \epsilon)$.

Let h be a hypothesis such that $l(h) > \epsilon$. It suffices to lower bound $\mathbb{P}_n(l_n(h) > \epsilon_2)$. Intuitively, this quantity is large because the empirical loss should be large ($> \epsilon_2$) wherever the expected loss is large ($> \epsilon$). Since $\epsilon_2 < \epsilon$, we have

$$\begin{aligned}\mathbb{P}_n(l_n(h) \leq \epsilon_2) &\leq \mathbb{P}(l(h) - l_n(h) > \epsilon - \epsilon_2) \\ &\leq \exp \left(-\frac{6n(\epsilon - \epsilon_2)^2}{4(\epsilon - \epsilon_2) + 3} \right) := \iota(n, \epsilon, \epsilon_2),\end{aligned} \quad (3.3)$$

by Bernstein's left-tail inequality. Coupled this with Lemma 3.2, we obtain the following result.

Corollary 3.5.

$$\begin{aligned}\mathbb{P}_n(l_n(h) \leq \epsilon_1, l(h) > \epsilon) \\ \leq (1 \vee (1 - \iota(n, \epsilon, \epsilon_2))^{-1}) \left(\frac{2en}{d} \right)^d \exp \left(\frac{-n(\epsilon_2 - \epsilon_1)^2}{2(\epsilon_2 + \epsilon_1)} \right).\end{aligned}$$

In this inequality, the parameter ϵ_2 is unspecified, and we can minimize the bound over $\epsilon_2 \in (\epsilon_1, \epsilon)$.

When $\rho = 0, 1$, the loss function $l^{(\rho)}$ is binary, and $nl_n(h)$ is a binomial random variable with parameters n and $l(h)$. We can thus use several lower bounds on the right-tails of the binomial to obtain

$$n\epsilon > 1 \Rightarrow \mathbb{P}_n(\{l_n(h) > \epsilon/4\} | \exists h : l(h) > \epsilon) > 1/4 \quad (3.4)$$

$$n\epsilon > 2 \Rightarrow \mathbb{P}_n(\{l_n(h) > \epsilon/2\} | \exists h : l(h) > \epsilon) > 1/2 \quad (3.5)$$

For the case $\epsilon_1 > 0$, we can set $\epsilon_2 = \epsilon$ and combine (3.4) with Lemma 3.2 to obtain the following result.

Corollary 3.6. [Vapnik and Chervonenkis, 1971] For $l = l^{(\rho)}$, $\rho = 0, 1$,

$$l(h) <_\delta 2l_n(h) + \frac{4}{n} \left(d \ln \frac{2en}{d} + \ln \frac{4}{\delta} \right). \quad (3.6)$$

For the case $\epsilon_1 = 0$, we can set $\epsilon_2 = \epsilon/2$ and combine (3.5) with Lemma 3.4 to obtain the following result.

Corollary 3.7. [Blumer *et al.*, 1989] For $l = l^{(\rho)}$, $\rho = 0, 1$,

$$l_n(h) = 0 \Rightarrow l(h) <_\delta \frac{2}{n \ln 2} \left(d \ln \frac{2en}{d} + \ln \frac{2}{\delta} \right). \quad (3.7)$$

Shawe-Taylor *et al.* [1993] further improve (3.7), using an argument that uses a second sample $\mathcal{S}'_k$ of size k , and $\epsilon_2 = r\epsilon$, where $r = 1 - \sqrt{2/(\epsilon k)}$, $k = n(\epsilon r n/d - 1)$.

Theorem 3.8. [Shawe-Taylor *et al.*, 1993] For $l = l^{(\rho)}$, $\rho = 0, 1$,

$$\begin{aligned}n\epsilon > 4d \Rightarrow \mathbb{P}_n(l_n(h) = 0, l(h) > \epsilon) \\ \leq 2 \exp \left(2\sqrt{2d} - \epsilon n + d \ln \epsilon + 2d \ln \frac{en}{d} \right).\end{aligned} \quad (3.8)$$

Compared with (3.7), the sample complexity derived from (3.8) is smaller by a factor of $\frac{4}{\ln 2}(1 - \sqrt{\epsilon}) \approx 5.7$ for typical values ϵ (say, < 0.05).

We point out that Corollary 3.6, 3.7, and Theorem 3.8 were previously stated and proved for the loss function $l = l^{(1)}$ only. Our analysis extends them to the case $l = l^{(0)}$ (and shows that they do *not* hold when $0 < \rho < 1$). The covering argument remains the same, while the symmetrization argument uses the simple observation that $nl_n^{(\rho)}(h)$ is a binomial random variable with parameters n and $l(h)$, regardless of whether $\rho = 0$ or $\rho = 1$.

3.2 Talagrand's Method

Observe that $l(h) \leq l_n(h) + \sup_{h \in H} (l(h) - l_n(h))$. The supremum is a random function of $\mathcal{S}_n$, where changing a single element z_i results in a change of at most $1/n$, and thus is $<_\delta$ -bounded by $\mathbb{E}_n(\sup_{h \in H} (l(h) - l_n(h))) + \sqrt{\frac{1}{2n} \ln \frac{1}{\delta}}$ by McDiarmid's inequality [McDiarmid, 1997]. The next quantity to bound is the expectation of the supremum. This is accomplished using the concept of Rademacher average. Let G be a class of functions from Z to the reals $\mathbb{R}$. The *Rademacher average* of G is defined as $R_n G = R(G, \mathcal{S}_n, \sigma) = \sup_{g \in G} (\frac{1}{n} \sum_{i=1}^n \sigma_i g(z_i))$. In this analysis, the role of G is played by the loss class associated with H : $G = l_H = \{z \mapsto l(h, z) : h \in H\}$. The symmetrization inequality (e.g. [Bartlett et al., 2005]) states that $\mathbb{E}_n(\sup_{h \in H} (l(h) - l_n(h))) \leq 2\mathbb{E}R_n l_H$. Thus it remains to bound the Rademacher average. The technique is well-established and based on the concepts of *covering number*. Denote by $N(u, l_H, L_2(\mu^n))$ the u -covering number of l_H with respect to the metric $L_2(\mu^n)$.

Lemma 3.9. [Dudley, 1999]

$$\mathbb{E}_\sigma R_n l_H \leq \frac{12}{\sqrt{n}} \int_0^1 \sqrt{\ln N(u, l_H, L_2(\mu^n))} du. \quad (3.9)$$

The last piece of the puzzle reveals a bound on $N(u, l_H, L_2(\mu^n))$, defined based on the VC dimension d .

Lemma 3.10. [Haussler, 1995] For all $\mathcal{S}_n$:

$$N(u, l_H, L_2(\mu^n)) \leq e(d+1) \left(\frac{2e}{u^2} \right)^d. \quad (3.10)$$

Combining (3.9) and (3.10), with some algebra we obtain the following result.

Theorem 3.11. (Dudley-Haussler)

$$l(h) <_\delta l_n(h) + 30\sqrt{\frac{d}{n}} + \sqrt{\frac{\ln(1/\delta)}{2n}}. \quad (3.11)$$

The bound (3.11) is based on an analysis of the absolute discrepancy $l(h) - l_n(h)$. Its deviation term is $O(\sqrt{d/n})$, which should not come as a surprise. It is natural to ask if this approach can be used to analyze some form of relative discrepancy between the expected loss $l(h)$ and the empirical loss $l_n(h)$. Consider

$$\tilde{L}_n = \sup \left\{ \frac{l(h) - l_n(h)}{\sqrt{l(h)}} : h \in H, l(h) > 0 \right\}. \quad (3.12)$$

Simple algebra shows that for all $h \in H$, $l(h) \leq (1 + \tilde{L}_n)l_n(h) + \tilde{L}_n^2$. Consequently, an upper bound on $\tilde{L}_n$ can be translated into an upper bound on $l(h)$. Unlike the previous analysis, it appears that we can not use McDiarmid's bounded difference inequality, as the introduction of the term $\sqrt{l(h)}$ renders the "difference" unbounded. The solution to this problem originates from the work of Talagrand on abstract concentration inequalities and their applications to bounding the suprema of empirical processes [Talagrand, 1994, 1996]. Talagrand's inequalities were later improved using the so-called entropy method. The following version provides the best known bound.

Lemma 3.12. [Bousquet, 2002] Let $\mathcal{F}$ be a countable set of functions from Z to $\mathbb{R}$. Let $b = \sup_{f \in \mathcal{F}} (\sup(\mathbb{E}(f) - f))$, $v = \sup_{f \in \mathcal{F}} \mathbb{V}(f)$ and $B_n = B(\mathcal{S}_n) = \sup_{f \in \mathcal{F}} (\mathbb{E}(f) - \frac{1}{n} \sum_{i=1}^n f(z_i))$. Then for any $\alpha > 0$,

$$B_n <_\delta (1 + \alpha)\mathbb{E}_n(B_n) + \sqrt{\frac{2v \ln(1/\delta)}{n}} + \left(\frac{1}{3} + \frac{1}{\alpha} \right) \frac{b \ln(1/\delta)}{n}.$$

We now apply Lemma 3.12 with $f(z) = l(h, z)/\sqrt{l(h)}$, $b = v = 1$, to obtain

$$\tilde{L}_n <_\delta (1 + \alpha)\mathbb{E}_n(\tilde{L}_n) + \sqrt{\frac{2 \ln(1/\delta)}{n}} + \left(\frac{1}{3} + \frac{1}{\alpha} \right) \frac{\ln(1/\delta)}{n}.$$

We then proceed to bound $\mathbb{E}_n(\tilde{L}_n)$ with a technique from Massart [2000] that is referred to as *peeling* and the concept of sub-root functions. A function $\psi : (0, \infty) \rightarrow (0, \infty)$ is called *sub-root* if ψ is non-decreasing and $\psi(r)/\sqrt{r}$ is non-increasing. Any sub-root function $\psi(r)$ is known to have a unique fix-point r^* (i.e. $\psi(r^*) = r^*$) [Bartlett et al., 2005]. Now, suppose that ψ is a sub-root function with fixed point r^* such that

$$\mathbb{E}_n(\sup\{|l(h) - l_n(h)| : l(h) \leq r\}) \leq \psi(r), \forall r > 0.$$

Then we can show that $\forall \alpha > 0$, $\tilde{L}_n$ is $<_\delta$ -bounded by

$$(1 + \alpha) \left(\sqrt{r^*} \left(1 + \frac{e}{2} (1 + \ln(\frac{1}{r^*})) \right) \right) + \sqrt{\frac{2 \ln \frac{1}{\delta}}{n}} + \frac{(3 + \alpha) \ln \frac{1}{\delta}}{3\alpha n}.$$

The final step is to bound r^* using d . This can be done [Koltchinskii and Panchenko, 2000] by setting $\psi(r)$ to be Dudley's entropy integral. What we end up with is a $<_\delta$ bound on $l(h)$ that is $O(d \ln n/n)$, asymptotically comparable to those obtained using the classical approach such as Theorem 3.8, albeit with worse constants.

3.3 Summary of VC Dimension-Based Bounds

Given a sample $\mathcal{S}_n$ of size n and a hypothesis h that has empirical loss $l_n(h)$ on $\mathcal{S}_n$, what can we say about the expected loss $l(h)$ of h ? The results in this section provide several answers to this question. They all have the general form of "If $l_n(h)$ is small, then with high probability, $l(h)$ is small". The common assumption is that h is selected from a hypothesis space H with finite VC dimension d . In the general case when l is only assumed to have range $[0, 1]$, Corollary 3.5 seems most useful. When $l = l^{(\rho)}$, $\rho = 0, 1$, we can exploit several binomial tail inequalities to obtain much

simpler bounds. When $l_n(h) > 0$, Corollary 3.6 should be used. When $l_n(h) = 0$, Theorem 3.8 provides the best-known bound. These results are all based on the idea of uniform convergence of *relative* discrepancies (UCRD). Even Corollary 3.7 and Theorem 3.8 can be viewed as based on degenerate cases of UCRD, with special-purpose combinatorial arguments replacing the general-purpose Hoeffding's bound. Results that are based on uniform convergence of absolute discrepancies such as Lemmas 3.1 and Theorem 3.11 are not as useful for our purpose, as they have to cover situations that Vapnik and Chervonenkis [1971] refer to as *pessimistic cases*. Simply put, pessimistic bounds are loose since they need to account for hypotheses with expected losses close to 0.5. In contrast, we only need to concern ourselves with hypotheses with zero or small empirical losses.

The statistical/computational learning theory literature contains a vast collection of generalization bounds in the general case of function learning and in the special case of learning binary classifiers. To our knowledge no work has explicitly derived generalization bounds for binary classifiers with weighted error penalties as defined in this paper. The bounds in Section 3.1 originate from the seminal work of Vapnik and Chervonenkis [1971]. Our contribution here is the extensions to the case $\rho = 0$, and Corollary 3.5.

Talagrand's approach provides a completely different way to arrive at generalization bounds for all loss function $l^{(\rho)}, \rho \in [0, 1]$ that are asymptotically equivalent with classical bounds. This approach analyzes the mean (or median [Panchenko, 2002]) of the supremum of the (sometimes weighted) discrepancies between the expected and empirical losses using Talagrand's various concentration inequalities, completed invariably with the symmetrization inequality, Dudley's entropy integral bound, and Haussler's packing bound. The resulting bounds often have much larger constants and are not as useful for our non-asymptotic purpose.

4 Experiments

We now describe the applications of the bounds derived in Section 3 to our OAV experiment as described in the introduction. We identify four factors that affect the computational time of the iterative algorithm, and choose 4-dimensional axis-parallel hyper-rectangles as our hypothesis space. The VC dimension of this hypothesis space is 8, as it is known that the VC dimension of axis-parallel hyper-rectangles in $\mathbb{R}^m$ is $2m$. Thus for $\delta = .05, \epsilon = .05$, we need 34,000 samples using Theorem 2.1. But using Theorem 3.8, we need only 1810 samples. With 34,000 samples, if we set $\delta = .05$, ϵ can be as small as 0.0035. The improvement in generalization bound (ϵ) is about 14-fold and in samples complexity (n) is about 18-fold.

The search for the best hyper-rectangle in this experiment is rather simple. For each sampled input, we determine if the iterative algorithm converges. It turns out that in 32,114 instances (roughly 94%), the algorithm converges. Despite this high (empirical) rate of success, in current practice it still loses out to a PID-like controller with fixed deterministic computation time. Next, we randomly choose an axis-parallel 4-dim hyper-rectangle in the input ranges as a hypothesis. If

the hyper-rectangle contains a sampled point for which the algorithm does not converge (an unsafe point), then we eliminate that hyper-rectangle (since the guarantees are based on Theorem 3.8 and require zero false positives). Otherwise, we count the number of safe points that lie outside the hyper-rectangle (false negatives), and choose the hyper-rectangle that has the fewest number of false negatives. After looking at 10,000 random hyper-rectangles, we are able to come up with one that contains 18,616 safe points and no unsafe points. This hypothesis thus has $32,114 - 18,616 = 13,498$ false negatives. Note that the number of false negatives is still quite large. This is because we use hyper-rectangles which constitute a simple hypothesis space that does not approximate the decision surface very well (this SOE is nevertheless a big improvement over the trivial, empty SOE that is the current state of the art). The advantage to this is that the VC dimension is low, and thus only a small number of samples are required to obtain the statistical guarantee, which reads *"The found hyper-rectangle has probability of false positive bounded by 0.0035, and we have at least 95% confidence in this statement."*

There are a number of alternatives to the above procedure. For example, we can use Corollary 3.6 instead of Theorem 3.8, if the requirement of zero empirical loss is too restrictive. In the OAV example with 34,000 samples, this leads to a hypothesis with 111 false positives but only 3272 false negatives (a reduction of 75%!) while still maintaining 99% confidence that the probability of false positive is less than 0.01. Furthermore, we can replace the criterion "as few false negatives as possible" with other criteria, for example one that prefers hypotheses with large volumes. Finally, if we are willing to make a decision-theoretic tradeoff between false negatives and false positives (e.g. one false positive is as costly as one thousand false negatives), we can set $\rho = .001$ and apply Corollary 3.5.

5 Summary and Related Work

It has been said that the divide between SLT and practice is of Grand Canyon proportions, perhaps because VC bounds are often too loose to be useful in practice. This paper offers a counterargument in the form of an SLT-based approach to verifying complex controllers. We demonstrated this approach on a problem of significant industrial and military interest: Deriving a safe operating envelope for a complex control algorithm. This approach offers control engineers a principled way to increasingly replace low-performance, simple control algorithms with high-performance, complex ones while still maintaining a statistically high confidence in safety. A key to making this offer attractive lies in deriving practical VC-style generalization bounds for weighted binary classification (a problem that hitherto has not been given much attention). Our VC analysis, which builds upon standard VC analysis of unweighted binary classification, shows that such bounds are indeed possible. They are significantly better than a general bound by Vapnik. Our analysis precisely pointed to the place where the false negative penalty had an effect, namely the symmetrization argument. We have successfully applied this verification framework to several other

control applications, to be reported in an extended version of this paper. We expect these results to have applications outside of the controller verification problem.

From a practical point of view, SLT-based generalization bounds have been used mostly in the model selection problem (see e.g. [Bartlett et al., 2002]). Aside from this, they have also been used in several control systems applications, for example, in deriving randomized algorithms for robust control problems whose exact solution is NP-hard, and in the context of system identification [Vidyasagar, 2003, Chapter 11]. Also, machine learning researchers have long recognized the importance of learning classifiers with general loss function. The work in this area is generally referred to as *cost-sensitive learning* [Turney, 2000].

The bounds derived in Section 3 consist of the empirical loss and a VC confidence term that is independent of the probability measure μ and the particular sample $\mathcal{S}_n$. They are thus necessarily “loose” since they need to hold for “bad” distributions and “bad” samples $\mathcal{S}_n$. Recent research has focused on data-based measures of hypothesis space complexity such as Rademacher averages [Koltchinskii, 2001; Bartlett et al., 2005]. This direction relies on Talagrand’s approach as described in Section 3.2. Although this approach yields VC dimension-based bounds that have worse constants compared to bounds using the classical approach such as (3.6), it can be used to derive bounds based entirely on data (i.e. $\mathcal{S}_n$) without *a priori* information about the hypothesis space H (such as its VC dimension d). The following result is an example of such data-dependent bounds.

Theorem 5.1. [Bartlett et al., 2005, Corollary 6.2] *Let $l = l^{(\rho)}$, $\rho = 0, 1$. Let the random function $\hat{\psi}_n$ be defined on $(0, \sqrt{1/2})$ as*

$$\hat{\psi}_n(r) = 20 \sup_{\beta \in [\sqrt{2r}, 1]} \beta \mathbb{E}_\sigma R_n \{z \mapsto l(h, z) : l_n(h) \leq \frac{2r}{\beta^2}\} + \frac{26}{n} \ln \frac{3}{\delta}.$$

Then $\hat{\psi}_n$ is sub-root with fixed point $\hat{r}^*$, and for any $\alpha > 0$,

$$l(h) <_\delta (1 + \alpha) l_n(h) + 6 \frac{\alpha + 1}{\alpha} \hat{r}^* + \frac{16\alpha + 5}{\alpha n} \ln \frac{3}{\delta}.$$

Compared to similar result in Section 3.2, the present one is completely data-dependent: In the definition of $\hat{\psi}_n(r)$, the bound $2r/\beta^2$ is on the empirical (as opposed to the expected) loss $l_n(h)$, and the empirical Rademacher average $\mathbb{E}_\sigma R_n$ (as opposed to $\mathbb{E} R_n$) is used. Thus in theory we can obtain a bound without *a priori* knowledge of the complexity (such as the VC dimension) of the hypothesis space H or of the underlying probability measure μ . However, in practice, computing or estimating $\hat{\psi}_n$ and its fixed point r^* is far from easy, although Bartlett et al. [2005, Section 6] had made some initial progress in this direction.

Acknowledgements

This work was supported in part by the U.S. Defense Advanced Research Projects Agency (DARPA) and the U.S. Air Force Research Laboratory under Contract No. F33615-01-C-1848. We thank Michael Elgersma for help with the OAV experiment, and XuanLong Nguyen for many useful comments.

References

- P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Mach. Learn.*, 48(1-3):85–113, 2002.
- P.L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Annals of Statistics*, 2005. To appear.
- A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.
- O. Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *C. R. Acad. Sci. Paris, Ser. I*, 334:495–500, 2002.
- R. Dudley. *Uniform Central Limit Theorems*. Cambridge University Press, Cambridge, MA, 1999.
- M.R. Elgersma and B.G. Morton. Nonlinear six-degree-of-freedom aircraft trim. *Journal of Guidance, Control, and Dynamics*, 23(2):305–311, 2000.
- D. Haussler. Sphere packing numbers for subsets of Boolean n -cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory (A)*, 69:217–232, 1995.
- V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. *High Dimensional Probability II*, 47:443–459, 2000.
- V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, July 2001.
- P. Massart. Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, IX(2):245–303, 2000.
- C. McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic method for algorithmic discrete mathematics*, pages 195–248. Springer-Verlag, New York, 1997.
- D. Panchenko. Some extensions of an inequality of Vapnik and Chervonenkis. *Electronic Communications in Probability*, 7:55–65, 2002.
- N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory*, 13:145–147, 1972.
- John Shawe-Taylor, Martin Anthony, and N. L. Biggs. Bounding sample size with the Vapnik-Chervonenkis dimension. *Discrete Appl. Math.*, 42(1):65–73, 1993.
- M. Talagrand. Sharper bounds for gaussian and empirical processes. *Annals of Probability*, 22:20–76, 1994.
- M. Talagrand. New concentration inequalities for product spaces. *Inventiones Mathematicae*, 126(3):505–563, 1996.
- P. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, pages 15–21, Stanford University, California, 2000.
- V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.
- V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 2000.
- M. Vidyasagar. *Learning and Generalization, with Applications to Neural Networks*. Springer, second edition, 2003.

The COMPSET Algorithm for Subset Selection

Yaniv Hamo and Shaul Markovitch

{hamo,shaulm}@cs.technion.ac.il

Computer Science Department, Technion, Haifa 32000, Israel

Abstract

Subset selection problems are relevant in many domains. Unfortunately, their combinatorial nature prohibits solving them optimally in most cases. Local search algorithms have been applied to subset selection with varying degrees of success. This work presents COMPSET, a general algorithm for subset selection that invokes an existing local search algorithm from a random subset and its complementary set, exchanging information between the two runs to help identify wrong moves. Preliminary results on complex SAT, Max Clique, 0/1 Multidimensional Knapsack and Vertex Cover problems show that COMPSET improves the efficient stochastic hill climbing and tabu search algorithms by up to two orders of magnitudes.

1 Introduction

The subset selection problem (SSP) is simply defined: Given a set of elements $E = \{e_1, e_2, \dots, e_n\}$ and a utility function $U : 2^E \mapsto R$, find a subset $S \subseteq E$ such that $U(S)$ is optimal. Many real-life problems are SSPs, or can be formulated as such. Classic examples include SAT, max clique, independent set, vertex cover, knapsack, set covering, feature subset selection (classification) and instance selection (for nearest-neighbor classifiers) to name a few.

Since the search space is exponential in the size of E , finding an optimal subset without relaxing assumptions is intractable. Problems associated with subset selection are typically NP-hard or NP-complete. Local search algorithms are among the common methods for solving hard combinatorial optimization problems such as subset selection. Hill climbing, simulated annealing [Kirkpatrick *et al.*, 1983] and tabu search [Glover and Laguna, 1993] have all been proven to provide good solutions in a variety of domains. The general technique of random restarts is applicable to all of them, yielding anytime behavior as well as the PAC property (probability to find the optimal solution converges to 1 as the running time approaches infinity).

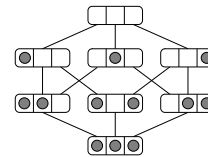
The problem with these local search algorithms is, however, that they are *too* general. The only heuristic they have about the domain is in a form of a black-box, the utility

function which they try to optimize. It is therefore common to have modifications to these algorithms that trade being general for additional domain-specific knowledge. The SAT domain is full of such variants [Gent and Walsh, 1993; Hoos and Stützle, 2005] but they are common in other domains as well [Khuri and Bäck, 1994; Evans, 1998].

In this paper we present a general modification to known local search algorithms which improve their performance on SSPs. The idea behind it is to exploit attributes that are specific to search spaces of subset selection. Knowing that it is a subset search space allows us to infer which moves were likely to be wrong. By reversing these moves and trying again, we start from a new context, and the probability to repeat the mistake is reduced. In experiments performed on complex SAT, max clique, 0/1 multidimensional knapsack and vertex cover problems, the new method has shown to significantly improve the underlying search algorithm.

2 The COMPSET Algorithm

Subset selection can be expressed as a search in a graph. Each node (state) in the graph represents a unique subset. Edges correspond to adding or removing an element from the subset, thus there are $n = |E|$ edges to every node. The following figure shows the search graph for 3 elements.



A state S can be represented by a bit vector where bit $S_{[i]}$ is 1 iff $e_i \in S$. Moving to a neighboring state in the graph is equivalent to flipping one bit.

Each state is associated with a utility value, which is the value of U on the subset it represents. Local search algorithms typically start from a random state and make successive improvements to it by moving to neighboring states. They vary from each other mainly in their definition of neighborhood, and their selection method.

Using this representation, all local search algorithms are also applicable to subset selection. However, being general, they overlook the specific characteristics of subset selection. COMPSET guides a given local search algorithm using knowledge specific to subset selection.

2.1 Characteristics of Subset Selection

In a selection problem of n elements, there are n operators: $F = \{f_1, f_2, \dots, f_n\}$ where f_i is the operator of toggling the membership of element i in a set (if it was in the set remove it, or add it if it was out). Applying f_i is equivalent to flipping the i th bit in the bit vector representation. Throughout the following discussion we assume a single optimal subset (solution) which we denote S^* .

We make the following observations about subset search:

Observation 1. Let S' be an arbitrary state (subset). From any state S , there exists a subset of the operators, $\sigma(S, S') \subseteq F$, that when applied to S results in S' .

Proof. Since S' , as S , is an n bits long vector, there are at most n bits of S' that do not agree with S and need to be flipped using an operator. $\square$

Observation 2. Let $\bar{S}$ be the complementary state of S i.e., the state derived from S by flipping all its bits. The subset of operators $\sigma(S, S^*)$ leading from S to the solution S^* , is the complementary subset of $\sigma(\bar{S}, S^*)$ in F . That is, $\sigma(S, S^*) \cup \sigma(\bar{S}, S^*) = F$ and $\sigma(S, S^*) \cap \sigma(\bar{S}, S^*) = \emptyset$.

Proof. We need to show, that for every $f_i \in F$ either $f_i \in \sigma(S, S^*)$ or $f_i \in \sigma(\bar{S}, S^*)$. If $S_{[i]} = S^*_{[i]}$ then $f_i \notin \sigma(S, S^*)$ (it does not need to be flipped). Moreover, if $S_{[i]} = S^*_{[i]}$, then necessarily $\bar{S}_{[i]} \neq S^*_{[i]}$, since in $\bar{S}$ all bits are flipped. Thus, $f_i \notin \sigma(S, S^*) \rightarrow f_i \in \sigma(\bar{S}, S^*)$. The same goes for the other possible case, in which $\bar{S}_{[i]} = S^*_{[i]}$. $\square$

The inherent problem in finding $\sigma(S, S^*)$ using local search, is that operators are applied successively and their effect is not necessarily of monotonic improvement due to interdependencies between elements. Such non-monotonic behavior of U confuses local search algorithms and often makes them trapped in local optima. In such a case, there are two options: either the search is progressing on the correct path to the solution (but the algorithm does not see a way of continuing), or it is off the correct path altogether. It would be beneficial to distinguish between these two scenarios.

Consider two independent hill climbing runs, one from S and one from $\bar{S}$. Given that the optimal solution is not found, the two runs have stopped in local optima, L_S and $L_{\bar{S}}$ respectively. We consider the subsets of operators leading from S to L_S and from $\bar{S}$ to $L_{\bar{S}}$. By observation 2, it is not possible that an operator f_i appears in both operator subsets if they are both on the path to S^* . If we do observe the same operator in both, it is a clear sign that one of them is wrong. This is the idea behind COMPSET, which is described next.

2.2 Description of the COMPSET Algorithm

Interdependencies between elements are distracting when searching for good solutions. Had all elements been independent, a simple linear search, which adds element after element as long as the utility value improves, would suffice. Local optima are an example where such interdependency brings the search to a full stop. It is likely that by applying the operators in a different order, or eliminating some, the local optimum would have been avoided. COMPSET uses

the above observations to try and identify wrong invocations of operators. It then cancels them (reverses their effect) and resumes the run towards another local optimum or, hopefully, the solution.

Procedure COMPSET(S , LOCALSEARCHALG)

```

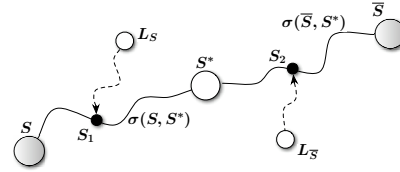
 $S_1 \leftarrow S$ ;  $S_2 \leftarrow \bar{S}$ ; agree  $\leftarrow$  false
loop until agree
   $L_S \leftarrow \text{LOCALSEARCHALG}(S_1)$ 
   $L_{\bar{S}} \leftarrow \text{LOCALSEARCHALG}(S_2)$ 
   $C \leftarrow \sigma(S, L_S) \cap \sigma(\bar{S}, L_{\bar{S}})$ 
  if  $C$  is empty then
    agree  $\leftarrow$  true
  else
     $S_1 \leftarrow C(L_S)$  // apply all operators in  $C$  on  $L_S$ 
     $S_2 \leftarrow C(L_{\bar{S}})$ 
    return the better between  $L_S$  and  $L_{\bar{S}}$ 
end

```

end

Given a start state S , COMPSET initiates two runs of the given local search algorithm, one from S and one from its complementary $\bar{S}$. Once two local optima are achieved, the series of operators that has led to each one is examined. Every operator that appears in both series must be wrong in one of them. We do not know which of the runs went wrong, so we reverse the effect of such operators in both local optima. Once all obviously wrong operators are undone, the local search is continued. The process repeats upon encountering the next pair of local optima. When no conflicting operators exist, and the optimal solution was not found, COMPSET ends and returns the better of the two local optima in hand.

The rationale behind COMPSET is illustrated here:



The solution S^* is reachable by applying all operators from $\sigma(S, S^*)$ to S (in any order), and by applying all operators from $\sigma(\bar{S}, S^*)$ to $\bar{S}$. The underlying search algorithm from S is in a correct path if it is using only operators from $\sigma(S, S^*)$.

However, a search can easily divert from this path, and it is often necessary for its overall convergence to the solution. Diversion occurs if an operator from $\sigma(\bar{S}, S^*)$ is applied to S , or an operator from $\sigma(S, S^*)$ is applied to $\bar{S}$. The problem is, that since we do not know $\sigma(S, S^*)$ or $\sigma(\bar{S}, S^*)$, it is difficult to detect such diversions. However, what we do know, is that the solution S^* conforms to $\sigma(S, S^*) \cap \sigma(\bar{S}, S^*) = \emptyset$. We can use this fact to try and identify diversions.

Once stopped in local optima L_S from S and $L_{\bar{S}}$ from $\bar{S}$, we check $\sigma(S, L_S) \cap \sigma(\bar{S}, L_{\bar{S}})$. If the intersection is not empty then by definition either L_S or $L_{\bar{S}}$ are off the path. Note, that if the intersection is empty, the local optima might still be off path, because either $\sigma(S, L_S)$ uses an operator from $\sigma(\bar{S}, S^*)$, or $\sigma(\bar{S}, L_{\bar{S}})$ uses an operator from $\sigma(S, S^*)$. However, $\sigma(S, L_S) \cap \sigma(\bar{S}, L_{\bar{S}}) \neq \emptyset$ means that for sure at

least one of the local optima is off path.

In general it is possible that the search algorithm would continue applying operators and finally return to the path, but being in a local optimum means that it has essentially "given up". Elimination of all conflicting operators from both sides brings us to S_1 and S_2 , $L_S \rightarrow S_1$ and $L_{\bar{S}} \rightarrow S_2$. Since all common operators were eliminated, $\sigma(S, S_1) \cap \sigma(\bar{S}, S_2) = \emptyset$ and thus S_1 and S_2 are on a possibly correct path. It is still possible that $\sigma(S, S_1)$ contains operators from $\sigma(\bar{S}, S^*)$ or that $\sigma(S, S_2)$ contains operators from $\sigma(S, S^*)$ thus still being an obstacle for reaching the solution.

An important point to notice, is that S_1 and S_2 are not necessarily states that the algorithm has visited before. Groups of operators are simultaneously eliminated, an operation which interdependencies would prohibit had the operators been successively eliminated. COMPSET effectively switches to another context which is mostly correct, in which the eliminated operators can be tried again.

3 Empirical Evaluation

The following algorithms were considered:

- *Stochastic Hill Climbing* (SHC) - starts from a random subset, iteratively picks a neighboring subset (differs in exactly one element) in random and moves there if it has a better or equal utility value. The simplicity of SHC often misleads; several works [Mitchell *et al.*, 1994; Baluja, 1995] showed that SHC does not fall from the complex GA mechanism. In the SAT domain such stochastic local search (SLS) methods have been shown to be comparable with state-of-the-art domain-specific algorithms [Hoos and Stützle, 2005].
- *Tabu Search* (TS) [Glover and Laguna, 1993] - examines the neighborhood of the current state for the best replacement. It moves to the chosen state even if it does not improve the current state, which might result in cycles. To avoid cycles, TS introduces the notion of a *tabu list* that stores the last t (*tabu tenure*) operators used. TS is prevented from using operators from the tabu list when it generates the neighborhood to be examined, unless certain conditions called *aspiration criteria* are met. In this paper we use a common aspiration criterion that allows operators which lead to better state than the best obtained so far.
- *Simulated Annealing* (SA) [Kirkpatrick *et al.*, 1983] - begins at high *temperature* which enables it to move to arbitrary neighboring states, including those which are worse than the current one. As the temperature declines, the search is less likely to choose a non-improving state, until it settles in a state which is a local minimum from which it cannot escape in low temperatures.

To test the effectiveness of COMPSET we have applied it to SHC and TS. COMPSET is not applicable to SA since SA begins with a high temperature at which it randomly moves far from the initial state. The concept of COMPSET is to set new start points for the underlying algorithm and by randomly moving away from them SA defeats its purpose.

The algorithms were tested in the following domains:

- *Propositional Satisfiability* (SAT) - the problem of finding a truth assignment that satisfies a given boolean formula represented by a conjunction of clauses (CNF) $C_1 \wedge \dots \wedge C_m$.

SAT is a classic SSP since we look for a subset of variables that when assigned a true value, makes the entire formula true. The utility function is the number of unsatisfied clauses when assigning true to all variables in S :

$$U(S) \equiv |\{C_i | C_i \text{ is false under } S, 0 \leq i \leq m\}|$$

The global minimum for U is 0, for satisfied formulas. A search algorithm using this utility function will attempt to maximize the number of satisfied clauses, which is a generalization of SAT called MAX-SAT. Problem instances for SAT were obtained from the SATLIB [Hoos and Stützle, 2000] repository of random 3-SAT. We use problems from the solubility phase transition region¹ [Cheeseman *et al.*, 1991].

- *Max Clique* - another classic SSP, where the goal is to find the maximum subset of vertices that forms a clique in a graph. Given a graph $G = (V, E)$ and a subset $S \subseteq V$, we define the following utility function:

$$U(S) \equiv \begin{cases} |V| - |S| & \text{S is a clique} \\ |V| - |S| + |V| + |S| \cdot (|S| - 1) - |E_S| & \text{else} \end{cases}$$

A clique should be maximized but our implementation always minimizes U , therefore we use $|V| - |S|$. Incomplete solutions are penalized by the number of additional edges they require for being a clique ($|S| \cdot (|S| - 1) - |E_S|$), plus a fixed value $|V|$ that is used to separate them from the legal solutions. By striving to minimize U , the search algorithm finds feasible solutions first, and then continues by minimizing their size. The global minimum of U corresponds to the maximum clique. Problem instances were obtained from the DIMACS [1993] benchmark for maximum clique.

- *0/1 Multidimensional Knapsack* (MKP) - the problem of filling m knapsacks with n objects. Each object is either placed in all m knapsacks, or in none of them (hence "0/1"). The knapsacks have capacities of $c_1, c_2, \dots, c_m$. Each object is associated with a profit p_i and m different weights, one for each knapsack. Object i weighs w_{ij} when put into knapsack j . The goal is to find a subset of objects yielding the maximum profit without overfilling any of the knapsacks. Knapsack j is overfilled in state S iff $\sum_{i=1}^n S[i] \cdot w_{ij} > c_j$. Let k be the number of overfilled knapsacks. We define:

$$U(S) \equiv \begin{cases} -\sum_{i=1}^n S[i] \cdot p_i & k=0 \\ k & k > 0 \end{cases}$$

The utility of feasible subsets is simply their profit (with minus sign for minimization purposes). Infeasible solutions are penalized for each knapsack they overfill. Problem instances for MKP were obtained from the OR-library [Beasley, 1997].

- *Vertex Cover* - the goal is to find the smallest subset of vertices in a graph that covers all edges. Given a graph $G = (V, E)$, we define:

$$U(S) \equiv \begin{cases} |S| & \text{S covers all edges} \\ |S| + |V| + |E \setminus E_S| & \text{else} \end{cases}$$

¹Random 3-SAT problem with 4.26 clauses per variable that are the hardest to solve using local search.

For legal vertex covers, U takes values less than or equal to $|V|$. Incomplete solutions are penalized by the number of edges they do not cover, plus a fixed value $|V|$ that is used to separate them from the legal solutions. The global minimum of U corresponds to the optimal vertex cover.

The complementary graphs of the instances from the original DIMACS benchmark were taken, so that the known maximum clique sizes could be translated to corresponding minimum vertex covers².

3.1 Experimental Methodology

We have tested five algorithms: SHC, TS, SA (with $T = 100$, $\alpha = 0.95$), COMPSET over SHC and COMPSET over TS. TS was used with $t = 5$ for all domains other than SAT, and $t = 9$ for SAT. Each run was limited to 10^7 U evaluations.

All algorithms use random restart to escape from local optima when they have still not exhausted their evaluations quota. They use random restart also when there is no improvement over the last k steps. We use $k = 10$ for domains other than SAT, and $k = 20$ for SAT. SAT is characterized by wide and frequent plateaus [Frank *et al.*, 1997] therefore we chose higher values of t and k for it.

100 runs of each algorithm were performed on each problem in the test sets. Each run started from a random state, that was common to all algorithms. We measured the number of U evaluations needed to obtain the optimal solution in each run, as well as the time taken.

3.2 Results

The results are summarized in Tables 1, 2, 3 and 4. For brevity, we did not include the timing information in these tables. The considered algorithms do not introduce a significant overhead, so the execution time is a linear function of the number of U evaluations. The tables show the characteristics of the problem instance, followed by the number of successful runs (columns titled *#ok*) and the average number of U evaluations for each algorithm. A successful run is a run in which the algorithm has found the optimal solution within the limit. We tested the statistical significance of the improvement introduced by COMPSET using the Wilcoxon matched-pairs signed-ranks test with the extension by Etzioni and Etzioni [1994] to cope with censored data³. A "+" sign in the *sg.* column between SHC and COMPSET/SHC indicates that COMPSET improved SHC with $p < 0.05$. A "-" sign indicates that SHC performed better with $p < 0.05$. A "?" sign indicates that the difference is not significant. Whenever it is not possible to draw definitive conclusions since there is too much censored data, *n/a* appears. The same holds for the *sg.* column between TS and COMPSET/TS.

The superiority of COMPSET over the other algorithms is striking, both in the number of evaluations, and the number and difficulty of instances solved. In the SAT domain, the best performing algorithms are COMPSET/TS and

COMPSET/SHC. The average success ratio of COMPSET/TS is 85% and of COMPSET/SHC is 76%. For comparison, the success ratios for SA, TS and SHC are 37%, 23% and 28% respectively. The speedup factor gained by using COMPSET is as large as 462 for SHC (instance *uf50-011*) and as large as 156 for TS (instance *uf75-013*). Note that these are lower bounds since SHC and TS were terminated because of resource limit for some of the runs.

The best performing algorithms in the Max Clique domain are COMPSET/SHC and SHC with average success ratios of 94% and 80% respectively. The speed up factor of COMPSET/SHC over SHC is as large as 14 (instance *sanr200-0.7*).

In the Knapsack domain, the best performing algorithm is COMPSET/SHC with an average success ratio of 89%. For comparison, the success ratios for TS, SA, COMPSET/SHC and SHC are 50%, 25%, 19% and 17% respectively. The speedup factor gained by using COMPSET is as large as 127 for TS (instance *WEISH07*) and as large as 3.8 for SHC (instance *WEISH04*).

The best performing algorithms in the Vertex Cover domain are COMPSET/SHC and SHC with average success ratios of 94% and 77% respectively. SA is relatively close with 71% but TS is far behind with 23%, improved by COMPSET to 28%. The speedup factor gained by using COMPSET is as large as 310 for TS (instance *hamming6-2*) and as large as 10 for SHC (instance *sanr200-0.7*).

Another interesting statistics is the number of random restarts required by the underlying search algorithm and COMPSET, as well as the number of operator eliminations performed by COMPSET and how many operators they spanned. We have collected this data throughout 100 runs on the *p_hat1000-1* vertex cover problem, a graph of 1000 vertices. SHC required 517.37 random restarts on average (in each run), while COMPSET required only 3.13. COMPSET has performed 20.52 operator eliminations, reversing the effect of 2.97 operators each time.

4 Conclusions and Future Work

In this paper, we have provided useful insights into the domain of subset selection. We have realized that using local search, paths from complementary subsets to the solution must be distinct in terms of the operators used. This has led us to conclude that if the paths contain common operators, it may serve as an indication of a mistake. To test our conjecture, we introduced COMPSET, a new guiding policy for local search algorithms in the context of SSP. The results show a significant improvement over both TS and SHC by up to two orders of magnitudes.

We currently in the process of running COMPSET on other subset selection domains, progressing towards a better understanding of its behavior. One interesting direction is to research for ways to incorporate knowledge of the entire search paths, instead of only the local minima at their end. In addition, it is beneficial to find out how characteristics of a specific problem affect its performance. Overall, the general idea of incorporating such SSP specific insights seems to be a promising lead to better subset selection algorithms.

²Note that while it is possible to take the complementary graph, solve the Max Clique problem, and then translate back to Vertex Cover, none of the algorithms in this paper has done so.

³The information about runs in which the solution was not found within the given bound is called *truncated* or *censored*.

3-SAT Instances			SHC		sg.	COMPSET/SHC		TS		sg.	COMPSET/TS		SA	
name	vars	clauses	#ok	evals		#ok	evals	#ok	evals		#ok	evals	#ok	evals
uf20-011	20	91	100	211	+	100	167	100	309	?	100	275	82	1,800,168
uf20-012	20	91	100	115	?	100	102	100	190	?	100	201	100	197
uf20-013	20	91	100	576	?	100	655	100	1,089	?	100	940	57	4,300,196
uf20-014	20	91	100	590	?	100	596	100	872	?	100	863	72	2,800,287
uf20-015	20	91	100	243	?	100	256	100	421	?	100	361	98	201,003
uf50-011	50	218	74	5,920,013	+	100	12,792	100	160,575	+	100	17,250	11	8,900,193
uf50-012	50	218	100	1,162,273	+	100	10,624	100	36,955	?	100	35,072	22	7,800,249
uf50-013	50	218	100	1,682,981	+	100	3,803	100	45,616	+	100	9,275	65	3,501,795
uf50-014	50	218	100	1,995,884	+	100	6,704	100	66,185	+	100	14,317	40	6,001,598
uf50-015	50	218	99	1,005,804	+	100	2,547	100	33,894	+	100	7,335	68	3,201,009
uf75-011	75	325	0	-	+	100	80,928	16	9,137,745	+	100	142,969	37	6,306,563
uf75-012	75	325	0	-	+	100	409,036	12	9,375,096	+	100	579,683	5	9,500,223
uf75-013	75	325	0	-	+	100	9,700	63	5,890,614	+	100	37,587	56	4,404,356
uf75-014	75	325	0	-	+	100	22,677	50	7,364,576	+	100	77,866	41	5,904,975
uf75-015	75	325	0	-	+	100	89,098	27	8,595,672	+	100	179,802	53	4,762,145
uf100-011	100	430	0	-	+	100	289,359	0	-	+	100	450,070	16	8,404,825
uf100-012	100	430	0	-	+	100	130,835	0	-	+	100	309,667	37	6,321,626
uf100-013	100	430	0	-	+	100	158,180	0	-	+	100	376,166	40	6,011,208
uf100-014	100	430	0	-	+	100	872,275	0	-	+	100	966,083	41	6,175,183
uf100-015	100	430	0	-	+	100	283,024	0	-	+	100	361,632	19	8,110,739
uf125-011	125	538	0	-	+	100	544,188	0	-	+	100	496,277	28	7,211,118
uf125-012	125	538	0	-	+	100	786,882	0	-	+	100	1,048,369	21	7,908,011
uf125-013	125	538	0	-	n/a	52	7,202,045	0	-	+	87	4,471,437	11	8,906,967
uf125-014	125	538	0	-	n/a	46	7,653,790	0	-	n/a	67	6,388,535	4	9,601,212
uf125-015	125	538	0	-	+	100	1,574,893	0	-	+	100	1,597,965	10	9,001,632
uf150-011	150	645	0	-	+	100	1,129,916	0	-	+	100	650,899	66	3,458,095
uf150-012	150	645	0	-	n/a	7	9,525,999	0	-	n/a	48	6,722,510	17	8,332,034
uf150-013	150	645	0	-	+	82	4,429,762	0	-	+	100	1,037,124	64	3,750,319
uf150-014	150	645	0	-	n/a	49	6,748,236	0	-	+	97	3,384,813	15	8,507,697
uf150-015	150	645	0	-	n/a	11	9,373,759	0	-	n/a	48	7,193,000	15	8,517,484
uf200-011	200	860	0	-	n/a	2	9,811,767	0	-	n/a	7	9,703,756	10	9,013,804
uf200-012	200	860	0	-	n/a	0	-	0	-	n/a	24	8,434,746	14	8,642,899
uf200-013	200	860	0	-	n/a	7	9,506,131	0	-	n/a	39	7,769,246	15	8,546,974
uf200-014	200	860	0	-	n/a	2	9,857,446	0	-	n/a	41	6,907,585	40	6,121,901
uf200-015	200	860	0	-	n/a	1	9,900,146	0	-	n/a	3	9,766,815	4	9,648,972

Table 1: SAT: average over all 100 runs, including censored data

Graphs			SHC		sg.	COMPSET/SHC		TS		sg.	COMPSET/TS		SA	
name	V	opt	#ok	evals		#ok	evals	#ok	evals		#ok	evals	#ok	evals
brock200-1	200	21	41	7,953,200	+	98	2,485,541	0	-	n/a	0	-	24	8,880,969
hamming6-2	64	32	100	659	?	100	633	80	2,984,249	+	100	14,163	100	1,129
hamming6-4	64	4	100	302	?	100	275	100	1,794	?	100	1,807	100	517
hamming8-2	256	128	100	4,317	?	100	4,519	6	9,402,266	n/a	14	8,612,975	100	6,486
hamming8-4	256	16	100	8,924	+	100	6,609	45	5,514,223	n/a	54	4,619,182	100	18,546
hamming10-2	1024	512	100	36,837	?	100	38,312	6	9,438,856	n/a	6	9,438,856	100	54,520
hamming10-4	1024	40	62	6,716,761	n/a	70	6,336,176	0	-	n/a	0	-	36	7,956,686
johnson8-2-4	28	4	100	81	?	100	90	100	299	?	100	305	100	204
johnson8-4-4	70	14	100	1,208	+	100	898	45	5,575,650	+	97	1,005,066	100	2,802
johnson16-2-4	120	8	100	740	?	100	631	100	6,470	?	100	6,305	100	745
johnson32-2-4	496	16	100	4,661	?	100	4,661	83	1,798,068	+	85	1,600,804	100	4,843
p.hat700-1	700	11	38	8,029,518	+	92	3,471,758	1	9,902,542	n/a	1	9,902,542	6	9,658,510
p.hat700-2	700	44	100	361,250	+	100	92,712	1	9,902,564	n/a	1	9,902,564	100	639,270
p.hat700-3	700	62	100	912,534	+	100	224,936	0	-	n/a	0	-	100	1,513,576
p.hat1000-1	1000	10	92	3,764,601	+	100	891,693	4	9,619,971	n/a	4	9,619,971	69	6,161,992
p.hat1000-2	1000	46	100	1,177,655	+	100	211,710	0	-	n/a	0	-	100	1,645,523
p.hat1000-3	1000	68	43	7,490,258	+	100	1,253,011	0	-	n/a	0	-	31	8,610,929
p.hat1500-1	1500	12	0	-	n/a	3	9,926,760	0	-	n/a	0	-	0	-
p.hat1500-2	1500	65	99	2,215,890	+	100	369,480	0	-	n/a	0	-	98	2,629,112
p.hat1500-3	1500	94	86	4,982,543	+	100	1,943,943	0	-	n/a	0	-	73	5,884,335
sanr200-0.7	200	18	100	2,189,803	+	100	151,344	0	-	n/a	1	9,900,505	91	3,530,091
sanr200-0.9	200	42	96	2,871,830	+	100	530,846	0	-	n/a	0	-	73	5,535,066
sanr400-0.5	400	13	26	8,561,953	+	86	4,167,521	0	-	n/a	0	-	7	9,683,128
sanr400-0.7	400	21	37	8,029,818	+	95	3,493,629	0	-	n/a	0	-	11	9,252,956

Table 2: Maximum Clique: average over all 100 runs, including censored data

References

- [Baluja, 1995] S. Baluja. An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, School of Computer Science, CMU, 1995.
- [Beasley, 1997] J. E. Beasley. OR-library: a collection of test data sets. Technical report, Management School, Imperial College, London, 1997.
- [Cheeseman *et al.*, 1991] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, pages 331–336, 1991.
- [DIMACS, 1993] DIMACS. Challenge problems for maximum clique, dimacs.rutgers.edu., 1993.
- [Etzioni and Etzioni, 1994] O. Etzioni and R. Etzioni. Statistical methods for analyzing speedup learning experiments. *Machine Learning*, 14(1):333–347, 1994.
- [Evans, 1998] Isaac K. Evans. Evolutionary algorithms for vertex cover. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, pages 377–386, 1998.
- [Frank *et al.*, 1997] J. Frank, P. Cheeseman, and J. Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

Problems				SHC			COMPSET/SHC			TS			COMPSET/TS			SA		
name	n	m	opt	#ok	evals	sg.	#ok	evals	#ok	evals	sg.	#ok	evals	#ok	evals	#ok	evals	
HP1	28	4	3418	80	4,911,555	n/a	56	6,544,038	100	719,239	-	98	1,968,905	96	3,404,960			
HP2	35	4	3186	13	9,252,758	n/a	7	9,594,786	53	6,704,029	n/a	65	6,402,176	17	9,235,896			
PB1	27	4	3090	96	3,072,586	-	87	3,952,437	100	670,184	-	100	1,200,320	100	1,982,159			
PB2	34	4	3186	16	9,158,395	n/a	8	9,670,627	61	5,895,593	n/a	76	5,213,605	31	8,258,914			
PB4	29	2	95168	12	9,400,912	n/a	14	9,278,001	100	141,267	-	100	186,156	2	9,958,448			
PB5	20	10	2139	100	352,913	?	100	299,587	100	80,253	-	100	735,297	100	318,187			
PB6	40	30	776	74	5,420,914	n/a	74	5,318,641	30	7,715,915	+	100	17,418	25	8,949,831			
PB7	37	30	1035	0	-	n/a	2	9,866,520	44	7,526,288	+	100	168,889	40	7,654,095			
SENT01	60	30	7772	0	-	n/a	0	-	1	9,900,037	+	100	167,722	0	-			
SENT02	60	30	8722	0	-	n/a	0	-	0	-	+	93	3,742,251	0	-			
WEING1	28	2	141278	1	9,951,662	n/a	1	9,967,393	100	430,212	?	100	295,032	1	9,913,058			
WEING2	28	2	130883	1	9,988,929	n/a	1	9,920,886	86	4,325,976	?	90	4,249,753	0	-			
WEING3	28	2	95677	7	9,669,592	n/a	3	9,851,757	37	8,002,117	n/a	38	7,917,398	1	9,922,652			
WEING4	28	2	119337	45	7,854,172	n/a	39	8,067,189	100	352,464	+	100	122,820	19	8,968,454			
WEING5	28	2	98796	9	9,355,108	n/a	7	9,720,010	79	4,962,961	n/a	80	4,560,088	3	9,894,039			
WEING6	28	2	130623	2	9,896,760	n/a	1	9,961,823	100	2,190,253	?	99	2,263,775	1	9,950,248			
WEISH01	30	5	4554	0	-	n/a	8	9,627,601	100	142,532	+	100	6,808	56	6,542,273			
WEISH02	30	5	4536	15	9,256,801	n/a	19	9,029,940	100	203,812	+	100	4,981	96	3,118,697			
WEISH03	30	5	4115	13	9,367,427	n/a	44	7,691,588	100	133,341	+	100	15,052	92	4,043,836			
WEISH04	30	5	4561	50	7,207,287	+	100	1,860,648	100	27,689	+	100	1,944	100	1,645,413			
WEISH05	30	5	4514	93	3,749,094	+	100	1,856,242	100	45,793	+	100	3,300	96	2,753,007			
WEISH06	40	5	5557	0	-	n/a	0	-	34	7,450,036	+	100	237,437	5	9,752,363			
WEISH07	40	5	5567	0	-	n/a	2	9,904,913	41	7,047,862	+	100	55,207	16	9,189,652			
WEISH08	40	5	5605	0	-	n/a	1	9,993,609	15	9,186,377	+	99	1,988,697	5	9,797,464			
WEISH09	40	5	5246	0	-	n/a	1	9,965,165	51	6,243,931	+	100	62,502	1	9,981,328			
WEISH10	50	5	6339	0	-	n/a	0	-	12	9,041,645	+	100	185,651	0	-			
WEISH11	50	5	5643	0	-	n/a	0	-	7	9,393,532	n/a	79	4,126,644	0	-			
WEISH12	50	5	6339	0	-	n/a	0	-	12	8,933,081	+	100	287,267	0	-			
WEISH13	50	5	6159	0	-	n/a	0	-	11	9,123,859	+	99	1,041,062	0	-			
WEISH14	60	5	6954	0	-	n/a	0	-	1	9,901,672	+	94	2,668,066	0	-			
WEISH15	60	5	7486	0	-	n/a	0	-	11	9,008,543	+	100	729,395	0	-			
WEISH16	60	5	7289	0	-	n/a	0	-	0	-	n/a	49	6,847,342	0	-			
WEISH17	60	5	8633	0	-	n/a	0	-	1	9,906,889	+	100	1,344,074	2	9,844,304			
WEISH18	70	5	9580	0	-	n/a	0	-	2	9,843,393	n/a	49	7,382,229	0	-			
WEISH19	70	5	7698	0	-	n/a	0	-	0	-	n/a	4	9,705,819	0	-			
WEISH20	70	5	9450	0	-	n/a	0	-	1	9,935,892	+	83	4,260,796	0	-			

Table 3: Knapsack: average over all 100 runs, including censored data

Graphs			SHC		sg.	COMPSET/SHC		TS		sg.	COMPSET/TS		SA	
name	V	opt	#ok	evals		#ok	evals	#ok	evals		#ok	evals	#ok	evals
brock200-1	200	179	44	7,703,031	+	97	2,407,132	0	-	n/a	0	-	24	8,706,894
hamming6-2	64	32	100	705	?	100	663	74	3,241,435	+	100	10,444	100	1,258
hamming6-4	64	60	100	325	?	100	299	100	1,813	?	100	1,790	100	535
hamming8-2	256	128	100	4,442	?	100	4,642	11	8,903,987	n/a	27	7,320,998	100	6,176
hamming8-4	256	240	100	9,399	+	100	6,287	39	6,112,570	n/a	49	5,117,786	100	18,168
hamming10-2	1024	512	100	44,727	+	100	38,674	5	9,535,915	n/a	7	9,353,124	100	59,437
hamming10-4	1024	984	58	6,412,744	n/a	71	5,645,469	0	-	n/a	0	-	37	7,811,420
johnson8-2-4	28	24	100	75	?	100	75	100	292	?	100	292	100	201
johnson8-4-4	70	56	100	1,122	?	100	962	52	4,911,854	+	98	632,835	100	2,889
johnson16-2-4	120	112	100	618	?	100	618	99	106,329	?	100	6,423	100	742
johnson32-2-4	496	480	100	4,564	?	100	4,564	79	2,192,741	n/a	79	2,192,741	100	4,339
p.hat700-1	700	689	24	8,778,686	+	96	3,214,600	1	9,902,454	n/a	1	9,902,454	10	9,588,452
p.hat700-2	700	656	100	461,639	+	100	102,745	0	-	n/a	0	-	100	630,340
p.hat700-3	700	638	100	1,189,058	+	100	210,935	0	-	n/a	0	-	100	1,507,522
p.hat1000-1	1000	990	86	3,917,288	+	100	829,765	1	9,905,190	n/a	1	9,905,190	60	6,770,941
p.hat1000-2	1000	954	100	1,562,316	+	100	195,142	1	9,904,961	n/a	1	9,904,961	100	2,048,317
p.hat1000-3	1000	932	32	8,093,992	+	100	1,163,768	0	-	n/a	0	-	22	8,633,648
p.hat1500-1	1500	1488	0	-	n/a	2	9,958,079	0	-	n/a	0	-	0	-
p.hat1500-2	1500	1435	99	2,210,577	+	100	408,447	0	-	n/a	0	-	96	2,669,373
p.hat1500-3	1500	1406	75	5,533,569	+	100	2,070,512	0	-	n/a	0	-	74	5,381,934
sanr200-0.7	200	182	99	1,922,341	+	100	181,238	0	-	n/a	0	-	85	4,625,237
sanr200-0.9	200	158	98	3,074,483	+	100	551,969	0	-	n/a	1	9,900,390	78	5,005,884
sanr400-0.5	400	387	15	9,083,301	+	87	4,860,526	0	-	n/a	0	-	8	9,662,423
sanr400-0.7	400	379	27	8,500,584	+	96	2,838,407	0	-	n/a	0	-	9	9,543,268

Table 4: Vertex cover: average over all 100 runs, including censored data

[Gent and Walsh, 1993] Ian P. Gent and Toby Walsh. Towards an understanding of hill-climbing procedures for SAT. In *National Conference on AI*, pages 28–33, 1993.

[Glover and Laguna, 1993] F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, 1993.

[Hoos and Stützle, 2000] Holger H. Hoos and Thomas Stützle. SATLIB: An Online Resource for Research on SAT. In *SAT2000: Highlights of Satisfiability Research in the year 2000*, pages 283–292. 2000.

[Hoos and Stützle, 2005] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search - Foundations and Ap-*

plications. Morgan Kaufman Publishers, 2005.

[Khuri and Bäck, 1994] S. Khuri and T. Bäck. An evolutionary heuristic for the minimum vertex cover problem. In *Genetic Algorithms within the Framework of Evolutionary Computation*, pages 86–90, 1994.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598:671–680, 1983.

[Mitchell et al., 1994] M. Mitchell, J. H. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing. In *Advances in NIPS*, volume 6, pages 51–58, 1994.

Combining Memory and Landmarks with Predictive State Representations

Michael R. James and Britton Wolfe and Satinder Singh

Computer Science and Engineering
University of Michigan
{mrjames, bdwolfe, baveja}@umich.edu

Abstract

It has recently been proposed that it is advantageous to have models of dynamical systems be based solely on observable quantities. Predictive state representations (PSRs) are a type of model that uses predictions about future observations to capture the state of a dynamical system. However, PSRs do not use memory of past observations. We propose a model called memory-PSRs that uses both memories of the past, and predictions of the future. We show that the use of memories provides a number of potential advantages. It can reduce the size of the model (in comparison to a PSR model). In addition many dynamical systems have memories that can serve as landmarks that completely determine the current state. The detection and recognition of landmarks is advantageous because they can serve to reset a model that has gotten off-track, as often happens when the model is learned from samples. This paper develops both memory-PSRs and the use and detection of landmarks.

1 Introduction

This paper explores the use of two types of observable quantities — the history of past observations and predictions about future observations — in creating models for complex dynamical systems. Models that use only predictions of future observations to capture state, called predictive state representations or PSRs [Littman *et al.*, 2001], have been shown (surprisingly) to be at least as expressive and compact [Singh *et al.*, 2004] as classical models such as partially observable Markov decision processes (POMDPs) that use hidden state variables. Models that only use observation history, such as k -order Markov models, are known to be not as expressive and hence not as widely applicable as PSRs or POMDPs. On the other hand, history-based models are easier to learn from data than either PSRs and POMDPs. In this paper, we propose an extension to PSRs, called memory-PSRs (*mPSRs*), that combines past observations (*memories*) with predictions about the future to define the state of a dynamical system.¹

¹Sutton and Tanner [2004] have also recently proposed a model that combines memory and predictions.

Our goal is to provide a model class that is as expressive as PSRs but that has some of the efficient learning properties of history-based models.

In addition to accelerating learning, *mPSRs* can also exploit *landmarks* or observations that capture the state of the system uniquely, by “resetting” the approximate state of the learned model during prediction whenever these landmark-observations are encountered. We present a method for finding landmarks in *mPSRs*, develop the basic theory of *mPSRs* and provide preliminary empirical results exploring the relative efficiency of learning *mPSRs* and PSRs.

2 PSRs and memory-PSRs

PSRs depart from other models of dynamical systems in that their representation of state is a vector of predictions of the outcomes of *tests* that may be performed on the dynamical system. A test $t = a_1 o_1, \dots, a_k o_k$ is a sequence of alternating actions $a_i \in \mathcal{A}$ and observations $o_j \in \mathcal{O}$, and its *prediction*, $p(t)$, is the conditional probability that the observation sequence occurs, given that the action sequence is taken, and so $p(t) = \text{prob}(o_1, \dots, o_k | a_1, \dots, a_k)$. Of course, the prediction of a test is dependent on the actions and observations that have occurred so far, called the *history*. The prediction of a test t at history h is $p(t|h) = \text{prob}(o_1, \dots, o_k | h a_1, \dots, a_k)$.

A PSR’s state representation consists of predictions of a special set Q of tests, called the *core* tests. The core tests are special because at any history, the predictions for *any* test can be computed as a linear function of the predictions of the core tests. The prediction vector $p(Q|h)$ is the $(n = |Q| \times 1)$ vector of predictions for the tests in Q at history h . Thus, $p(Q|h)$ is the PSR’s representation of state and is the counterpart to belief-states in POMDPs and the last k observations in k -order Markov models.

In addition to the set of core tests, a PSR has model parameters: a set of $(n \times n)$ matrices M_{ao} , and $(n \times 1)$ vectors m_{ao} , for all a, o . The model parameters allow linear computation of the prediction for any test $t = \{a_1 o_1 \dots a_k o_k\}$ as follows:

$$p(t|h) = p(Q|h)^T M_{a_1 o_1} \dots M_{a_{k-1} o_{k-1}} m_{a_k o_k}. \quad (1)$$

Updating the current state, or prediction vector, when action a is taken in history h and o is observed, is accomplished by

$$p(Q|hao)^T = \frac{p(aoQ|h)}{p(ao|h)} = \frac{p(Q|h)^T M_{ao}}{p(Q|h)^T m_{ao}}. \quad (2)$$

The matrices M_{ao} and vectors m_{ao} have a special form. The i^{th} column of M_{ao} is the $(n \times 1)$ constant vector that computes the prediction of the (ao) one-step extension of the i^{th} core test $q_i \in Q$, i.e., of test aoq_i . The vector m_{ao} is the constant $(n \times 1)$ vector that computes the prediction for the one-step test $t = ao$. The fact that the model parameters have these meanings is the foundation of existing PSR learning algorithms [James and Singh, 2004; Rosencrantz *et al.*, 2004] as well as the new mPSR learning algorithm presented here.

		Tests			
		t_1	$\dots$	t_j	$\dots$
Histories	$\phi = h_1$	$p(t_1 h_1) \dots p(t_j h_1)$			
	$\vdots$	$\vdots$		$\vdots$	
	h_i	$p(t_1 h_i) \dots p(t_j h_i)$			
	$\vdots$	$\vdots$		$\vdots$	
	$\vdots$	$\vdots$		$\vdots$	

Figure 1: The system dynamics matrix.

System Dynamics Matrix

The theoretical development of mPSRs is best explained using the system dynamics matrix, D , that was developed in Singh *et al.* [2004]. The matrix D , shown in Figure 1, has all possible histories as rows and all possible tests as columns. The columns (rows) are arranged by test (history) length and within the same length in lexicographic ordering. The entry $D_{ij} = p(t_j | h_i)$ is the prediction for the j^{th} test at the i^{th} history. There is a one to one correspondence between system dynamics matrices and dynamical systems. *The rank of D is the dimension of the corresponding dynamical system and is the number of core tests needed by the PSR model.* The core tests of a PSR model correspond to a maximal set of linearly independent columns. Similarly, we can define a notion of *core histories* that correspond to a maximal set of linearly independent rows. We will only be interested in finite dimensional systems. This assumption is not terribly restrictive because all POMDPs with n underlying or nominal states are dynamical systems of dimension at most n . A full derivation of PSRs through use of the system dynamics matrix is presented in Singh *et al.* [2004].

memory-PSRs

The central idea behind mPSRs is to *partition* D into a set of m submatrices $D^1 \dots D^m$ by partitioning the set of histories (rows) but not the tests (columns), i.e., each submatrix D^i contains all the columns but only a subset of the histories. How do memories enter this picture? We will use memory of past observations to partition histories, and every memory will correspond to a partition; we elaborate on this later. For now, note that each submatrix will have its own rank and its own core tests and core histories as well as model-update

parameters. The set of memories and core tests for each partition form the basis of mPSR models. It is straightforward that the rank of each submatrix is at most the rank of the full matrix D . In the worst case, the ranks of all the submatrices are the same as the rank of D , in which case the resulting mPSR model may have (many) more parameters than the PSR model. But in other cases the ranks of the submatrices may be much smaller than the rank of D and then the resulting mPSR model may be more compact than the PSR model. We provide examples of both cases in Section 5. The size of the model is important because a model with fewer parameters should in general be more efficient (i.e., require less data) to learn. We also test this empirically in Section 5.

In this paper we do not address the question of automatically discovering useful ways of partitioning histories and instead assume that partitions correspond to history suffixes of some fixed length.

Definition of memory: For this paper, given an mPSR using length k suffixes, a *memory* is a specific sequence of length k that identifies a partition. So, when considering a history h and memories of length one, the memory at h is just the last (most recent) observation in h , and there will be $|\mathcal{O}|$ memories in the mPSR. For memories of length two, the memory at h is the last (most recent) action-observation pair in h , and there will be $|\mathcal{A}| * |\mathcal{O}|$ memories in the mPSR. In general, the set of possible memories in an mPSR that uses length- k suffixes is the set of all length- k sequences of alternating actions and observations that end in an observation; we will denote the size of such a set as m_k .

Let $\mu_1 \dots \mu_{m_k}$ represent all the distinct memories in an mPSR model. Also, let the memory at history h be denoted $\mu(h)$. Each memory μ_i has a corresponding submatrix D^{μ_i} created by the partition of histories corresponding to μ_i . The core tests for partition D^{μ} are referred to as μ -core tests to distinguish them from the core tests of the PSR model of the same system. Thus, the μ -core tests for the submatrices $D^{\mu_1} \dots D^{\mu_{m_k}}$ are denoted $Q^{\mu_1} \dots Q^{\mu_{m_k}}$ respectively. The prediction vector $p(Q^{\mu(h)} | h)$ contains the predictions for the μ -core tests $Q^{\mu(h)}$ at memory μ_h .

Definition of mPSR state: The *mPSR state* at history h is denoted by the concatenation of the memory at h and the prediction vector for that memory at h , i.e., as $[\mu(h), p(Q^{\mu(h)} | h)]$. Thus, the mPSR state contains both a memory of the past as well as predictions about the future while a PSR state contains only predictions about the future.

Definition of mPSR update parameters: For each memory, μ_i , we will keep a set of matrices $M_{ao}^{\mu_i}$ and vectors $m_{ao}^{\mu_i}$ for all a, o . There is a subtle issue in defining these parameters. It is not the case that each submatrix is a separate dynamical system. Indeed if the memory corresponding to current history h is say μ_i and we take action a and observe o , the memory corresponding to the next history hao could be different from μ_i . Lets say it is μ_j . Thus the update parameters $M_{ao}^{\mu_i}$ must transform the current prediction vector that makes prediction for μ -core tests Q^{μ_i} in history h to the prediction vector for μ -core tests Q^{μ_j} in history hao . Note that all histories belonging to memory μ_i will transition to the same memory μ_j for action-observation pair ao , i.e., j is uniquely deter-

mined by i and the pair ao ; and so the model parameter $M_{ao}^{\mu_i}$ need not explicitly reference to the next memory μ_j . Thus one can define the state update for mPSRs as follows: upon taking action a in history h and observing o

$$p(Q^{\mu_j}|hao) = \frac{p(aoQ^{\mu_j}|h)}{p(ao|h)} = \frac{p(Q^{\mu_i}|h)^T M_{ao}^{\mu_i}}{p(Q^{\mu_i}|h)^T m_{ao}^{\mu_i}} \quad (3)$$

where $\mu(h) = \mu_i$ and $\mu(hao) = \mu_j$. The matrix $M_{ao}^{\mu_i}$ is of size $(|Q^{\mu_i}| \times |Q^{\mu_j}|)$ and the vector $m_{ao}^{\mu_i}$ is of size $(|Q^{\mu_i}| \times 1)$. As in PSRs the update parameters in mPSRs have meaning. For instance, the k^{th} column of $M_{ao}^{\mu_i}$ is the $(|Q^{\mu_i}| \times 1)$ -sized constant parameter vector that computes the prediction for the test that is the k^{th} μ -core test for memory μ_j .

The mPSR model parameters allow linear computation of the prediction for any test $t = \{a_1 o_1 \dots a_k o_k\}$ at history h as follows:

$$p(t|h) = p(Q|h)^T M_{a_1 o_1}^{\mu_0} \dots M_{a_{k-1} o_{k-1}}^{\mu_{k-2}} m_{a_k o_k}^{\mu_{k-1}}. \quad (4)$$

where μ_0 is the memory corresponding to history h , μ_1 is the memory corresponding to history $ha_1 o_1$ and so on.

Definition of mPSR: An mPSR model is defined by the tuple $\langle \mathcal{A}, \mathcal{O}, \mu_1 \dots \mu_{m_k}, Q^{\mu_1} \dots Q^{\mu_{m_k}}, \mathcal{M}, [\mu_0, p(Q^{\mu_0}|\emptyset)] \rangle$ where $\mathcal{A}$ is the set of actions; $\mathcal{O}$ is the set of observations; $\mu_1 \dots \mu_{m_k}$ are the possible memories; Q^{μ_i} is the set of μ -core tests for memory μ_i ; $\mathcal{M}$ is the set of update parameters $M_{ao}^{\mu_i}$ and $m_{ao}^{\mu_i}$ for all a, o, μ_i ; μ_0 is the initial memory; and $p(Q^{\mu_0}|\emptyset)$ is the initial prediction vector.

A special and interesting case arises when a memory by itself serves as state and no predictions are needed.

Definition of landmark: Given an mPSR, a *landmark* is a memory that serves as state, i.e., is a sufficient statistic of history. Landmarks can be quite beneficial for making accurate predictions. We will discuss the identification of landmarks and exploiting them for prediction in Section 3.1.

3 Basic Theory of mPSRs

Our first result is to show that for any dynamical system it is possible to find μ -core tests for all memories in an mPSR model from the set Q of core tests for the corresponding PSR model.

Lemma 1. *Let a dynamical system be modeled as a PSR with core tests Q . Then there always exists an mPSR model for the system such that for each memory μ_i the corresponding μ -core tests Q^{μ_i} satisfy $Q^{\mu_i} \subset Q$.*

Proof. We provide a constructive proof which shows how to derive the subset Q^{μ_i} from Q . Recall that all columns of D are linear combinations of the columns corresponding to Q in D . Consider the submatrix D^{μ_i} for memory μ_i . It must be the case that all columns of D^{μ_i} are linear combinations of the columns corresponding to Q in D^{μ_i} . Thus, there exists $Q^{\mu_i} \subset Q$. $\square$

In this paper, we don't exploit Lemma 1 for two reasons: 1) the core tests of PSRs are not unique, and 2) in any case when learning a model of a dynamical system from experience data we do not have a PSR model and thus its core tests Q to begin with.

Lemma 2. *For any dynamical system of finite dimension and any choice of (fixed-length suffix) memories, the size of the resulting mPSR model for that system is at most the number of memories times the size of a PSR model for that system.*

Proof. In the worst case, the rank of each of the submatrices in the mPSR is exactly the rank of the full system dynamics matrix. In such a case, if we happen to find different μ -core tests for each submatrix, then the model for each submatrix will be exactly as large as the PSR model. $\square$

The above lemma holds no matter how one chooses the memories. But what if we can choose the memories to use in constructing an mPSR judiciously (by which we mean to minimize the size of the resulting model)?

Theorem 1. *With a judicious choice of memories, the size of the corresponding mPSR for any dynamical system is at least as compact as the size of the PSR for that dynamical system. Furthermore, there exist dynamical systems for which some choice of memories leads to an mPSR model that is more compact than the PSR model of the same dynamical system.*

Proof. The proof of the first statement follows from the fact that a PSR is also an mPSR with the null set as memories. We prove the second statement by constructing such a dynamical system. Table 1 compares the size of mPSRs and PSRs for various standard test dynamical systems (based on POMDPs). In all instances, a suffix of length one was used to partition the histories. For at least three problems, Cheese, Shuttle and Four-Three, the mPSR is more compact than the PSR. $\square$

3.1 Landmarks

We show that landmarks are equivalent to memories for which there is only one μ -core test and furthermore that the prediction of any test at a landmark is the same at all histories that map to that landmark.

Lemma 3. *For an mPSR model of a dynamical system,*

- *any memory that is a landmark has only one μ -core test, and every memory that has a μ -core set of size one is a landmark.*
- *for all histories that map to a landmark the prediction vector is a constant, i.e., independent of history.*
- *for all histories that map to a landmark the prediction of any test t is a constant, i.e., independent of history.*

Proof. If μ is a landmark, then the predictions for all tests are fully determined when μ is known. Therefore, at any two histories that have memory μ , the prediction of every test is the same at both histories. This means that every row of D^{μ} must be identical, so the rank of D^{μ} is one, and only one μ -core test exists for μ .

For a memory μ that has a single μ -core test, q^{μ} , let H^{μ} denote the histories that map to memory μ . For any $h \in H^{\mu}$, and any action a , it must hold that $\sum_o p(ao|h) = 1$ which implies that $\sum_o p(q^{\mu}|h)^T m_{ao}^{\mu} = 1$, which in turn implies that $p(q^{\mu}|h)^T = 1 / \sum_o m_{ao}^{\mu}$. Recall that m_{ao}^{μ} are independent of history, and thus the prediction of q^{μ} must be independent of

Table 1: Comparison of the size of PSRs and mPSRs

Problem	# Core Tests		# Param. Entries	
	PSR	mPSR	PSR	mPSR
Tiger	2	2,2	36	72
Paint	2	2,2	48	96
Cheese	11	1,1,1,1,2,2,3	3696	792
Network	7	4, 6	448	480
Bridge	5	2,2,4,4,5	1800	4488
Shuttle	7	1,1,2,2,4	840	450
Four Three	10	1,1,1,1,3,4	2640	748
Float Reset	5	1,5	120	96

the history. This calculation exploits the fact that the prediction vector and update parameters are scalars.

Furthermore, because the prediction of q^μ is independent of history, and the update vector m_t^μ for any test t is independent of history, the prediction $p(t|h) = p(q^\mu|h)^T m_t^\mu$ of t at history h must also be independent of history. So, all rows of D^μ must be identical, and μ is therefore a landmark. $\square$

Landmarks come in handy because they can be used to keep a learned — and therefore only approximately correct — model from progressively drifting farther and farther away from reality as we make longer and longer term predictions from such a model. Every time we observe a landmark memory, we can reset the prediction vector to the constant corresponding to the landmark irrespective of the actual observed history. This can keep the error in the prediction for very long tests from growing with the length of the test. We exploit this ability in our empirical results below.

4 Learning mPSR models from data

In this section we present an algorithm for learning mPSR models from data under the assumption that the algorithm has the ability to reset the dynamical system being modeled to an initial state. We present the algorithm in two stages. In the first stage, we will assume that the algorithm has access to an oracle that if given a history h and test t will return the prediction $p(t|h)$ for the dynamical system. In the second stage we will show how the oracle can be replaced by the use of sample data from the dynamical system with reset. Because we wish to minimize the amount of sample data needed by our algorithm, we will attempt to minimize the number of calls to the oracle in the first stage. Our algorithm is a relatively straightforward adaptation to mPSRs of the algorithm presented by James and Singh [2004] for PSRs.

4.1 Oracle based algorithm

Recall that there are two components to an mPSR: the μ -core tests for the different memories and the model-update parameters for the memories. In our empirical work below, we always use memories of length one. Clearly the model parameters for a memory depend on the choice of μ -core tests for that memory (recall that the μ -core tests are not unique). The process of computing μ -core tests and update parameters for each memory is identical and so we can just discuss how to do this for any one memory, say μ_i .

Determining μ -core tests and histories

The algorithm proceeds in iterations. At iteration t , the algorithm has a current set of linearly independent or μ -core tests and histories for each memory. Let $H_t^{\mu_i}$ ($T_t^{\mu_i}$) be the set of μ -core tests (μ -core histories) found by iteration t for memory μ_i . These μ -core tests (histories) start off as the empty set at the first iteration. We also keep a set of candidate tests and histories at each iteration. The set of candidate tests for memory μ_i is the set of one-step tests; and for every ao pair, the set of ao -extensions for all the current μ -core tests for memory μ_j , where μ_j is the next memory when action a is taken in memory μ_i and o is observed. The set of candidate histories is similarly generated. We ask the oracle for the predictions of all the candidate tests at all the candidate histories. We compute the rank of this oracle-generated matrix, denoted X_t . The linearly independent rows and columns of X_t become the new $H_{t+1}^{\mu_i}$ and $T_{t+1}^{\mu_i}$ respectively. In selecting the new μ -core tests and histories we always include the previous μ -core tests and histories. We stop at iteration s if the rank of X_s is the same as the rank of X_{s-1} for all memories.

The above algorithm is an adaptation of the similar algorithm for determining core test and histories for PSRs by James and Singh [2004]. Just like for PSRs this algorithm is not guaranteed to find all μ -core tests and histories, but also just like for PSRs it seems to work for all empirical work done so far.

Computing the model-update parameters

Now we discuss how to compute the model-update matrix M_{ao}^μ for any ao pair and any memory μ . We define a matrix A for memory μ that contains the predictions for all the μ -core tests at all the μ -core histories for memory μ . This matrix will have full rank and is computed in the algorithm above. Let μ_j be the memory achieved when action a is taken in memory μ and o is observed. The k^{th} column of M_{ao}^μ , denoted x , is the constant parameter vector that computes the prediction for the ao -extension of the k^{th} μ -core test, denoted y , of memory μ_j . Let b be the column vector of predictions of test y for the μ -core histories of memory μ . We ask the oracle for the entries of b . Then from Equation 3 above, $Ax = b$. Since A is full rank, it is invertible and hence $x = A^{-1}b$. We can use the same idea to compute the model-update parameter vector m_{ao}^μ .

In the next section we show how the oracle needed for the above algorithm can be replaced by using data from the dynamical system with reset.

4.2 Learning mPSRs from sample data

We show how to obtain an estimate for the prediction $p(t|h)$, something we went to the oracle for in the previous algorithm. We generate samples from the distribution $p(t|h)$ by first taking the action sequence in h after a reset. If the observation sequence in h happens, then we have succeeded in generating history h . If we don't succeed in generating the history, we reset the system and try again. If we do succeed in generating the history, we take the action sequence in test t . If the observation sequence in test t happens, the test succeeds, else the test fails. We get an estimate of $p(t|h)$ from the empirical success rate of the test t at history h . Of course, this is ex-

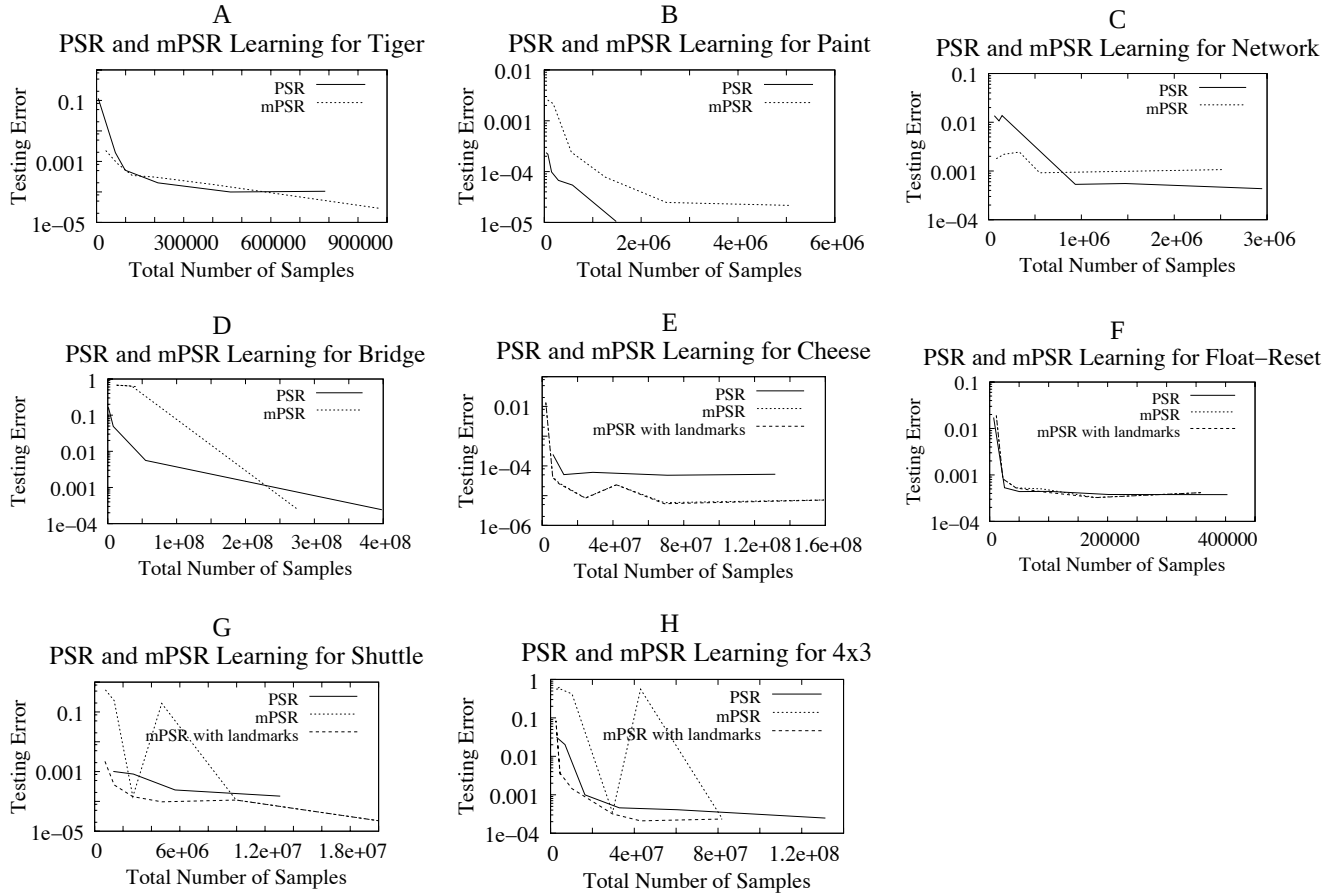


Figure 2: Results of running mPSR learning algorithms on a suite of test problems. See text for details.

tremely wasteful in the number of data samples needed. We implement this basic idea much more efficiently by executing the action sequence in *ht* regardless of the observations obtained and then mining the data generated for all possible history and test pairs that can be extracted from it.

One problem that occurs with this estimation technique is that the sampled entries of the system dynamics matrix will almost certainly result in inaccurate calculations of rank (needed in the above oracle-based algorithm). We use a more robust rank calculation procedure developed in the PSR learning algorithm by James and Singh [2004]. The central idea is that we can use the number of samples that went into each entry of the matrix we wish to find the rank of to generate a threshold on singular values obtained via singular value decomposition. This threshold is more conservative when we have fewer samples and generally leads to a smaller rank than otherwise. As more samples are included, the threshold becomes less conservative and the calculated rank is closer to the straightforward calculation of rank.

5 Empirical results

We conducted experiments with a view towards validating four major ideas presented above. The first experiments are meant to compare the relative size of PSRs and mPSRs. The

second is to test whether the oracle-based algorithm for computing mPSRs finds exact mPSR models. The third is to test the efficacy of the sample-data based learning algorithm for mPSRs. Finally, we are also interested in whether using landmarks, where available, to reset prediction vectors during the process of computing predictions for long test sequences provides a measurable benefit. All of these experiments were on a suite of standard POMDP-based dynamical systems [Cassandra, 1999] that have been used for both POMDP-learning as well for PSR-learning.

Again, for all of the results presented below, the memories used for the mPSR models were the most recent observation.

5.1 Comparing PSRs and mPSRs

For PSRs and mPSRs, the size of the model can be measured both in terms of the number of core tests (μ -core tests), and in the number of entries in the model parameters. Table 1 compares the sizes of PSR and mPSR models for the suite of test problems. For mPSRs, the number of μ -core tests at each memory is listed. Fully half of the test problems have landmarks (memories with only one μ -core test), indicating that this may be a fairly common situation.

The number of entries in the model parameters are listed for both PSRs and mPSRs. On three of the problems (Cheese,

Shuttle, and Four-Three) there are significantly fewer entries needed for the mPSR model than for the PSR model; on three other problems (Tiger, Bridge, and Paint) there are significantly more; and on the two remaining problems there are approximately an equal number needed. This proves that mPSRs can be more compact than PSRs. This also illustrates that the unless we choose memories wisely, mPSRs can also be less compact than PSRs.

5.2 Constructing mPSRs from exact predictions

We tested the oracle algorithm for computing mPSRs by providing access to the true predictions computed analytically for the test problems. As we mentioned in Section 4.1, this algorithm is not guaranteed to find all μ -core tests for every memory. We found that in fact for all the test problems, the algorithm does indeed find all the μ -core tests and histories for all the models. We also verified that the resulting mPSR models were perfect, in the sense that any prediction error was due to machine round-off error. Furthermore, because correct sets of μ -core tests were discovered, all landmarks were identified.

5.3 Learning mPSRs from samples

Here we present the results of learning mPSR models for all our dynamical systems. As mentioned above we assumed a reset. This makes our results directly comparable to the results of James and Singh [2004] on the same set of problems under the same reset assumption. We applied the algorithm of Section 4.2, stopping every so often during learning to test the model learned so far. For each test of the learned model, we asked the model to make predictions while tracking the system for 100,000 steps (with randomly chosen actions). The test error reported is the average error in the one-step predictions of the model relative to the true predictions. This is the same error measure as used in James and Singh [2004].

The results of this testing are presented in Figure 2. We compare our results to the results for PSR learning for the same dynamical systems from James and Singh [2004]. Plots A to D present the results for the systems without any landmarks, and show the results of PSR and mPSR learning. In two of these systems, PSR learning is more efficient while in the other two they are roughly equivalent. We note that in all of these problems the PSR models are more compact or equal in size than the mPSR models. Plots E to H are for systems with landmarks. Here we plot three results: PSR learning, mPSR learning and mPSR learning with landmarks. The last algorithm, mPSR learning with landmarks, uses the mPSR learning algorithm of Section 4.2 but during testing resets the prediction vector when a landmark memory is encountered to the constant prediction for that memory. In three of the systems corresponding to graphs E, G and H, the size of the mPSR model is smaller than the PSR model. As we surmised in Section 2 this leads to mPSR learning being more efficient than PSR learning. In the system for plot F the two are roughly equivalent as are their sizes. The effect of using landmarks is most easily seen in plots G and H. Here mPSR learning without using landmarks is very noisy for if the system gets off-track in its predictions the error accumulates rapidly as the test sequence lengthens, while using the

landmarks stabilizes the error and gives the best results.

6 Conclusion

In this paper we proposed a new class of models, mPSRs, that combines memory of past observations and predictions of future observations into its state representation. We showed that mPSRs are as expressive as and can be more compact than PSRs that solely use predictions of future observations in its state representation. Our preliminary empirical results showed that in dynamical systems where the mPSR model of the system is more compact than the PSR model, learning the mPSR model is more efficient than learning the PSR model. Finally, we formalized the notion of landmark memories and demonstrated how to find them and how to exploit them in making long-term predictions.

As future work, we will explore the interesting challenge of automatically finding good memories to combine with predictions of tests to make compact and efficiently learnable mPSR models.

Acknowledgements The research reported in this paper was supported by NSF grant IIS-0413004. The authors also thank Rich Sutton for helpful comments.

References

- [Cassandra, 1999] A. Cassandra. Tony's pomdp page. <http://www.cs.brown.edu/research/ai/pomdp/index.html>, 1999.
- [Golub and Van Loan, 1996] G.H. Golub and Ch. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.
- [James and Singh, 2004] Michael R. James and Satinder Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *The 21st International Conference on Machine Learning*, 2004.
- [Littman et al., 2001] Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Advances In Neural Information Processing Systems 14*, 2001.
- [Rosencrantz et al., 2004] Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *The Twenty-First International Conference on Machine Learning*, 2004.
- [Singh et al., 2003] Satinder Singh, Michael L. Littman, Nicholas K. Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *The Twentieth International Conference on Machine Learning*, 2003.
- [Singh et al., 2004] Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations, a new theory for modeling dynamical systems. In *20th Conference on Uncertainty in Artificial Intelligence*, 2004.
- [Sutton and Tanner, 2004] R. S. Sutton and B. Tanner. Temporal-difference networks. In *Advances in Neural Information Processing Systems 17*, 2004.

Learning with Labeled Sessions

Rong Jin*, Huan Liu†

*Dept. of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824

rongjin@cse.msu.edu

† Department of Computer Science and Engineering, Arizona State University, Tempe, AZ85287-8809

hliu@asu.edu

Abstract

Traditional supervised learning deals with labeled instances. In many applications such as physiological data modeling and speaker identification, however, training examples are often *labeled objects* and each of the labeled objects consists of *multiple unlabeled instances*. When classifying a new object, its class is determined by the majority of its instance classes. As a consequence of this decision rule, one challenge to learning with labeled objects (or sessions) is to determine during training which subset of the instances inside an object should belong to the class of the object. We call this type of learning ‘*session-based learning*’ to distinguish it from the traditional supervised learning. In this paper, we introduce session-based learning problems, give a formal description of session-based learning in the context of related work, and propose an approach that is particularly designed for session-based learning. Empirical studies with UCI datasets and real-world data show that the proposed approach is effective for session-based learning.

1 Introduction

A typical supervised learning problem is to learn a model from training examples that are usually labeled instances. But for many applications, training examples are *labeled objects* and each labeled object consists of *multiple unlabeled instances*. During classification, an object is labeled as class ‘*a*’ if the majority of its instances are classified as ‘*a*’.

One application is physiological data modeling, whose goal is to predict context activities of individual users based on their physiological data. Typically, physiological signals are measured and recorded continuously for every other second. Continuous physiological signals are then divided into a number of sessions. Each session is of minutes long and consists of hundreds of records of physiological data. To predict the user’s activities for a session, prediction is first made for each record, and the dominant activity predicted for records in the session is then used as the activity for the session. Although each session is labeled as a single activity, a user may perform activities other than the labeled one. For example, during a session of ‘watching TV’, the

user may fall into sleep for a short period of time. More information can be found from the website <http://www.cs.utexas.edu/users/sherstov/pdmc/>.

Another application is speaker identification. To determine the speaker for a speech sample that is a minute long, a typical strategy is to first divide the long speech sample into a number of short ones. Then, a standard classification model, such as Gaussian Mixture Model (GMM), is applied to determine the speaker identity for each short sample. Finally, the dominant speaker that is classified for short samples will be used as the predicted speaker for the long sample. Compared with the strategy that extracts a single set of features for the whole long speech sample, this majority vote approach is usually more robust and accurate [Reynolds, 1995]. This is because features extracted from a long speech sample may include significant amounts of background noise, while the noise can be reduced substantially when a long speech sample is divided into short ones.

The common characteristics of the above two applications are that (1) training examples are labeled objects (e.g., sessions of physiological records in physiological data modeling, and a long speech sample in speaker identification); (2) every object consists of multiple instances that are not labeled; and (3) in predicting an object’s class, it is determined by the class that is assigned to the majority of its instances. To distinguish this new type of learning from the traditional supervised learning, we call it ‘**session-based learning**’.

The challenge of session-based learning arises from the majority vote strategy that is used to determine the label of an object. This decision rule makes it ambiguous, during training, as to which subset of the instances inside an object should belong to the class of the object – we name it the **label ambiguity problem**. A straightforward strategy toward this problem is to treat every unlabeled instance within a labeled object as a positive example for the class of the object. In physiological data modeling, every physiological record within a labeled session is treated as a training instance for the activity assigned to the session. In speaker identification, every short sample within a long speech is used as a positive training instance for the speaker of the long speech. We call this simple strategy the ‘**naive approach**’.

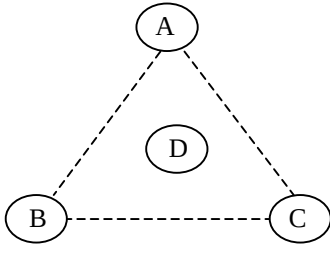


Figure 1: A toy example of session-based learning problem with four classes ‘A’, ‘B’, ‘C’, and ‘D’.

One obvious problem with the naïve approach is that instances within a single object often belong to multiple different classes, not just to the class of the object. Thus, by treating every instance within an object as a positive example for the class of the object, we likely introduce training examples with noisy labels, which, as a result, degrades the quality of classification models. To further illustrate the problem with the naïve approach, let us consider a toy learning problem in Figure 1. It has four classes, with class ‘A’, ‘B’, and ‘C’ centering on the vertices of a triangle, and the fourth class ‘D’ sitting on the center of the triangle. In traditional supervised learning, training examples are labeled instances. Since these four classes are well separated, we would expect that a simple Naïve Bayes model should work fine for this problem. But, for a session-based learning problem, training examples are labeled objects that consist of instances from different classes. In particular, consider the case when training objects for classes ‘A’, ‘B’, and ‘C’ are mixtures of instances from these three classes, while training objects for class ‘D’ only contain instances from ‘D’. With appropriate mixtures, the input means of instances from labeled objects for the four classes can stay close to each other, which makes it hard for the Naïve Bayes method to learn if assigning each instance with its object class.

In the following, we first give a formal description of session-based learning, and elaborate on the differences between this new type of learning problem and other related learning problems, such as multiple-instance learning. Then, we present a novel approach that is particularly designed for session-based learning. The key idea is to develop an innovative way that handles the label ambiguity problem. Finally, we demonstrate the effectiveness of the proposed approach for session-based learning using both UCI datasets and data from physiological data modeling.

2 Formal Description of Session-based Learning

Let training examples be denoted by $D = \{(o_1, y_1), (o_2, y_2), \dots, (o_n, y_n)\}$, where each (o_i, y_i) is a labeled object and $y_i \in \mathbf{Y}$ is the class label assigned to object o_i . Domain $\mathbf{Y}$ is $\{-1, 1\}$ for binary-class classification problems, and $[1..K]$ for multiple-class classification prob-

lems where K ($K > 2$) is the number of classes. Let $c(o)$ be an object-based classification function, which takes an object o as input and outputs its class label. In general, any supervised learning problem can be formulated as an optimization problem:

$$c^* = \arg \min_c \sum_{i=1}^n L(c(o_i), y_i) \quad (1)$$

where L is a loss function that determines the amount of punishment when prediction $c(o_i)$ is different from y_i .

For a traditional supervised learning problem, each object only contains a single instance. As a result, training examples can be simplified as $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$,

where each instance $\mathbf{x}_i \in \mathcal{R}^d$ is a vector in a d dimension space. Furthermore, the object-based classification function $c(o)$ becomes an instance-based classification function $f(\mathbf{x}): \mathcal{R}^d \rightarrow \mathbf{Y}$. Thus, a traditional supervised learning problem is usually formulated as follows:

$$f^* = \arg \min_f \sum_{i=1}^n L(f(\mathbf{x}_i), y_i) \quad (1a)$$

In session-based learning, each object consists of multiple unlabeled instances. Let the training data be denoted by $D = \{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_n, y_n)\}$, where each object

$\mathbf{X}_i = \{\mathbf{x}_{i,j}\}_{j=1}^{m_i}$ contains m_i different instances. Given an

instance-based classification function $f(\mathbf{x}): \mathcal{R}^d \rightarrow \mathbf{Y}$, the decision rule for session-based learning is that, an object $\mathbf{X}_i$ is labeled as class ‘a’ if the majority of its instances are classified as ‘a’. Thus, in session-based learning, the object-based classification function c_s can be written into the following form of the instance-based classification $f(\mathbf{x})$:

$$c_s(\mathbf{X}_i; f) = \arg \max_{y \in \mathbf{Y}} \sum_{j=1}^{m_i} \delta(f(\mathbf{x}_{i,j}) = y) \quad (2)$$

where $\delta(\cdot)$ is a delta function that outputs 1 when the input is positive and zero otherwise.

Compared to traditional supervised learning, the challenge of session-based learning is due to the label ambiguity problem. Although each training object is provided with a class label, the label information of instances within objects is not given. Hence, it is difficult to learn an instance-based classification function from labeled objects. In the naïve approach, each instance within an object is treated as a positive example for the class of the object. As a result, a session-based learning problem is simplified as a traditional supervised learning problem:

$$f^* = \arg \min_f \sum_{i=1}^n \sum_{j=1}^{m_i} L(f(\mathbf{x}_{i,j}), y_i) \quad (1b)$$

The most related work to session-based learning is multiple-instance learning [Dietterich, et al., 1997]. Similar to session-based learning, in multiple-instance learning, class la-

bels are assigned to objects that consist of multiple instances, which are called “bags” in multiple-instance learning. In the past, there have been many studies on multiple-instance learning, including the approach of learning axis-parallel rectangles [Dietterich, et al.,1997], the diverse density algorithm [Maron and Lozano-Pérez,1998], the approaches based on support vector machines [Andrews, et al.,2002, Tao, et al.,2004], the nearest neighbor approach [Amar et.al. 2001; Wang and Zucker, 2000], and the boosting approach [Andrews and Hofmann,2003].

Multiple-instance learning differs from session-based learning in its decision rule. In multiple-instance learning, an object $\mathbf{X}_i$ is labeled as positive class when at least one of its instances is classified as positive. A negative class is assigned to an object when all of its instances are classified as negative. Thus, given the instance-based classification function $f(\mathbf{x}): \mathcal{R}^d \rightarrow \mathbf{Y}$, the object-based classification function c_m for multiple-instance learning is written as:

$$c_m(\mathbf{X}_i; f) = \begin{cases} +1 & \exists j \in [1 \dots m_i], f(\mathbf{x}_{i,j}) = +1 \\ -1 & \forall i \in [1 \dots m_i], f(\mathbf{x}_{i,j}) = -1 \end{cases} \quad (2')$$

Examples for the three different types of learning are shown in Figure 2 for comparison. The first column shows traditional supervised learning with instances and their labels. The columns for Multiple Instance Learning and Session-based Learning show how these two different strategies aggregate instances and assign labels to the aggregates.

Session-based learning is more challenging than multiple-instance learning in the following sense: in multiple-instance learning, when an object is labeled as ‘negative’, all of its instances will belong to the negative class. Thus, for multiple-instance learning, there is no label ambiguity for negatively labeled objects. In contrast, the label ambiguity problem exists for session-based learning regardless of the sign of labels. For instance, in Figure 2, for session-based learning, both the first and second objects are labeled as ‘negative’. However, the labels of instances in these two objects are different. The two learning schemes also differ in degrees of difficulty when deciding positive classes for objects: multiple-instance learning adopts the “at-least one” strategy and session based learning uses the majority one.

3 SBoost – An Algorithm for Session-based Learning

In this session, we will present a boosting-based algorithm for session-based learning problems of binary classes. The key for designing a learning algorithm for session-based learning is to define a simple yet effective loss function $L(c(\mathbf{X}_i; f), y_i)$. A simple choice is $L(c(\mathbf{X}_i; f), y_i) = \delta(c(\mathbf{X}_i; f) \neq y_i)$. However, this choice will lead to a non-smooth objective function, which is usually difficult for optimization. Hence, we choose to use the exponential loss function to approximate classification er-

Trad. Super- vised Learning	Multiple-instance Learning	Session-based Learning
0.1 0.2 0.3 -1	0.1 0.2 0.3	0.1 0.2 0.3
0.2 0.5 0.4 -1	0.2 0.5 0.4	0.2 0.5 0.4
0.6 0.2 0.1 -1	0.6 0.2 0.1	0.6 0.2 0.1
.....		
0.3 0.7 0.6 -1	0.3 0.7 0.6	0.3 0.7 0.6
0.7 0.2 0.2 -1	0.7 0.2 0.2	0.7 0.2 0.2
0.8 0.1 0.0 +1	0.8 0.1 0.0	0.8 0.1 0.0
.....		
0.5 0.7 0.1 +1	0.5 0.7 0.1	0.5 0.7 0.1
0.1 0.6 0.9 -1	0.1 0.6 0.9	0.1 0.6 0.9
0.7 0.1 0.2 +1	0.7 0.1 0.2	0.7 0.1 0.2

Figure 2: Training examples for traditional supervised learning, multiple-instance learning, and session-based learning.

rors, which has been demonstrated to be effective in the AdaBoost algorithm [Freund and Schapire, 1997].

In particular, in designing objective functions, two types of errors are considered: instance-based errors and session-based errors. An instance-based error is a prediction mistake made for an instance, and a session-based error is a prediction mistake made for a session. One key difference between session-based learning and traditional supervised learning is that the former is concerned with both session-based errors and instance-based errors while the latter concerns with only instance-based errors. To include both types of errors, given an instance-based classification function $H(x)$, we define the loss function as:

$$L(c(\mathbf{X}_i; H), y_i) = \underbrace{\exp\left(-\frac{\gamma y_i}{m_i} \sum_{j=1}^{m_i} H(x_{i,j})\right)}_{\text{Session-based error}} \underbrace{\left\{ \sum_{j=1}^{m_i} \exp(-H(x_{i,j})y_i) \right\}}_{\text{Instance-based error}} \quad (3)$$

In the above expression, both session-based and instance-based errors are approximated by an exponential function. The loss function is defined as the product of these two errors. In other words, a misclassified instance is important only when its related session is also misclassified. Constant γ in (3) determines the relative importance between the session-based error and the instance-based error. In experiment, a cross validation with 20/80 split of training data is used to determine appropriate values for γ . In the following, we will discuss how to efficiently find $H(x)$ that minimizes the loss function in (3).

3.1 Boosting-based Optimization Algorithm

With the loss function defined in (3), our goal is then to search for optimal $H(x)$ that minimizes the overall cost for the training data, i.e.,

$$\begin{aligned} H^* &= \arg \min_H \text{err} = \arg \min_H \sum_{i=1}^n L(c(\mathbf{X}_i; H), y_i) \\ &= \arg \min_H \sum_{i=1}^n \exp\left(-\frac{\gamma y_i}{m_i} \sum_{j=1}^{m_i} H(x_{i,j})\right) \left\{ \sum_{j=1}^{m_i} \exp(-H(x_{i,j})y_i) \right\} \end{aligned} \quad (4)$$

Given: $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_m, y_m)$ where $\mathbf{X}_i = \{\mathbf{x}_{i,j}\}_{j=1}^{m_i}$, $\mathbf{x}_{i,j} \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$; Weighting constant γ

Initialize the weight distribution $D_0(i, j) = 1/\left(\sum_{i=1}^n m_i\right)$, $H_0(x) = 0$

For $t = 1, \dots, T$

1. Sample training instances $\{\mathbf{x}_{1,1}, \dots, \mathbf{x}_{1,m_1}, \dots, \mathbf{x}_{n,1}, \dots, \mathbf{x}_{n,m_n}\}$ according to D_{t-1}
2. Train a weak classifier $h_t(\mathbf{x}): \mathbb{R}^d \rightarrow \{-1, 1\}$ on sampled examples
3. Compute $g_i = \exp\left(-\frac{y_i \gamma}{m_i} \sum_{j=1}^{m_i} H_{t-1}(\mathbf{x}_{i,j})\right)$, $a_i = \sum_{j=1}^{m_i} \exp(-H_{t-1}(\mathbf{x}_{i,j})y_i)$, and

$$b_i = \sum_{j=1}^{m_i} y_i h(\mathbf{x}_{i,j}) \exp(-H_{t-1}(\mathbf{x}_{i,j})y_i) + \frac{y_i a_i \gamma}{m_i} \sum_{j=1}^{m_i} h(\mathbf{x}_{i,j})$$
 for each session.
4. Let $\alpha_t = \frac{1}{2(1+\gamma)} \ln \left\{ \frac{\sum_{i=1}^n g_i [(1+\gamma)a_i + b_i]}{\sum_{i=1}^n g_i [(1+\gamma)a_i - b_i]} \right\}$
5. Update the weight distribution $D_t(i, j) = \frac{g_i \left(\exp(-H_{t-1}(\mathbf{x}_{i,j})y_i) + \gamma a_i / m_i \right)}{Z_t}$, where Z_t is a normalization factor (chosen so that D_{t+1} is sum to 1).
6. Update the classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$

Output the final hypothesis: $F_T(\mathbf{x}) = \arg \max_{y \in \{-1, 1\}} \sum_{t=1}^T \alpha_t h_t(\mathbf{x}) y$

Figure 3: Description of the SBoost algorithm.

An efficient approach for optimizing Eq. (4) is to divide it into a series of simple learning problems that do not have the label ambiguity problem and thus can be resolved by traditional supervised learning techniques. We solve the label ambiguity problem by maintaining weights for different instances such that only instances with large weights are assigned to the class of their objects and used for training. Since boosting [Freund and Schapire, 1997] is a learning algorithm that uses weighted instances to efficiently update classification functions, we design a boosting-based learning algorithm for learning with labeled sessions below.

Let $h_t(\mathbf{x})$ be the ‘weak’ classifier of the t -th iteration that is learned using traditional supervised learning techniques. The combined classifier $H_T(\mathbf{x})$ for the first T iterations is

$H_T(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where α_t is the combination constant for the t -th iteration. Our goal is to find another ‘weak’ classifier $h_{T+1}(\mathbf{x})$ and a constant α_{T+1} such that the new combined classifier $H_{T+1}(\mathbf{x}) = H_T(\mathbf{x}) + \alpha_{T+1} h_{T+1}(\mathbf{x})$ will effectively minimize the function in (4). Given classifier $H_{T+1}(\mathbf{x})$, the objective function in (4) is rewritten as:

$$err = \sum_{i=1}^n \left\{ \exp\left(-\frac{\gamma y_i}{m_i} \sum_{j=1}^{m_i} [H(\mathbf{x}_{i,j}) + \alpha h(\mathbf{x}_{i,j})]\right) \times \sum_{j=1}^{m_i} \exp(-[H(\mathbf{x}_{i,j}) + \alpha h(\mathbf{x}_{i,j})]y_i) \right\} \quad (5)$$

In the above expression, for the convenience of presentation, we drop the index for $H_T(\mathbf{x})$, $h_{T+1}(\mathbf{x})$, and α_{T+1} . Using the convexity of an exponential function, we have

$$err \leq \sum_{i=1}^n \frac{g_i}{m_i} \sum_{k,j=1}^{m_i} \exp(-\alpha [h(\mathbf{x}_{i,j}) + \gamma h(\mathbf{x}_{i,k})] y_i - H(\mathbf{x}_{i,j}) y_i) \quad (6)$$

where $g_i \equiv \exp\left(-\frac{y_i \gamma}{m_i} \sum_{j=1}^{m_i} H(\mathbf{x}_{i,j})\right)$. Then, using inequality $e^{\alpha x} \leq \frac{1+x}{2} e^{\alpha} + \frac{1-x}{2} e^{-\alpha}$, $\forall x \in [-1, 1]$, we can further upper bound (6) by the following expression:

$$\begin{aligned}
err &\leq err_{upper} \\
&= \frac{e^{\alpha(1+\gamma)} + e^{-\alpha(1+\gamma)}}{2} \sum_{i=1}^n a_i g_i \\
&\quad - \frac{e^{\alpha(1+\gamma)} - e^{-\alpha(1+\gamma)}}{2(1+\gamma)} \sum_{i=1}^n \sum_{j=1}^{m_i} y_i h(\mathbf{x}_{i,j}) \left\{ g_i \exp(-H(\mathbf{x}_{i,j})y_i) \right. \\
&\quad \left. + g_i \frac{\gamma a_i}{m_i} \right\} \quad (7)
\end{aligned}$$

where $a_i \equiv \sum_{j=1}^{m_i} \exp(-H(\mathbf{x}_{i,j})y_i)$. Define weight

$D(i, j) \equiv g_i \left(\exp(-H(\mathbf{x}_{i,j})y_i) + \gamma a_i / m_i \right)$ and rewrite the second term in (7) as:

$$\frac{e^{\alpha(1+\gamma)} - e^{-\alpha(1+\gamma)}}{2(1+\gamma)} \sum_{i=1}^n \sum_{j=1}^{m_i} y_i h(\mathbf{x}_{i,j}) D(i, j) \quad (8)$$

Clearly, to minimize the objective err in (7), we need to maximize the above expression (8). The best case is that the output of weak classifier $h(\mathbf{x})$ is consistent with y_i , for all instances in an object. When this is not the case, the label ambiguity problem is then resolved based on weight $D(i, j)$. In particular, instances with large weights are assigned with the class of their objects, while the labels for instances with small weights remain unlabeled. This is because the contribution of an instance to (8) is mainly determined by its weight $D(i, j)$. When an instance has a small weight, its contribution to (8) will be ignorable. Furthermore, notice that $D(i, j)$ is proportional to g_i , which is related to session-based error. Thus, a misclassified instance will not be assigned with a large weight if the related session error is small. Finally, the combination constant α can be obtained by setting the derivative of the upper bound in (7) w.r.t. to α to be zero. That is,

$$\alpha = \frac{1}{2(1+\gamma)} \ln \left\{ \frac{\sum_{i=1}^n g_i [(1+\gamma)a_i + b_i]}{\sum_{i=1}^n g_i [(1+\gamma)a_i - b_i]} \right\} \quad (9)$$

where

$$b_i \equiv \sum_{j=1}^{m_i} y_i h(\mathbf{x}_{i,j}) \exp(-H(\mathbf{x}_{i,j})y_i) + \frac{y_i a_i \gamma}{m_i} \sum_{j=1}^{m_i} h(\mathbf{x}_{i,j}).$$

For later reference, we name this algorithm **SBoost** for ‘**session-based boosting**’. The details are given in Figure 3.

4 Experiments

The goal of this section is to examine the effectiveness of SBoost for session-based learning. We compare SBoost with the simple naïve approach that treats each instance within an object as a positive example for the class of the object. In particular, the naïve approach is applied to ses-

Data Set	# Examples	# Features
spam	4600	58
cmc	1470	10
german	680	24

Table 1: Statistics of UCI datasets used for synthesized data

	Positive Class	Negative Class
# Instances	4343	3,6495
# Objects	55	186
# Avg of instances per object	69.3455	199.0538

Table 2: Statistics for the physiological data

sion-based learning with both a Decision Tree and AdaBoost using a Decision Tree as its base classifier.

Two types of data are used in experiments to evaluate the effectiveness of SBoost:

1) **Synthesized data** that are generated from binary UCI datasets [Blake and Merz,1998] by combining multiple instances into objects. Each object consists of ten different instances. To create an object for the positive class, a random number between 1 and 5 is first generated, and the corresponding number of instances from the negative class are randomly chosen and added to the object. The rest of the object is filled out with instances randomly selected from the positive class. A similar procedure is applied to generate objects for the negative class. By doing so, we guarantee that the class of an object is consistent with the dominant class assigned to its instances. The details of UCI datasets used in this experiment are listed in Table 1.

2) **Physiological data** that come from the workshop of physiological data modeling at the ICML 2004. We use the dataset for ‘watching TV’ with code 3004. In the original problem, the number of instances for the negative class is overwhelmingly larger than that for the positive class. Since we focus on the study of session-based learning, we intentionally reduce the effect of rare class by randomly selecting parts of negative instances. The resulting data set has 241 sessions with 40,838 instances. The details of this dataset are listed in Table 2.

A decision tree [Quinlan,1993] is used as the baseline classifier throughout the experiments. The session-based classification error, i.e., the percentage of sessions that are misclassified, is used for evaluation. 50% of data are randomly selected for training and the rest is used for testing. The same experiment is repeated 10 times, and the average session-based classification errors are reported. Finally, for both SBoost and AdaBoost, the maximum number of iterations is set to be 30.

4.1 Results

The results of three methods (SBoost, the naïve approach using a single decision tree, and with AdaBoost) are shown in Table 3. First, we compare the performance of AdaBoost with that of a decision tree as both adopting the naïve approach to session based learning. We observe that AdaBoost

Data Set	SBoost	AdaBoost	Decision Tree
spam	0.189 (0.022)	0.258 (0.059)	0.232 (0.051)
cmc	0.439 (0.044)	0.444 (0.070)	0.519 (0.052)
german	0.342 (0.028)	0.424 (0.069)	0.476 (0.081)
phys.	0.163 (0.038)	0.191 (0.035)	0.176(0.030)

Table 3: Classification errors with variances for the SBoost algorithm, the AdaBoost, and the decision tree

does not guarantee to improve performance over the baseline classifier (a decision tree here). In fact, for the spam dataset and the physiological data, the classification errors of AdaBoost are even greater than those of a decision tree. This is because the naïve approach treats all instances in an object as positive examples for the class of the object, while in reality, some instances in an object may belong to a class other than the class of the object. Hence, the naïve approach introduces noisy labels to training data, which will likely cause AdaBoost to overfit as observed in previous study of Boosting [Quanlin, 1996, Dietterich, 2000, Jin et al., 2003].

Second, comparing SBoost with the decision tree, we observe that SBoost always outperforms its baseline classifier. According to t-test, it is statistically significantly better than the decision over all the UCI datasets with $p < 0.05$. Thus, SBoost does not suffer from the problem of overfitting as AdaBoost does. To further investigate this issue, we obtain average classification errors of AdaBoost and SBoost in different iterations, respectively, as shown in Figure 4. Clearly, AdaBoost tends to overfit the training data after the first several iterations whereas SBoost does not.

5. Conclusion

In this paper, we formulate a new type of learning problem, session-based learning. It differs from the traditional supervised learning in that training examples are labeled objects and each object consists of multiple unlabeled instances. Furthermore, session-based learning adopts a different decision rule from that of multiple-instance learning: an object is classified as class ‘a’ when the majority of its instances are classified as ‘a’. This majority vote decision rule causes the

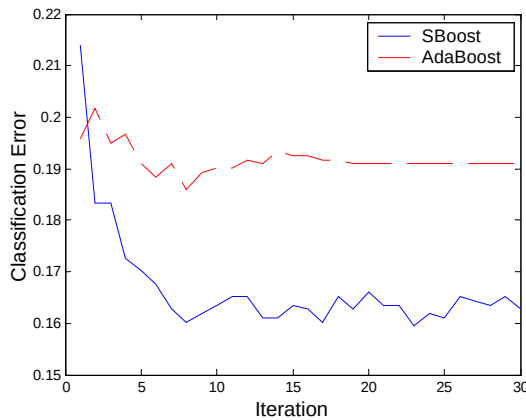


Figure 4: Classification errors for physiological data

label ambiguity problem and has made session-based learning very challenging. We formally describe this new learning problem and propose a novel boosting algorithm, named Sboost. Empirical studies have demonstrated the effectiveness of the Sboost algorithm.

References

- [Amar et.al. 2001] Robert A. Amar, Daniel R. Dooly, Sally A. Goldman, Qi Zhang, *Multiple-Instance Learning of Real-Valued Data*, Proc. 18th ICML, 2001.
- [Andrews and Hofmann, 2003] Stuart Andrews and Thomas Hofmann, *Multiple-Instance Learning via Disjunctive Programming Boosting*, NIPS, 2003.
- [Andrews, et al., 2002] Stuart Andrews, Thomas Hofmann and Ioannis Tsochantaridis, *Multiple Instance Learning with Generalized Support Vector Machines*, Proc. 18th AAAI , 2002.
- [Blake and Merz,1998] C.L. Blake and C.J. Merz, *UCI Repository of Machine Learning Databases*,1998.
- [Dietterich, et al.,1997] T. G. Dietterich, Richard H. Lathrop and Tomas Lozano-Perez, *Solving the Multiple-Instance Problem with Axis-Parallel Rectangles*, Artificial Intelligence, 89(1-2), 31-71,1997.
- [Dietterich, 2000] T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157,2000.
- [Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire, *A Decision-theoretic Generalization of On-line Learning and an Application to Boosting*, Journal of Computer and System Sciences, 55(1), 119-139, 1997.
- [Maron and Lozano-Pérez,1998] Oded Maron and Tomás Lozano-Pérez, *A Framework for Multiple-Instance Learning*, NIPS, 1998.
- [Jin et al. ,2003] Rong Jin, Yan Liu , Luo Si, Jaime Carbonell, Alexander G. Hauptmann, *A New Boosting Algorithm Using Input-Dependent Regularizer*, Proc. 20th ICML, 2003
- [Quinlan,1993] R.J. Quinlan,C4.5: *Programs for Machine Learning*, Morgan Kaufmann,1993.
- [Tao, et al.,2004] Qingping Tao, Stephen Scott, N. V. Vinodchandran and Thomas Takeo Osugi, *SVM-based Generalized Multiple-Instance Learning via Approximate Box Counting*, Proc. 21st ICML, 2004.
- [Wang and Zucker, 2000] Jun Wang, Jean-Daniel Zucker, *Solving the Multiple-Instance Problem: A Lazy Learning Approach*, Proc. 17th ICML, 2000.
- [Quinlan, 1996] Quinlan, J. R. (1996). *Bagging, Boosting, and C4.5*. Proceedings of the 13th National Conference on Artificial Intelligence, 322–330.
- [Reynolds, 1995] D. A. Reynolds, *Speaker identification and verification using Gaussian mixture speaker models*, Speech Communications, vol. 17, pp. 91–108, 1995.

A Novel Approach to Model Generation for Heterogeneous Data Classification

Rong Jin*, Huan Liu†

*Dept. of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824

rongjin@cse.msu.edu

† Department of Computer Science and Engineering, Arizona State University, Tempe, AZ85287-8809

hliu@asu.edu

Abstract

Ensemble methods such as bagging and boosting have been successfully applied to classification problems. Two important issues associated with an ensemble approach are: how to generate models to construct an ensemble, and how to combine them for classification. In this paper, we focus on the problem of model generation for *heterogeneous data classification*. If we could partition heterogeneous data into a number of homogeneous partitions, we will likely generate reliable and accurate classification models over the homogeneous partitions. We examine different ways of forming homogeneous subsets and propose a novel method that allows a data point to be assigned multiple times in order to generate homogeneous partitions for ensemble learning. We present the details of the new algorithm and empirical studies over the UCI benchmark datasets and datasets of image classification, and show that the proposed approach is effective for heterogeneous data classification.

1 Introduction

Ensemble approaches such as bagging and boosting have been successfully applied to many classification problems [Dietterich, 2000; Bauer and Kohavi, 1999]. The basic idea of ensemble methods is to construct a number of classifiers over training data and then classify new data points by taking a (weighted) vote of their predictions. Thus, two important issues associated with an ensemble approach are: 1) how to generate *accurate* yet *diverse* classification models, and 2) how to combine the models for ensemble classification. Diverse classifiers ensure good ensembles [Quinlan, 1996]. In this paper, we focus on the first issue with an emphasis on *heterogeneous data classification*. Heterogeneous data classification refers to the problem when input data of a single class are widely distributed into multiple modes. It arises when training data are collected under different environments or through different sources. An example of heterogeneous data classification is image classification, in which labeled images are acquired from multiple resources and exhibit disparate characteristics. For instance, some images are black and white, and others are colorful.

A widely used approach for constructing an ensemble of models is to sample different subsets from the training data and create a classification model for each subset. Bagging [Briemann, 1996] and AdaBoost [Schapire and Singer, 1999] are two representative methods in this category. Bagging randomly draws samples from the training data with replacement and AdaBoost samples training data according to a dynamically changed distribution, which is updated by putting more weight on the misclassified examples and smaller weights on the correctly classified examples. Clearly, both methods do not treat homogeneous data and heterogeneous data differently.

For ensemble methods to work effectively on heterogeneous data, one intuitive solution is to first divide the heterogeneous data into a set of homogeneous partitions and then to create a model for each partition of data. Member classifiers built with different homogeneous partitions will likely result in good diversity of an ensemble. One way to realize this homogeneity-based partition is to employ standard clustering algorithms, such as K-means [Hartigan and Wong, 1979] and the EM clustering algorithm [Celeux and Govaert, 1992]. An example is the Gaussian Mixture Model (GMM). But, in general, there are two problems with this simple clustering approach:

- *Single cluster membership*. Most clustering algorithms assume that cluster membership is mutually exclusive and each data point can only belong to a single cluster. Even though the EM clustering algorithm allows soft membership for a data point, in the resulting clusters, each data point still only belongs to a single cluster [Witten and Frank, 2000]. Therefore, when we use these clustering algorithms to partition data, if the number of clusters is large and the subsets of training data formed by a clustering algorithm are mutually disjoint, some clusters may have a very small number of data points, which can lead to unreliable classification models. This is similar to the data fragmentation problem occurred in decision tree induction [Quinlan, 1993]. In contrast, the subsets of training data produced by Bagging and AdaBoost are not mutually disjoint. For example, in bootstrap sampling, each subset contains around 63.2% of the original training data.

- *Unbalanced cluster sizes*. Since most clustering algorithms do not have control over cluster sizes, unbalanced cluster sizes resulting from clustering cannot be easily cor-

rected. When the resulting clusters have very different sizes, a classifier built over a small cluster can be unreliable and thus degrade the performance of the ensemble in forming final ensemble classification. On the contrary, both Bagging and AdaBoost have data samples of similar sizes when learning different models. Note that there have been previous efforts on balancing the sizes of different clusters, particularly for spectral clustering algorithms (e.g., the normalized cut algorithm [Melta & Shi, 2001]). But, since the control of cluster size comes indirectly from the objective function, the resulting clusters can still have unbalanced sizes.

In sum, a clustering approach may produce homogeneous data partitions, but cannot ensure similar sizes of different partitions; methods like Bagging can produce equally sized partitions, but partitions are not homogeneous. Therefore, we need a novel approach to partitioning data into homogeneous subsets of similar sizes in ensemble learning for heterogeneous data classification.

The goal of this work is to divide heterogeneous data into homogeneous subsets of similar sizes in order to generate reliable and accurate classification models. By focusing on homogeneous subsets, we do not require that each data point belong to one subset; by ensuring similar sizes of data subsets, each classification model can be built with a similar number of data points. In this paper, we propose a HISS (Homogeneous data In Similar Size) algorithm specially designed for the above purposes for heterogeneous data classification. Specifically, HISS allows the user to specify the size of a subset. For example, the user can ask the algorithm to create 20 subsets with each containing 40% of the original data. This algorithm is similar to the bootstrap sampling procedure in that both the number of subsets and the percentage of training data covered by each cluster can be specified and varied. However, it differs from the simple bootstrap sampling procedure in that it puts the similar data points into a single subset while bootstrap sampling randomly selects data to form a subset. This property is important in ensemble learning for classifying heterogeneous data. We will use *strata* for the *homogeneous data partitions*, and *subsets* for data partitions resulting from random sampling.

2 Related Work

There have been many previous studies on how to create an ensemble of models. The methods for constructing an ensemble of models can be categorized into five groups [Dietterich, 2000]: 1) *Bayesian methods*, which creates an ensemble of model by sampling them from a estimated posterior model distribution; 2) *Sampling training examples*, which creates multiple subsets of training examples and trains a classifier for each of the subsets; 3) *Sampling input features*, which creates a number of subsets of the input features and a classifier is built for each subset of input features; 4) *Error correct output code (ECOC)*, which convert a multiple class problem into a set of binary class problems; 5) *Injecting randomness*, that generates ensembles of classifiers by injecting randomness into the learning algorithm.

Among the five categories, our work is closely related to the second one, which creates multiple classifiers by sam-

pling training examples. Important methods in this group include Bagging [Brieman, 1996] and AdaBoost [Schapire and Singer, 1999]. Although these methods have been shown to be effective for classification, they are not designed to take into account characteristics of heterogeneous data. In this paper, we propose HISS –an algorithm that constructs homogeneous strata from heterogeneous data while maintains the nice property of bootstrap sampling procedure - each stratum contains a similar number of data points.

Another line of research closely related to this work is the study of clustering algorithms. In general, clustering algorithms can be categorized into parametric approaches and non-parametric approaches. The parametric approach is to find a parametric model that minimizes a cost function associated with instance-cluster assignments. Such methods include the Mixture Model [Celeux and Govaert, 1992] and K-means algorithm. For the non-parametric approaches, a cost function is minimized by either merging two separate clusters into a larger one or dividing a cluster into two smaller ones. The representative examples of this category are the agglomerative approach and the divisive approach.

Most clustering approaches assume that each data point only belongs to a single cluster. This assumption may not be appropriate since the ultimate goal of clustering is to group similar data points together. When it is uncertain to assign a data point to a single cluster, it is better off assigning it to multiple clusters. Although the traditional probabilistic model and the fuzzy clustering algorithm allow for multi- or soft-memberships, the uncertainty of cluster membership is only exploited during the process of estimation. In the resulting clusters, each data point is assigned to only a single cluster. Furthermore, most clustering algorithms do not have any control over the size of clusters. Hence, the resulting clusters can be very unbalanced in size and the clusters of too small sizes could be useless in learning.

3 The HISS Algorithm for Model Generation

3.1 From Probabilistic Clustering to HISS

We first describe the traditional probabilistic clustering algorithm, and then introduce algorithm HISS.

The general idea of probabilistic clustering is to describe data with a mixture of generative models. Optimal parameters are usually obtained by maximizing the likelihood of data using the mixture model. Let n be the number of input data points, K be the number clusters, $\{x_1, x_2, \dots, x_n\}$ be the input data, and $\{m_1, m_2, \dots, m_K\}$ be the underlying models that generate the data. By assuming that each data point is generated from a mixture of models $\{m_1, m_2, \dots, m_K\}$, we have the likelihood of the data written as:

$$l(\{m_i\}_{i=1}^K, \tau) = \sum_{i=1}^n \log \left(\sum_{j=1}^K \tau_i^j p(x_i | m_j) \right) \quad (1)$$

where $p(x_i | m_j)$ is the likelihood of generating x_i from the model m_j , and τ_i^j is the likelihood for data point x_i to be in the j -th cluster. Based on the assumption that each data point can only belong to a single cluster, we have constraint $\sum_{j=1}^K \tau_i^j = 1$. An example of probabilistic clustering is the Gaussian Mixture Model (GMM), in which both τ_i^j and $p(x_i | m_j)$ are parameterized as:

$$\tau_i^j = \theta_j, \text{ and} \quad p(x_i | m_j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)}{2}\right) \quad (2)$$

where θ_j denotes the prior for the j -th cluster, and μ_j and Σ_j are the mean and variance matrix for the j -th cluster, respectively. Expectation and Maximization algorithm (EM) (Dempster et al, 1977) can be used to search for the optimal parameters.

By removing the constraint $\sum_{j=1}^K \tau_i^j = 1$, we allow each data point to belong to multiple homogeneous clusters, or in short, strata. Hence, the optimization problem becomes

$$\max_{m_j, \tau_i^j} l(\{m_j\}_{j=1}^K, \tau) = \sum_{i=1}^n \log \left(\sum_{j=1}^K \tau_i^j p(x_i | m_j) \right) \quad (3)$$

subject to

$$0 \leq \tau_i^j \leq 1 \text{ for } i=1, \dots, n, \text{ and } j=1, \dots, K$$

where all τ_i^j are constrained to between 0 and 1 to maintain the probability interpretation. It is easy to see that the optimal solution is to set all τ_i^j to be 1, which means that each data point is included in every stratum.

To avoid the trivial solution for τ_i^j , we choose to enforce the percentage of training data that are covered by each cluster to be a predefined constant γ , i.e.,

$$\frac{1}{n} \sum_{i=1}^n \tau_i^j = \gamma, \quad 0 \leq \gamma \leq 1, \text{ for } j=1, \dots, K \quad (4)$$

With the above constraint, we guarantee that the number of data points that support each stratum is around γn .

Compared to the single membership constraint, this new constraint has the following two advantages: 1) **It does not assume that each data point has to belong to one stratum.** For this new stratifying method, on average each data point can belong to γK number of strata. Therefore, when γK is larger than one, each data point is allowed to be in more than one stratum simultaneously. 2) **It ensures that different strata have balanced numbers of data points.** In contrast to most clustering algorithms, the new algorithm ensures almost the same size for each stratum. This is particularly important to the research goal of this paper - generat-

ing a reliable and accurate ensemble for heterogeneous data. By setting γ to be a reasonably large value (0.4 in this work), we ensure that each stratum has a sufficiently large number of examples for building a statistical learning model. For later reference, we refer this new clustering approach as “HISS”, which stands for Homogeneous data In Similar Size.

3.2 Optimization for HISS

Putting Equations (3) and (4) together, we have:

$$\max_{m_j, \tau_i^j} l(\{m_j\}_{j=1}^K, \tau) = \sum_{i=1}^n \log \left(\sum_{j=1}^K \tau_i^j p(x_i | m_j) \right) \quad (5)$$

subject to

$$\frac{1}{n} \sum_{i=1}^n \tau_i^j = \gamma, \quad 0 \leq \gamma \leq 1, \text{ for } j=1, \dots, K$$

$$0 \leq \tau_i^j \leq 1 \text{ for } i=1, \dots, n, \text{ and } j=1, \dots, K$$

Let us assume the Gaussian distribution for $p(x_i | m_j)$, i.e., $p(x | m_j) \sim N(\mu_j, \sigma_j)$. Following the idea of the EM algorithm, the difference in the likelihood of data between two consecutive iterations is bound by:

$$l(\{m_j(t+1)\}_{j=1}^K, \tau(t+1)) - l(\{m_j(t)\}_{j=1}^K, \tau(t)) \geq \sum_{i=1}^n \sum_{j=1}^K \nu_i^j(t) \log \left[\frac{\tau_i^j(t+1)}{\tau_i^j(t)} \right] + \sum_{i=1}^n \sum_{j=1}^K \nu_i^j(t) \log \left[\frac{p(x_i | m_j(t+1))}{p(x_i | m_j(t))} \right] \quad (6)$$

where ν_i^j is defined as

$$\nu_i^j(t) = \frac{\tau_i^j(t) p(x_i | m_j(t))}{\sum_{j=1}^K \tau_i^j(t) p(x_i | m_j(t))} \quad (7)$$

Thus, the optimal solutions for the mean and variance of Gaussian distribution can be obtained as follows:

$$\mu_j(t+1) = \frac{\sum_{i=1}^n \nu_i^j(t) x_i}{\sum_{i=1}^n \nu_i^j(t)}, \quad \sigma_j^2(t+1) = \frac{\sum_{i=1}^n \nu_i^j(t) x_i^2}{\sum_{i=1}^n \nu_i^j(t)} - \mu_j^2(t+1)$$

However, the optimal solution for τ_i^j is rather difficult to obtain because of the inequality constraints $0 \leq \tau_i^j \leq 1$. Directly optimizing the Equation (6) with only the equality constraint will result in the following solution for $\tau_i^j(t+1)$:

$$\tau_i^j(t+1) = \frac{\nu_i^j(t) \gamma n}{\sum_{i=1}^n \nu_i^j(t)} \quad (8)$$

Apparently, the above solution will always be nonnegative if $\tau_i^j(t)$ is nonnegative. However, it does not guarantee that $\tau_i^j(t+1)$ is not greater than 1.

Finding Optimal $\tau_i^j(t+1)$

Inputs: $v_i^j(t)$ for $i=1, \dots, n$ and $j=1, \dots, K$

Outputs: $\tau_i^j(t+1)$ that maximizes Equation (7).

Initialization:

$$\tau_i^j(t+1) = 0 \text{ for } i=1, \dots, n \text{ and } j=1, \dots, K$$

for each cluster j

do

For all examples i ,

$$\text{set } \tau_i^j(t+1) = 1 \text{ if } \tau_i^j(t+1) > 1$$

Compute the probability mass

$$s = \gamma n - \left| \{i \mid \tau_i^j(t+1) = 1\} \right|$$

Re-compute

$$\tau_i^j(t+1) = s \frac{v_i^j(t)}{\sum_{\{i \mid \tau_i^j(t+1) < 1\}} v_i^j(t)} \quad \forall j \text{ s.t. } \tau_i^j(t+1) < 1$$

while ($\exists j$ s.t. $\tau_i^j(t+1) > 1$)

end

Figure 1: Algorithm for finding optimal $\tau_i^j(t+1)$

In order to satisfy the inequality constraints $0 \leq \tau_i^j \leq 1$, we use the KKT conditions [Fletcher, 1987] to efficiently adjust the value of $\tau_i^j(t+1)$. The basic idea is to reset τ_i^j to be 1 whenever the output from Equation (10) violates the constraint $0 \leq \tau_i^j \leq 1$. After the adjustment, we will re-compute $\tau_i^j(t+1)$ that are less than 1 using Equation (8). The procedure of adjusting and recomputing $\tau_i^j(t+1)$ will continue until no $\tau_i^j(t+1)$ violates the constraint. Figure 1 shows the detailed steps for finding the optimal solution for $\tau_i^j(t+1)$. Due to the space limit, the proof for the optimality of the algorithm in Figure 1 is not provided here.

3.3 Classifying Heterogeneous Data

For classification problems, heterogeneous data can be found in many applications and in experiments:

- 1) *Data acquired from multiple sources.* In many cases, training data are acquired from multiple sources. Because each source has its own data distribution that may be different from others, the data merged from multiple sources are therefore heterogeneous. For example, consider building a classification model for outdoor scenes. The training images are collected from several different types of videos. Some of the videos are news stories and some of them are of advertisement. Some of them are of high quality and some of them are not. Thus, the widely disparate characteristics in videos cause the merged data to be heterogeneous.
- 2) *Data by converting a multiple class problem into a set of binary class problems.* In order to apply the binary class classification algorithm to multiple class case, we need to

Data Set	# Examples	#Class	# Features
Ecoli	327	5	7
Pendigit	2000	10	16
Glass	204	5	10
Yeast	1479	10	8
Vehicle	946	4	17
Image/Indoor	3500	2	190
Image/Outdoor	3500	2	126

Table 1: Description of datasets for the experiment for heterogeneous data classification.

convert the classification problem of multiple classes into a set of binary class problems. The representative examples include the one-against-all approach and error correct output coding (ECOC) method [Dietterich, 1995]. During this process, multiple classes are grouped into two subsets of classes. Data points from one subset of classes are used as positive examples and the remaining are used as negative examples. Because both the positive and negative pools can be comprised of examples from multiple classes, it will create data heterogeneity for each of the binary classes.

As discussed, an intuitive solution to classifying heterogeneous data is to create a set of classification models with each classifier built on a homogeneous partition (stratum) of the data, and then combine classifiers for the final prediction. The traditional clustering algorithms are not designed for this task because of the potential unbalanced cluster-sizes and the data fragmentation problem. With the proposed algorithm HISS, we can avoid these two problems by setting the parameter to be large (0.4 in the experiment).

In sum, to classify heterogeneous data, we first apply HISS to obtain homogeneous strata and then create a classification model for each stratum to form an ensemble. We will refer to this model generation method as ‘HISS-based Model Generation’ in our empirical study next. Finally, a stacking approach [Wolpert, 1992] is used to combine models that are generated by the HISS-based model generation method for the final prediction of the ensemble.

4. Experimental Study

The experimental study is designed to answer the following questions:

- 1) *Is the proposed model generation method effective for classifying heterogeneous data?* To this end, we compare the proposed model generation method to Bagging and AdaBoost in classifying heterogeneous datasets.
- 2) *Is the proposed HISS algorithm effective for generating reliable models?* To address this question, we will apply both the proposed HISS algorithm and the probabilistic clustering algorithm to partition the training data and build a classification model for each partition.

4.1 Experimental Design

Seven different datasets are used in the experiments: five multiple class datasets from the UCI Machine Learning repository [Blake and Merz, 1998] and two binary class data-

Data Set	Baseline	AdaBoost (Standard)	Bagging (Standard)	HISS-based Ensemble	Bagging (Stacking)	AdaBoost (Stacking)
Ecoli	0.047 (0.012)	0.046 (0.006)	0.057 (0.014)	0.037 (0.006)	0.046 (0.006)	0.059 (0.006)
Pendigit	0.010 (0.003)	0.013 (0.003)	0.012 (0.002)	0.008 (0.002)	0.012 (0.001)	0.012 (0.003)
Glass	0.382 (0.027)	0.385 (0.081)	0.379 (0.046)	0.161 (0.044)	0.379 (0.027)	0.379 (0.027)
Yeast	0.314 (0.012)	0.320 (0.023)	0.313 (0.013)	0.313 (0.013)	0.315 (0.012)	0.315 (0.008)
Vehicle	0.103 (0.020)	0.163 (0.048)	0.131 (0.024)	0.048 (0.012)	0.100 (0.017)	0.085 (0.033)
Image/Indoor	0.153 (0.008)	0.140 (0.007)	0.156 (0.014)	0.140 (0.013)	0.157 (0.011)	0.144 (0.007)
Image/Outdoor	0.116 (0.008)	0.111 (0.017)	0.120 (0.011)	0.088 (0.005)	0.114 (0.006)	0.112 (0.007)

Table 2: Classification errors for the baseline model (SVM), AdaBoost, Bagging and the propose model generation method (‘HISS-based Ensemble’). The column ‘Bagging (Stacking)’ refers to the case when the ensemble of models is created by the Bagging algorithm but combined through the stacking approach using an SVM. The same is for the column ‘AdaBoost (Stacking)’. The variance of classification error is listed in parenthesis.

sets for image classification. The characteristics of these seven datasets are listed in Table 1.

For the multiple class datasets, we introduce the heterogeneity into the data by converting the original multiple-class problem into a binary one. Similar to the one-against-all approach, examples from the most popular class are used as the positive instances and examples from the remaining classes are assigned to the negative class. Because data of the negative class are from multiple classes, we would expect some degree of heterogeneity inside the negative class. For the two datasets of image classification, they both are binary classification problems. The heterogeneity of data is due to the fact that images are from seven different video clips and each video clip provides 500 images. Since each video clip is of different type (e.g., varied quality in images), we would expect certain amount of heterogeneity within the data.

The baseline algorithm used in this experiment is support vector machine [Burger, 1998]. In all the experiments, each ensemble method generates 20 different SVMs; a stacking approach [Wolpert, 1992] that also uses a SVM is employed to combine the outputs from all 20 models to form the final prediction of the ensemble. For each experiment, we randomly select 70% of the data as training and the remaining 30% as testing. The experiment is repeated 10 times and the average classification error of the ten runs is used as the final result with the variance of classification errors.

4.2 Heterogeneous Data Classification

Table 2 shows classification errors for the baseline support

Data Set	HISS	EM (3 Clusters)	EM (10 Clusters)
Ecoli	0.037(0.006)	0.448 (0.021)	0.448 (0.021)
Pendigit	0.008(0.002)	0.081 (0.043)	0.110 (0.023)
Glass	0.161(0.044)	0.292 (0.101)	0.353 (0.017)
Yeast	0.313 (0.013)	0.314 (0.013)	0.314 (0.019)
Vehicle	0.048 (0.012)	0.219 (0.068)	0.052 (0.026)
Image/Indoor	0.140(0.013)	0.184 (0.022)	0.203 (0.014)
Image/Outdoor	0.088(0.005)	0.156 (0.031)	0.182 (0.036)

Table 3: Classification error for using different clustering algorithms for model generation. ‘EM’ refers to using Expectation-Maximization algorithm to cluster data.

vector machine, the proposed HISS-based ensemble learning approach, standard Bagging and standard AdaBoost. First, we can see that the baseline model performs well comparing with both standard Bagging and AdaBoost. This observation indicates that these seven heterogeneous datasets are rather difficult for the standard ensemble approaches to learn. In contrast, the proposed HISS-based ensemble method performs better than the baseline model and the two standard ensemble methods. For the datasets ‘Glass’, ‘Vehicle’, and ‘Image/Outdoor’, the improvement is substantial, from 38.2% to 16.1% for ‘Glass’, 10.3% to 4.8% for ‘Vehicle’, and from 11.6% to 8.8% for ‘Image/Outdoor’.

Since the HISS-based ensemble method uses the stacking approach for combining different models, it is different from the combination method that is used by AdaBoost and Bagging. To address this difference, we conduct the experiments that apply a stacking method to combine the models generated by both Bagging and AdaBoost. The results are listed in Table 2 on the right side of the HISS-based approach, titled as ‘Bagging (Stacking)’ and ‘AdaBoost (Stacking)’, respectively. Compared these results to the results of ‘Bagging (Standard)’ and ‘AdaBoost (Standard)’, we see that there is no substantial change in classification errors when using a stacking approach to combine models in ensemble learning. For all the seven datasets, the ensemble of models generated by HISS performs the best. The reason why a stacking approach is useful for the HISS-based model generation method but not to the other two is that models generated by the HISS-based algorithm are much more diverse than the ones generated by both Bagging and AdaBoost. As a result, applying another layer of classification model to combine the outputs from the distinguishable models (or stacking) will be able to take full advantage of all the models and obtain the best performance.

Based on the above discussion, we conclude that the HISS-based ensemble model is more effective for classifying heterogeneous data than existing ensemble approaches.

4.3 Comparison with Other Clustering-based Ensemble Methods

The advantage of HISS versus the traditional clustering algorithms is that HISS allows each data point to be in multiple different strata. Thus it can ensure that the number of

data points distributed over each stratum is of similar size and sufficiently large.

In this experiment, we use both the traditional clustering algorithm and the proposed HISS algorithm for model generation and see how different they are in classifying the heterogeneous datasets. To observe the effect due to the trade-off between the number of strata and the number of data points in each stratum, we consider two different numbers of strata (or clusters) for the traditional clustering algorithm: 10 and $3(\approx 1/\gamma)$. We did not use 20 clusters in the comparison because for some datasets the traditional clustering algorithm is unable to produce the full twenty clusters. The traditional clustering algorithm used in the experiment is the probabilistic EM clustering algorithm. Similar to the HISS-based ensemble approach, a stacking method is used to combine models generated by the EM clustering algorithm. The results for using EM clustering algorithms for model construction are listed in Table 3, titled ‘EM (3 clusters)’ and ‘EM (10 clusters)’. As suggested by Table 3, the increasing number of clusters can lead to degraded performance. This is because a large number of clusters will form clusters with a small number of data points, which can be insufficient for building a reliable classification model. On the other hand, as already indicated in the previous study [Dietterich, 2000], being able to generate a relatively large number of models is critical to the success of the ensemble approach. The proposed HISS algorithm can satisfy both needs by introducing the substantial overlapping between different clusters. As shown in Table 3, the HISS-based method outperforms the EM_clustering-based ensemble approaches substantially for almost all datasets except for ‘Yeast’ (similar). The most noticeable cases are ‘Ecoli’ and ‘Pendigit’, for which the classification errors of EM-based clustering approaches are one order more than that of the HISS-based ensemble algorithm.

Based on the above experiments and analysis, we conclude that the HISS-based model generation is an effective method for model generation in ensemble learning for heterogeneous data classification.

5. Conclusion and Future Work

In this paper, we propose and examine a new method for generating an ensemble of models, which is to first partition data into homogeneous subsets and then create a model for each subset. A traditional clustering algorithm like EM is not suitable for the task of partitioning data due to potential size-unbalanced clusters and the data fragmentation problem. To address these two problems, we propose a novel algorithm HISS, which allows for data overlapping between different clusters (strata) and promises size-balanced clusters. Empirical studies over seven different heterogeneous datasets have shown that this new HISS-based model generation method performs very well for heterogeneous data classification. Currently, the proposed HISS algorithm assumes equal size for each stratum (cluster). One possible extension is to examine alternatives to balance sizes of clusters. For example, instead of enforcing all the clusters to have one size, we can constrain the sizes of the clusters into

a specified range to allow some flexibility in maintaining high homogeneity of clusters.

References

- [Hartigan and Wong, 1979] Hartigan, J.A. and Wong, M.A., A K-means Clustering Algorithm, *Applied Statistics* 28: 100-108
- [Briemann, 1996] Briemann, L., Bagging Predictor, *Machine Learning* 26, 123-140, 1996
- [Schapire and Singer, 1999] Schapire, R.E. and Singer, Y., Improved boosting algorithms using confidence-rated predictions, *Machine Learning* 37 (3): 291-336, 1999
- [Celeux and Govaert, 1992] Celeux, G. and Govaert, G., A Classification EM Algorithm for Clustering and Two Stochastic Versions, *Computational Statistics & Data Analysis*, vol. 14, pp. 315-332, 1992
- [Dietterich, 2000] Dietterich, T.G., Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, Cagliari, Italy, 2000
- [Dempster et al., 1977] Dempster, A.P., Laird, N.M., and Rubin, D.B., Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal statistical Society, Series B*, 39(1): 1-38, 1977
- [Fletcher, 1987] Fletcher, R., *Practical Methods of Optimization*. John Wiley and Sons, Inc., 2nd edition, 1987.
- [Dietterich and Bakiri, 1995] Dietterich, T.G. and Bakiri, G., Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, (2):263-286, 1995.
- [Wolpert, 1992] Wolpert, D.H., Stacked Generalization. *Neural Networks*, 5:241-259, Pergamon Press, 1992.
- [Burges, 1998] Burges, C.J.C., A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):955-974, 1998.
- [Witten and Frank, 2000] Witten, I.H. and Frank, E., *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, 2000.
- [Blake and Merz, 1998] Blake, C. and Merz, C., UCI repository of machine learning databases. <http://www.ics.uci.edu/mllearn/MLRepository.html>.
- [Quinlan, 1993] Quinlan, R.J., *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.
- [Quinlan, 1996] Quinlan R.J., Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 96)*, 1996.
- [Shi and Malik, 2000] Shi, J., and Malik, J., Normalized Cut and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000
- [Bauer and Kohavi, 1999] Bauer, E. and Kohavi, R., An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36(1):105-139, 1999.

State Abstraction Discovery from Irrelevant State Variables

Nicholas K. Jong

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
nkj@cs.utexas.edu

Peter Stone

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
pstone@cs.utexas.edu

Abstract

Abstraction is a powerful form of domain knowledge that allows reinforcement-learning agents to cope with complex environments, but in most cases a human must supply this knowledge. In the absence of such prior knowledge or a given model, we propose an algorithm for the automatic discovery of state abstraction from policies learned in one domain for use in other domains that have similar structure. To this end, we introduce a novel condition for state abstraction in terms of the relevance of state features to optimal behavior, and we exhibit statistical methods that detect this condition robustly. Finally, we show how to apply temporal abstraction to benefit safely from even partial state abstraction in the presence of generalization error.

1 Introduction

Humans can cope with an unfathomably complex world due to their ability to focus on pertinent information while ignoring irrelevant detail. In contrast, most of the research into artificial intelligence relies on fixed problem representations. Typically, the researcher must engineer a feature space rich enough to allow the algorithm to find a solution but small enough to achieve reasonable efficiency. In this paper we consider the *reinforcement learning* (RL) problem, in which an agent must learn to maximize rewards in an initially unknown, stochastic environment [Sutton and Barto, 1998]. The agent must consider enough aspects of each situation to inform its choices without spending resources worrying about minutiae. In practice, the complexity of this state representation is a key factor limiting the application of standard RL algorithms to real-world problems.

One approach to adjusting problem representation is *state abstraction*, which maps two distinct states in the original formulation to a single abstract state if an agent should treat the two states in exactly the same way. The agent can still learn optimal behavior if the Markov decision process (MDP) that formalizes the underlying domain obeys certain conditions: the relevant states must share the same local behavior in the abstract state space [Dean and Givan, 1997; Ravindran and Barto, 2003]. However, this prior research only applies in a planning context, in which the MDP model

is given, or if the user manually determines that the conditions hold and supplies the corresponding state abstraction to the RL algorithm.

We propose an alternative basis to state abstraction that is more conducive to automatic discovery. Intuitively, if it is possible to behave optimally while ignoring a certain aspect of the state representation, then an agent has reason to ignore that aspect during learning. Recognizing that discovering structure tends to be slower than learning an optimal behavior policy [Thrun and Schwartz, 1995], this approach suggests a knowledge-transfer framework, in which we analyze policies learned in one domain to discover abstractions that might improve learning in similar domains. To test whether abstraction is possible in a given region of the state space, we give two statistical methods that trade off computational and sample complexity.

We must take care when we apply our discovered abstractions, since the criteria we use in discovery are strictly weaker than those given in prior work on safe state abstraction. Transferring abstractions from one domain to another may also introduce generalization error. To preserve convergence to an optimal policy, we encapsulate our state abstractions in *temporal abstractions*, which construe sequences of primitive actions as constituting a single abstract action [Sutton *et al.*, 1999]. In contrast to previous work with temporal abstraction, we discover abstract actions intended just to simplify the state representation, not to achieve a certain goal state. RL agents equipped with these abstract actions thus learn when to apply state abstraction the same way they learn when to execute any other action.

In Section 2, we describe our first contribution, an alternative condition for state abstraction and statistical mechanisms for discovery. In Section 3, we describe our second contribution, an approach to discovering state abstractions and then encapsulating them within temporal abstractions. In Section 4, we present an empirical validation of our approach. In Section 5, we discuss related work, and in Section 6, we conclude.

2 Policy irrelevance

2.1 Defining irrelevance

First we recapitulate the standard MDP notation. An MDP $\langle S, A, P, R \rangle$ comprises a finite set of states S , a finite set

of actions A , a transition function $P : S \times A \times S \rightarrow [0, 1]$, and a reward function $R : S \times A \rightarrow \mathbb{R}$. Executing an action a in a state s yields an expected immediate reward $R(s, a)$ and causes a transition to state s' with probability $P(s, a, s')$. A policy $\pi : S \rightarrow A$ specifies an action $\pi(s)$ for every state s and induces a value function $V^\pi : S \rightarrow \mathbb{R}$ that satisfies the Bellman equations $V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s, \pi(s), s') V^\pi(s')$, where $\gamma \in [0, 1]$ is a discount factor for future reward that may be necessary to make the equations satisfiable. For every MDP at least one optimal policy π^* exists that maximizes the value function at every state simultaneously. We denote the unique optimal value function V^* . Many learning algorithms converge to optimal policies by estimating the optimal state-action value function $Q^* : S \times A \rightarrow \mathbb{R}$, with $Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V^*(s')$.

Without loss of generality, assume that the state space is the cartesian product of (the domains of) n state variables $\mathcal{X} = \{X_1, \dots, X_n\}$ and m state variables $\mathcal{Y} = \{Y_1, \dots, Y_m\}$, so $S = X_1 \times \dots \times X_n \times Y_1 \times \dots \times Y_m$. We write $[s]_{\mathcal{X}}$ to denote the projection of s onto $\mathcal{X}$ and $s' \models [s]_{\mathcal{X}}$ to denote that s' agrees with s on every state variable in $\mathcal{X}$. Our goal is to determine when we can safely abstract away $\mathcal{Y}$. In this work we introduce a novel approach to state abstraction called *policy irrelevance*. Intuitively, if an agent can behave optimally while ignoring a state variable, then we should abstract that state variable away. More formally, we say that $\mathcal{Y}$ is policy irrelevant at s if some optimal policy specifies the same action for every s' such that $s' \models [s]_{\mathcal{X}}$:

$$\exists a \forall s' \models [s]_{\mathcal{X}} \forall a' Q^*(s', a) \geq Q^*(s', a'). \quad (1)$$

If $\mathcal{Y}$ is policy irrelevant for every s , then $\mathcal{Y}$ is policy irrelevant for the entire domain.

Consider the illustrative toy domain shown in Figure 1. It has just four nonterminal states described by two state variables, X and Y . It has two deterministic actions, represented by the solid and dashed arrows respectively. When $X = 1$, both actions terminate the episode but determine the final reward, as indicated in the figure. This domain has two optimal policies, one of which we can express without Y : take the solid arrow when $X = 0$ and the dashed arrow when $X = 1$. We thus say that Y is policy irrelevant across the entire domain.

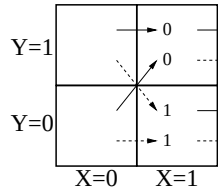


Figure 1: A domain with four nonterminal states and two actions. When $X = 1$ both actions transition to an absorbing state, not shown.

Note however that we cannot simply aggregate the four states into two states. As McCallum pointed out, the state distinctions sufficient to represent the optimal policy are not necessarily sufficient to learn the optimal policy [McCallum, 1995]. In this example, observe that if we treat $X = 1$ as a single abstract state, then in $X = 0$ we will learn to take the dashed arrow, since it transitions to the same abstract state as the solid arrow but earns a greater immediate reward. We demonstrate how to circumvent this problem while still benefiting from the abstraction in Section 3.2.

2.2 Testing irrelevance

If we have access to the transition and reward functions, we can evaluate the policy irrelevance of a candidate set of state variables $\mathcal{Y}$ by solving the MDP using a method, such as policy iteration, that can yield the set of optimal actions $\pi^*(s) \subseteq A$ at each state s . Then $\mathcal{Y}$ is policy irrelevant at s if some action is in each of these sets for each assignment to $\mathcal{Y}$: $\bigcap_{s' \models [s]_{\mathcal{X}}} \pi^*(s') \neq \emptyset$.

However, testing policy irrelevance in an RL context is trickier if the domain has more than one optimal policy, which is often the case for domains that contain structure or symmetry. Most current RL algorithms focus on finding a single optimal action at each state, not all the optimal actions. For example, Figure 2 shows the Q values learned from

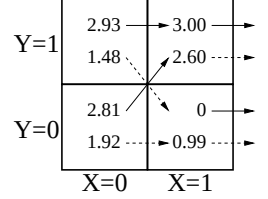


Figure 2: The domain of Figure 1 with some learned Q values.

a run of Q-learning,¹ a standard algorithm that employs stochastic approximation to learn Q^* [Watkins, 1989]. Even though the state variable Y is actually policy irrelevant, from this data we would conclude that an agent must know the value of Y to behave optimally when $X = 1$. In this trial we allowed the learning algorithm enough exploration to find an optimal policy but not enough to converge to accurate Q values for every state-action pair. We argue that this phenomenon is quite common in practical applications, but even with sufficient exploration the inherent stochasticity of the domain may disguise state variable irrelevance. We propose two methods for detecting policy irrelevance in a manner robust to this variability.

Statistical hypothesis testing

Hypothesis testing is a method for drawing inferences about the true distributions underlying sample data. In this section, we describe how to apply this method to the problem of inferring policy irrelevance. To this end, we interpret an RL algorithm's learned value $Q(s, a)$ as a random variable, whose distribution depends on both the learning algorithm and the domain. Ideally, we could then directly test that hypothesis (1) holds, but we lack an appropriate test statistic. Instead, we assume that for a reasonable RL algorithm, the means of these distributions share the same relationships as the corresponding true Q values: $\overline{Q}(s, a) \geq \overline{Q}(s, a') \equiv Q^*(s, a) \geq Q^*(s, a')$. We then test propositions of the form

$$\overline{Q}(s, a) \geq \overline{Q}(s, a'), \quad (2)$$

using a standard procedure such as a one-sided paired t -test or Wilcoxon signed ranks test [Degroot, 1986]. These tests output for each hypothesis (2) a significance level $p_{s,a,a'}$. If $\overline{Q}(s, a) = \overline{Q}(s, a')$ then this value is a uniformly random number from the interval $(0, 1)$. Otherwise, $p_{s,a,a'}$ will tend towards 1 if hypothesis (2) is true and towards 0 if it is false. We combine these values in a straightforward way to obtain a confidence measure for hypothesis (1):

$$p = \max_a \min_{s' \models [s]_{\mathcal{X}}} \min_{a' \neq a} p_{s',a,a'}. \quad (3)$$

¹No discounting, learning rate 0.25, Boltzmann exploration with starting temperature 50, cooling rate 0.95, for 50 episodes

Figure 3 shows these p values for our toy domain. To obtain the data necessary to run the test, we ran 25 independent trials of Q-learning. We used the Wilcoxon signed-ranks test, which unlike the t -test does not assume that $Q(s, a)$ is Gaussian. In Figure 3a we see “random” looking values, so we accept that Y is policy irrelevant for both values of X . In Figure 3b we see values very close to 0, so we must reject our hypothesis that X is policy irrelevant for either value of Y . In our work, we use 0.05 as a threshold for rejecting hypothesis (1). If p exceeds 0.05 for every s , then $\mathcal{Y}$ is irrelevant across the entire domain. In practice this number seems quite conservative, since in those cases when the hypothesis is false we consistently see p values orders of magnitude smaller.

		Y=1	0.001
		Y=0	0.000
0.367	0.731		
X=0	X=1		

(a)

(b)

Figure 3: The value of p for each of the two abstract states when testing the policy irrelevance of (a) Y and (b) X .

Monte Carlo simulation

The hypothesis testing approach is computationally efficient, but it requires a large amount of data. We explored an alternative approach designed to conserve experience data when interaction with the domain is expensive. We draw upon work in Bayesian MDP models [Dearden *et al.*, 1999] to reason more directly about the distribution of each $Q(s, a)$. This technique regards the successor state for a given state-action pair as a random variable with an unknown multinomial distribution. For each multinomial distribution, we perform Bayesian estimation, which maintains a probability distribution over multinomial parameters. After conditioning on state transition data from a run of an arbitrary RL algorithm, the joint distribution over the parameters of these multinomials gives us a distribution over transition functions. The variance of this distribution goes to 0 and its mean converges on the true transition function as the amount of data increases.²

Once we have a Bayesian model of the domain, we can apply Monte Carlo simulation to make probabilistic statements about the Q values. We sample MDPs from the model and solve them³ to obtain a sample for each Q value. Then we can estimate the probability that $Q^*(s, a) \geq Q^*(s, a')$ holds as the fraction of the sample for which it holds. We use this probability estimate in the same way that we used the significance levels in the hypothesis testing approach to obtain a confidence measure for the policy irrelevance of $\mathcal{Y}$ at some s :

$$p = \max_a \min_{s' \models [s]_{\mathcal{X}}} \min_{a' \neq a} \Pr(Q^*(s', a) \geq Q^*(s', a')). \quad (4)$$

This method seems to yield qualitatively similar results to the hypothesis testing method. We almost always obtain a

²It is also possible to build a Bayesian model of the reward function, but all the domains that we have studied use deterministic rewards.

³We use standard value iteration.

value of $p = 0$ for cases in which $\mathcal{Y}$ actually is relevant; we obtain a value near 1 when only one action is optimal; we obtain a uniformly random number in $(0, 1)$ when more than one action is optimal. Although it achieves similar results using less data, this method incurs a higher computational cost due to the need to solve multiple MDPs.⁴

3 Abstraction discovery

3.1 Discovering irrelevance

The techniques described in Section 2.2 both involve two stages of computation. In the first stage, they acquire samples of state-action values, either by solving the task repeatedly or by solving sampled MDPs repeatedly. In the second stage, they use this data to test the relevance of arbitrary sets of state variables at arbitrary states. Any one of these tests in the second stage is very cheap relative to the cost of the first stage, but the number of possible tests is astronomical. We must limit both the sets of state variables that we test and the states at which we test them.

First consider the sets of state variables. It is straightforward to prove that if $\mathcal{Y}$ is policy irrelevant at s , then every subset of $\mathcal{Y}$ is also policy irrelevant at s .⁵ A corollary is that we only need to test the policy irrelevance of $\{Y_1, \dots, Y_k\}$ at s if both $\{Y_1, \dots, Y_{k-1}\}$ and $\{Y_k\}$ are policy irrelevant at s . This observation suggests an inductive procedure that first tests each individual state variable for policy irrelevance and then tests increasingly larger sets only as necessary. This inductive process will continue only so long as we find increasingly powerful abstractions.

We can afford to test each state variable at a given state, since the number of variables is relatively small. In contrast, the total number of states is quite large: exponential in the number of variables. We hence adopt an heuristic approach, which tests for policy irrelevance only at those states visited on some small number of trajectories through the task. For these states, we then determine what sets of state variables are policy irrelevant, as described above. For each set of state variables we can then construct a binary classification problem with a training set comprising the visited states. An appropriate classification algorithm then allows us to generalize the region over which each set of state variables is policy irrelevant. Note that in Section 3.2 we take steps to ensure that the classifiers’ generalization errors do not lead to the application of unsafe abstractions.

3.2 Exploiting irrelevance

Section 3.1 describes how to represent as a learned classifier the region of the state space where a given set of state variables is policy irrelevant. A straightforward approach to state abstraction would simply aggregate together all those

⁴We ameliorate this cost somewhat by initializing each MDP’s value function with the value function for the maximum likelihood MDP, as in [Strens, 2000].

⁵The converse is not necessarily true. Suppose we duplicate an otherwise always relevant state variable. Then each copy of the state variable is always policy irrelevant given the remainder of the state representation, but the pair of them is not.

states in this region that differ only on the irrelevant variables. However, this approach may prevent an RL algorithm from learning the correct value function and therefore the optimal policy. In Section 2.1 we gave a simple example of such an abstraction failure, even with perfect knowledge of policy irrelevance. Generalizing the learned classifier from visited states in one domain to unvisited states in a similar domain introduces another source of error. A solution to all of these problems is to encapsulate each learned state abstraction inside a temporal abstraction. In particular, we apply each state space aggregation only inside an option [Sutton *et al.*, 1999], which is an abstract action that may persist for multiple time steps in the original MDP.

Formally, for a set of state variables $\mathcal{Y}$ that is policy irrelevant over some $S' \subseteq S$, we construct an option $o = \langle \pi, \mathcal{I}, \beta \rangle$, comprising an option policy $\pi : [S']_{\mathcal{X}} \rightarrow A$, an initiation set $\mathcal{I} \subset S$, and a termination condition $\beta : S \rightarrow [0, 1]$. Once an agent executes an option o from a state in $\mathcal{I}$, it always executes primitive action $\pi(s)$ at each state s , until terminating with probability $\beta(s)$. We set $\mathcal{I} = S'$ and $\beta(s) = 0.01$ for $s \in \mathcal{I}$ and $\beta(s) = 1$ otherwise.⁶ Since $\mathcal{Y}$ is policy irrelevant over S' , we may choose an option policy π equal to the projection onto $[S']_{\mathcal{X}}$ of an optimal policy for the original MDP. An agent augmented with such options can behave optimally in the original MDP by executing one of these options whenever possible.

Although we believe that the discovery of this structure is interesting in its own right, its utility becomes most apparent when we consider transferring the discovered options to novel domains, for which we do not yet have access to an optimal policy. To transfer an option to a new domain, we simply copy the initiation set and termination condition. This straightforward approach suffices for domains that share precisely the same state space as the original domain. Even when the state space changes, our representation of $\mathcal{I}$ and β as a learned classifier gives us hope for reasonable generalization. We can also copy the option policy π , if we expect the optimal behavior from the original domain to remain optimal in the new domain.

In this paper we assume only that the policy irrelevance remains the same. We thus relearn the option policy concurrently with the learning of the high-level policy, which chooses among the original primitive actions and the discovered options. For each option, we establish an RL subproblem with state space $[\mathcal{I}]_{\mathcal{X}}$ and the same action space A . Whenever an option terminates in a state s , we augment the reward from the environment with a pseudoreward equal to the current estimate of the optimal high-level value function evaluated at s . We therefore think of the option not as learning to achieve a subgoal but learning to behave while ignoring certain state variables. In other words, the option adopts the goals of the high-level agent, but learns in a reduced state space.

Since each option is just another action for the high-level agent to select, RL algorithms will learn to disregard options as suboptimal in those states where the corresponding abstractions are unsafe. The options that correspond to safe

⁶The nonzero termination probability for $s \in \mathcal{I}$ serves as a probabilistic timeout to escape from bad abstractions.

state abstractions join the set of optimal actions at each appropriate state. The smaller state representation should allow the option policies to converge quickly, so RL algorithms will learn to exploit these optimal policy fragments instead of uncovering the whole optimal policy the hard way. We illustrate this process in the next section.

4 Results

We use Dietterich’s Taxi domain [Dietterich, 2000], illustrated in Figure 4, as the setting for our work. This domain has four state variables. The first two correspond to the taxi’s current position in the grid world. The third indicates the passenger’s current location, at one of the four labeled positions (Red, Green, Blue, and Yellow) or inside the taxi. The fourth indicates the labeled position where the passenger would like to go. The domain therefore has $5 \times 5 \times 5 \times 4 = 500$ possible states. At each time step, the taxi may move north, move south, move east, move west, attempt to pick up the passenger, or attempt to put down the passenger. Actions that would move the taxi through a wall or off the grid have no effect. Every action has a reward of -1, except illegal attempts to pick up or put down the passenger, which have reward -10. The agent receives a reward of +20 for achieving a goal state, in which the passenger is at the destination (and not inside the taxi). In this paper, we consider the stochastic version of the domain. Whenever the taxi attempts to move, the resulting motion occurs in a random perpendicular direction with probability 0.2. Furthermore, once the taxi picks up the passenger and begins to move, the destination changes with probability 0.3.

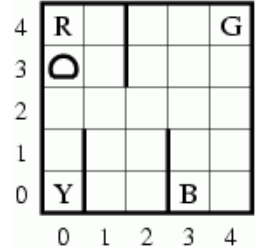


Figure 4: The Taxi domain.

This domain’s representation requires all four of its state variables in general, but it still affords opportunity for abstraction. In particular, note that the passenger’s destination is only relevant once the agent has picked up the passenger. We applied the methodology described in Sections 2 and 3 to the task of discovering this abstraction, as follows. First, we ran 25 independent trials of Q-learning to obtain samples of Q^* . For each trial, we set the learning rate $\alpha = 0.25$ and used ϵ -greedy exploration with $\epsilon = 0.1$. Learning to convergence required about 75000 time steps for each trial. This data allows us to compute the policy irrelevance of any state variable at any state. For example, consider again the passenger’s destination. To demonstrate the typical behavior of the testing procedure, we show in Figure 5a the output for every location in the domain, when the passenger is waiting at the upper left corner (the Red landmark), using the Wilcoxon signed-ranks test. The nonzero p values at every state imply that the passenger’s destination is policy irrelevant in this case. Note that the values are extremely close to 1 whenever the agent has only one optimal action to get to the upper left corner, which the procedure can then identify confidently. The squares with intermediate values are precisely the states in which more than one optimal action exists. Now consider

Figure 5b, which shows the output of the same test when the passenger is inside the taxi. The p values are extremely close to 0 in every state except for the four at the bottom middle, where due to the layout of the domain the agent can always behave optimally by moving north.

0.9999	0.9999	0.9999	0.9919	0.3784
0.9999	0.3188	0.9999	0.4731	0.5799
0.9999	0.1766	0.9999	0.9999	0.9999
0.9999	0.9999	0.5799	0.9999	0.4095
0.9999	0.9994	0.7940	0.9999	0.7703

(a)

0.0000	0.0000	0.0001	0.0002	0.0000
0.0000	0.0000	0.0004	0.0003	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.4412	0.4946	0.0000	0.0003
0.0000	0.9171	0.6518	0.0000	0.0000

(b)

Figure 5: The results of the Wilcoxon signed-ranks test for determining the policy irrelevance of the passenger’s destination in the Taxi domain. We show the result of the test for each possible taxi location for (a) a case when the passenger is not yet in the taxi and (b) the case when the passenger is inside the taxi.

Rather than compute the outcome of the test for every subset of state variables at every state, we followed the approach described in Section 3.1 and sampled 20 trajectories from the domain using one of the learned policies. We tested each individual state variable at each state visited, again using the hypothesis testing approach. We created a binary classification problem for each variable, using the visited states as the training set. For the positive examples, we took each state at which the hypothesis test returns a p value above the conservative threshold of 0.05. Finally, we applied a simple rule-learning classifier to each problem: the Incremental Reduced Error Pruning (IREP) algorithm, as described in [Cohen, 1995]. A typical set of induced rules follows:

1. Taxi’s x -coordinate:
 - (a) $y = 1 \wedge \text{passenger in taxi} \wedge \text{destination Red} \Rightarrow \text{policy irrelevant}$
 - (b) otherwise, *policy relevant*
2. Taxi’s y -coordinate:
 - (a) $x = 4 \wedge \text{passenger in taxi} \Rightarrow \text{policy irrelevant}$
 - (b) otherwise, *policy relevant*
3. Passenger’s destination:
 - (a) passenger in taxi $\Rightarrow \text{policy relevant}$
 - (b) otherwise, *policy irrelevant*
4. Passenger’s location and destination
 - (a) $(x = 1 \wedge y = 2) \vee (x = 1 \wedge y = 1) \Rightarrow \text{policy irrelevant}$
 - (b) otherwise, *policy relevant*

The sets of state variables not mentioned either had no positive training examples or induced an empty rule set, which classifies the state variables as relevant at every state. Rule set 3 captures the abstraction that motivated our analysis of this domain, specifying that the passenger’s destination is policy

relevant only when the passenger is in the taxi. The other three rules classify state variables as usually relevant, except in narrow cases. For example, rule 1a holds because the Red destination is in the upper half of the map, $y = 1$ specifies that the taxi is in the lower half, and all the obstacles in this particular map are vertical. Rule 2a is an example of an over-generalization. When holding the passenger on the rightmost column, it is usually optimal just to go left, unless the passenger wants to go the Green landmark in the upper-right corner.

We tested the generalization performance of these learned abstractions on 10×10 instances of the Taxi domain with randomly generated obstacles, running both horizontally and vertically. We placed one landmark near each corner and otherwise gave these domains the same dynamics as the original. Each abstraction was implemented as an option, as discussed in Section 3.2. Since the locations of the landmarks moved, we could not have simply transferred option policies from the original Taxi domain. In all our experiments, we used Q-learning with ϵ -greedy exploration⁷ to learn both the option policies and the high-level policy that chose when to apply each option and thus each state abstraction.⁸ To improve learning efficiency, we added off-policy training [Sutton *et al.*, 1999] as follows. Whenever a primitive action a was executed from a state s , we updated $Q(s, a)$ for the high-level agent as well as for every option that includes s in its initiation set. Whenever an option o terminated, we updated $Q(s, o)$ for every state s visited during the execution of o . Each state-action estimate in the system therefore received exactly one update for each timestep the action executed in the state. Figure 6 compares the learning performance of this system to a Q-learner without abstraction. The abstractions allowed the experimental Q-learner to converge much faster to an optimal policy, despite estimating a strict superset of the parameters of the baseline Q-learner.

5 Related work

Our approach to state abstraction discovery bears a strong resemblance to aspects of McCallum’s U-tree algorithm [McCallum, 1995], which uses statistical hypothesis testing to determine what features to include in its state representation. U-tree is an online instance-based algorithm that adds a state variable to its representation if different values of the variable predict different distributions of expected future reward. The algorithm computes these distributions of values in part from the current representation, resulting in a circularity that prevents it from guaranteeing convergence on an optimal state abstraction. In contrast, we seek explicitly to preserve optimality.

Our encapsulation of partial state abstractions into options is inspired by Ravindran and Barto’s work on MDP homomorphisms [Ravindran and Barto, 2003] and in particular their discussion of partial homomorphisms and relativized options. However, their work focuses on developing a more

⁷ $\epsilon = 0.1$ and $\alpha = 0.25$

⁸In general, SMDP Q-learning is necessary to learn the high-level policy, since the actions may last for more than one time step. However, this algorithm reduces to standard Q-learning in the absence of discounting, which the Taxi domain does not require.

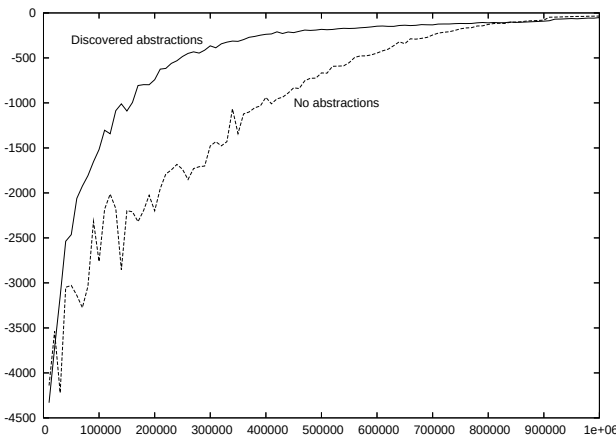


Figure 6: The average reward per episode earned by agents with learned abstractions encapsulated as options and only primitive actions, respectively, on a 10×10 version of the Taxi domain. The reward is averaged over 10000-step intervals. The results are the average of 25 independent trials.

agile framework for MDP minimization, not on the discovery of abstractions in an RL context.

Our work is also related to recent research into the automatic discovery of temporal abstractions [Özgür Şimşek and Barto, 2004; Mannor *et al.*, 2004], usually in the options framework. These techniques all seek to identify individual subgoal states that serve as chokepoints between well-connected clusters of states or that otherwise facilitate better exploration of environments. Our usage of options suggests an alternative purpose for temporal abstractions: to enable the safe application of state abstractions. Note that we can construe our definition of policy irrelevance as a statement about when a single reusable subtask *could* have contributed to several parts of an optimal policy.

The connection to hierarchical RL suggests the recursive application of our abstraction discovery technique to create hierarchies of temporal abstractions that explicitly facilitate state abstractions, as in MAXQ task decompositions [Dietterich, 2000]. This possibility highlights the need for robust testing of optimal actions, since each application of our method adds new potentially optimal actions to the agent. However, we leave the development of these ideas to future work.

6 Conclusion

This paper addressed the problem of discovering state abstractions automatically, given only prior experience in a similar domain. We defined a condition for abstraction in terms of the relevance of state variables for expressing an optimal policy. We described two statistical methods for testing this condition for a given state and set of variables. One method applies efficient statistical hypothesis tests to Q-values obtained from independent runs of an RL algorithm. The other method applies Monte Carlo simulation to a learned Bayesian model to conserve experience data. Then we exhibited an efficient algorithm to use one of these methods to discover what sets of state variables are irrelevant over what regions of

the state space. Finally, we showed that encapsulating these learned state abstractions inside temporal abstractions allows an RL algorithm to benefit from the abstractions while preserving convergence to an optimal policy.

Acknowledgments

We would like to thank Greg Kuhlmann for helpful comments and suggestions. This research was supported in part by NSF CAREER award IIS-0237699 and DARPA grant HR0011-04-1-0035.

References

- [Cohen, 1995] William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.
- [Dean and Givan, 1997] Thomas Dean and Robert Givan. Model minimization in Markov decision processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.
- [Dearden *et al.*, 1999] Richard Dearden, Nir Friedman, and David Andre. Model based Bayesian exploration. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.
- [Degroot, 1986] Morris H. Degroot. *Probability and Statistics*. Addison-Wesley Pub Co, 2nd edition, 1986.
- [Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
- [Mannor *et al.*, 2004] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 560–567, 2004.
- [McCallum, 1995] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1995.
- [Özgür Şimşek and Barto, 2004] Özgür Şimşek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 751–758, 2004.
- [Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.
- [Strens, 2000] Malcolm Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950, 2000.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [Sutton *et al.*, 1999] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.
- [Thrun and Schwartz, 1995] Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*, 1995.
- [Watkins, 1989] Watkins. *Learning From Delayed Rewards*. PhD thesis, University of Cambridge, England, 1989.

Signal-to-Score Music Transcription using Graphical Models

Emir Kapanci and Avi Pfeffer

Harvard University

Division of Engineering and Applied Sciences

33 Oxford Street, Cambridge, MA 02138

{kapanci,avi}@eecs.harvard.edu

Abstract

We present a transcription system that takes a music signal as input and returns its musical score. Two stages of processing are used. The first employs a fundamental frequency detector and an onset detector to transform input signals into a sequence of sound events. The onset detection is inherently noisy. This paper focuses on the second stage, going from sound events to a notated score. We use a family of graphical models for this task. We allow the results of onset detection to be noisy, necessitating a search over possible segmentations of the sound events. We use a large corpus of monophonic vocal music to evaluate our system. Our results show that our approach is well-suited to the problem of music transcription. The initial onset detection reduces the number of observations and makes the system less instrument specific. The search over segmentations corrects the errors in the onset detection. Without such reasoning, these errors are magnified in subsequent rhythm transcription.

1 Introduction

Music transcription is the task of transforming a music signal into a symbolic representation of the musical events it contains. There are three levels of representation of music that are relevant here. On the third level is the musical score in common music notation. This consists of musical events with attributes notated pitch (such as “A” in octave 4), notated duration (such as quarter note), and notated rhythmic position (such as the first beat of measure 22). On the second level is the representation of performance in terms of sound events. This is the level of MIDI data. The rhythmic attributes of events on this level are the actual onset time (such as 3.4 seconds from the beginning of the piece) and the actual duration (such as 0.5 seconds). The pitch attribute on this level could be discrete (such as MIDI note 66) or continuous (such as 66.46). On the first level is the representation of the sound signal. It is a common practice to transform the signal into a sequence of frames (through short-time Fourier transform), which are snapshots of the signal over a short period of time (e.g. 20 msec). A frame captures the composition of the sound by specifying the amplitude for every frequency.

The third level is the target representation of transcription. Previous authors [Raphael, 2002b; Cemgil and Kappen, 2003] have proposed using graphical models for automated music transcription, but their systems take MIDI data as input and only go from the second to the third level. Our goal is to build an end-to-end transcription system that goes all the way from a first level audio signal to a third level musical score. In this work we focus on monophonic transcription from a complex instrument such as voice. Monophonic transcription is a difficult enough task, and well worthy of study. It raises many issues that need to be addressed before polyphonic transcription can be attempted. Going from level 1 to 3 is a much more challenging task than going from level 2 to 3. In going from level 2 to 3, there is a one-to-one mapping between musical events. Furthermore, the input is often MIDI data where the pitch is given and does not need to be detected. Therefore the only work that needs to be done is to track the tempo of the music, thereby transforming actual onset times and durations into notated rhythmic positions and durations.

Going from level 1 directly to level 3 is too hard. Frames are too small a unit and even short music signals will be represented by very long sequences of frames. Each musical event consists of many frames that need to be grouped together. Furthermore, the information in a frame needs to be summarized. Instead of a frequency-amplitude spectrum, we need to know the fundamental frequency of a given frame. Therefore we use a preprocessing step in which we go from level 1 to level 2. This step consists of a fundamental frequency detector, which determines the fundamental frequency sounding in each frame, and an onset detector, which detects the beginnings of musical events from the frame sequence. Once a musical event has been identified, an overall pitch for the event can be deduced by analyzing all the frames constituting the event. We have previously built an onset detector capable of handling the soft onsets prevalent in vocal music, in which there is a smooth pitch change, or a vowel change [Kapanci and Pfeffer, 2004]. While achieving state-of-the-art performance, it is not completely reliable, and no perfect onset detector exists. It might miss some onsets, causing it to merge musical events, while reporting other spurious onsets, causing it to separate events. Furthermore, the pitch data produced by the onset detector is continuous-valued and noisy.

This paper focuses on the transformation from level 2 to 3 when the level 2 input is noisy. We do not assume a one-to-

one mapping between events at the two levels. Instead, we allow for the possibility that reported onsets are false positives. The onset detector can be tuned to trade off the rate of false positives with false negatives. We use a setting with a very low false negative rate, so that most of the errors in onset detection are false positives. We also detect the pitch, allowing the tuning of the performance to change over time. So we need to simultaneously (1) detect false positives in the segmentation, (2) do tempo tracking and rhythm transcription, and (3) do tuning tracking and pitch transcription.

We propose a family of graphical models to accomplish these tasks. In the models, the properties of sound events depend probabilistically on the properties of notated events to which they are related. Because there may be false onsets, several sound events may actually be related to the same notated event. The segmentation determines which sound events are grouped together into a single note event, and therefore it determines the structure of the graphical model. We have a different graphical model corresponding to each possible segmentation. We use Markov Chain Monte Carlo (MCMC) sampling to search over the different structures and parameters.

We evaluate our model using a corpus of monophonic vocal music. Our results show that in the absence of perfect onset detectors, reasoning about segmentation is necessary for a signal-to-score system. Without such reasoning, the errors in the segmentation are magnified in subsequent rhythm transcription. We also show that the transcription benefits from musical knowledge, which is easily incorporated into our graphical model as prior distributions over the scores.

2 Problem Statement

The input of our system is a music signal, and the output is a musical score. The music signal is first processed using an onset detector to yield the sound event observations $y_{1:N}$ for the graphical model. Each sound observation y_n consists of an observed duration t_n and a frequency f_n . The task is to find a segmentation $s_{1:K}$ and a score $x_{1:K}$. Each score event x_k consists of a notated duration ρ_k and a pitch π_k . The segmentation variable s_k is the number of observed events related to the k th score event. Thus the segmentation determines the mapping between score events and observed events. N is the number of observations (fixed), while K is the number of note events (dependent on the segmentation).

The goal is to find the segmentation $s_{1:K}$ and the score $x_{1:K}$ that maximize the posterior probability of the segmentation and score given the observed data:

$$\begin{aligned} (s_{1:K}, x_{1:K})^* &= \arg \max_{s_{1:K}, x_{1:K}} p(s_{1:K}, x_{1:K} | y_{1:N}) \\ &= \arg \max_{s_{1:K}, x_{1:K}} p(y_{1:N} | s_{1:K}, x_{1:K}) p(s_{1:K}) p(x_{1:K}) \end{aligned} \quad (1)$$

assuming independence of the prior distributions over score and segmentation. The graphical model detailed in section 3 defines $p(y_{1:N} | s_{1:K}, x_{1:K})$. We have a style specific score prior $p(x_{1:K})$, and a segmentation prior $p(s_{1:K})$ based on the onset detector's outputs. The output of the system is the score $x_{1:K}^*$ and the mapping of observations to the score events $s_{1:K}^*$. We evaluate it using a metric presented in Section 5.

For our onset detector, we use our previously published hierarchical method that detects onsets in a signal by comparing frames one to another [Kapanci and Pfeffer, 2004]. A comparison function based on the fundamental frequencies and amplitudes returns the dissimilarity of two frames. If this value is above a threshold, an onset is declared over the region delimited by the two frames. The system begins by comparing adjacent frames, and then gradually increases the distance between the compared frames. By comparing frames some distance apart, the detector is able to detect soft onsets that happen smoothly over time.

To reduce the number of missed onsets, we use a low dissimilarity threshold. That way, we get many extraneous onsets (false positives) which need to be eliminated in later stages, but missed onsets (false negatives) are infrequent. In our experiments, 40% of the onsets were extraneous, while only 1.39% of the onsets were missed.

Once the onsets are marked, the signal is segmented by cutting it at these onsets. A continuous pitch attribute is computed for each event by the average fundamental frequency. The comparison value for each detected onset is also stored to be used in assigning a prior segmentation probability to each observation y_n , as described in the next section.

3 Model

The likelihood of a sequence of observed sound events is given by a family of graphical models. Due to the possibility of false onsets, there are many possible groupings of sound events (observations) into note events. The segmentation determines which sound events are grouped together, and therefore it determines the structure of the graphical model. For each segmentation $s_{1:K}$, the likelihood model defines the probability of observations $y_{1:N} = (t_{1:N}, f_{1:N})$ given $x_{1:K} = (\rho_{1:K}, \pi_{1:K})$. The probabilistic dependency of sound events on the notated events is the same in different structures. The difference is in the connections between the two layers as illustrated in Figures 1 and 2: the leaf nodes correspond to the observations, and the segmentation determines the structure of the graph above them. The mapping between observation indexes (ns) and note event indexes (ks) is given by $g(n) = 1 + \max_{S_k < n} k$, where $S_k = \sum_{i=1}^{k-1} s_i$.

The tempo and tuning are continuous processes changing smoothly over time: we model both as first-order Gaussian processes (Equations 2 and 5). The tempo determines how notated durations become observed durations: the notated duration is multiplied by the tempo to produce the observed duration. In addition, we model the fact that durations will deviate without reflecting a change in tempo. This is accomplished, first of all, by including a Gaussian noise ζ_{τ_k} in the model for the observed duration (Equation 4).

We further model the fact that as the tempo remains constant, successive deviations are likely to compensate for each other, so that a note that sounds for shorter than its written duration will be followed by a note that sounds for longer, and vice versa. This may be the result of both intentional and unintentional deviations. It is common in various styles to intentionally produce this effect. For example, in swing rhythm, given two successive eighth notes, the first is lengthened and

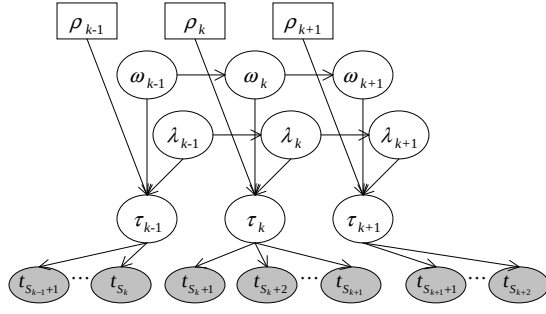


Figure 1: Rhythm Model

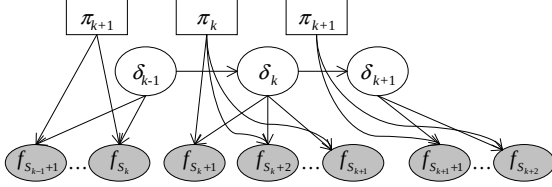


Figure 2: Pitch Model

the second is shortened. As another example, double-dotting is a common effect in Baroque music. Here a dotted note is lengthened and the subsequent note is shortened. This compensation model is also a natural model of unintentional deviation. The performer is likely to maintain a constant beat, so that the notes that subdivide a beat have a given amount of time to fill. As a result, shorter notes will be compensated for by longer notes to fill up the required time. We capture this effect by including a first-order stealing process (Equation 3), which allows successive durations to deviate in opposite directions without affecting the overall tempo. This is accomplished by having a negative value for the stealing coefficient c_λ . The value -0.25 was used in our experiments. Given the note duration ρ_k s, the observed durations depend linearly on the tempo and stealing processes (Equation 4).

For the pitch process, the tuning is an additive offset that is added to the notated pitch to produce the observed frequency (Equation 6). In addition, we allow the frequency to deviate independent of the tuning, so Equation 6 includes a Gaussian noise term ζ_{f_n} . The noise terms in Equations 4 and 6 model observational errors as well as local deviations.

$$\text{Tempo: } \omega_k = \omega_{k-1} + \zeta_{\omega_k} \quad \text{for } k > 0 \quad (2)$$

$$\text{Stealing: } \lambda_k = c_\lambda \lambda_{k-1} + \zeta_{\lambda_k} \quad \text{for } k > 0 \quad (3)$$

$$\text{Duration: } \tau_k = (\omega_k + \lambda_k) \rho_k + \zeta_{\tau_k} \quad (4)$$

$$\text{Tuning: } \delta_k = \delta_{k-1} + \zeta_{\delta_k} \quad \text{for } k > 0 \quad (5)$$

$$\text{Frequency: } f_n = \delta_k + \pi_k + \zeta_{f_n} \quad \forall k, n \text{ s.t. } k = g(n) \quad (6)$$

We assume that ω_0 is $\mathcal{N}(1, q_\omega)$, λ_0 is $\mathcal{N}(0, q_\lambda)$, δ_0 is $\mathcal{N}(0, q_\delta)$, ζ_{ω_k} is $\mathcal{N}(0, Q_\omega)$, ζ_{λ_k} is $\mathcal{N}(0, Q_\lambda)$, ζ_{τ_k} is $\mathcal{N}(0, Q_\tau)$, ζ_{δ_k} is $\mathcal{N}(0, Q_\delta)$ and ζ_{f_n} is $\mathcal{N}(0, Q_f)$, all independent of each other. The variance values used in the experiments were: $q_\omega = 10^{-2}$, $q_\lambda = 10^{-3}$, $q_\delta = 10^{-1}$, $Q_\omega = 10^{-4}$, $Q_\lambda = 10^{-3}$, $Q_\tau = 10^{-2}$, $Q_\delta = 10^{-2}$ and $Q_f = 10^{-1}$.

Note that we don't observe τ_k directly, but only through the durations t_n of the individual sound events, where $\tau_k = \sum_{n=1+S_k}^{S_{k+1}} t_n$. On the other hand, the model directly specifies the observed frequency f_n of the individual sound events.

We define the rhythm prior $p(\rho_1, \dots, \rho_K)$ by a Hidden Markov Model (HMM), learned offline using rhythm sequences of a corpus of music in the same style as our evaluation data. For the pitch prior we use an n -gram model (with $n=3$), which assumes that each element in a sequence only depends on the previous $n-1$ elements. We define the prior over intervals rather than individual pitches since this allows for generalization to different keys. The probabilities in the model are estimated offline by the n -gram frequencies in the training data.

The segmentation prior indicates, for each observed sound event, the probability that there is truly an onset immediately after the event. To obtain the segmentation prior, we use the dissimilarity values returned by the onset detector at each observation. The goal is to individually estimate $p(\text{onset}_n | d_n)$, the probability that there is an actual onset right after the n th observation, given its dissimilarity d_n to the following observation. Although not a formal density estimation method, sigmoid fitting provides an appropriate solution [Platt, 1999]. This was suggested to transform a classifier's output values into probabilities. Using a training corpus for onset detection, we learned the following function: $f(d) = (1 + \exp(-2.64d + 2.59))^{-1}$. The shape of the function makes sense: as the dissimilarity increases, it assigns increasing probabilities in $[0,1]$ to the presence of an onset. In order to avoid probabilities close to 0 and 1, we used a weighted average of $f(d)$ and 0.5 as our prior: $p(\text{onset}_n | d_n) = 0.8f(d_n) + 0.2 \times 0.5$.

Once individual segmentation priors of all observations are computed, the prior probability of a segmentation $s_{1:K}$ is computed by multiplying these together:

$$p(s_{1:K} | d_{1:N}) = \prod_{n=1; g(n) \neq g(n+1)}^{N-1} p(\text{onset}_n | d_n) \times \prod_{n=1; g(n) = g(n+1)}^{N-1} (1 - p(\text{onset}_n | d_n))$$

4 Inference

Our goal is to compute the maximum a posteriori (MAP) segmentation-score configuration as given in Equation 1. Let $\mathcal{R}$ be the set of possible note durations and $\mathcal{P}$ the set of possible pitches. For a given observation sequence $y_{1:N}$, there are 2^{N-1} possible segmentations and $(|\mathcal{R}| \times |\mathcal{P}|)^K$ possible scores for each segmentation with K note events. The exponential number of the possible $(s_{1:K}, x_{1:K})$ pairs makes exact computation of the MAP configuration intractable. We therefore use MCMC methods [Gilks *et al.*, 1996] to sample from the space of possible pairs.

Our inference procedure is similar to the handling of reference uncertainty in first-order probabilistic models [Pasula and Russell, 2001]. We alternate between MCMC steps to change the segmentation, and Gibbs steps to sample the duration and pitch for a given segmentation. When changing the segmentation, reversible jump MCMC [Green, 1995] is employed since the model dimension changes whenever we split or merge events.

4.1 Reversible Jump MCMC

MCMC methods are used to generate samples from a complex target distribution where analytical or numerical techniques are not applicable. In our case, the target is the posterior distribution of the transcription given the observations, $p(z|y) = p(s, \delta, \pi|y)$, and we are trying to find the mode z^* of this distribution. There are well-known methods to construct a Markov chain whose stationary distribution is guaranteed to be the target distribution. In the Metropolis-Hastings algorithm, the candidate samples are generated from a proposal distribution $q(z'|z)$ and are accepted with probability $\alpha(z, z') = \min\left(1, \frac{p(z')q(z|z')}{p(z)q(z'|z)}\right)$. This is not applicable when $p(z|y)$ does not have a fixed dimensionality, such as in our problem: different segmentations may correspond to different numbers of notes, and therefore have different numbers of parameters. The reversible jump MCMC method allows us to sample from a target distribution even when the dimensionality of the model is not fixed. The acceptance probability of moving from one model to another model of different dimension is given by:

$$\alpha_{mm'}(z, z') = \min\left(1, \frac{p(m') p_{m'}(z'|y) p_{m'm} q_{mm'}(z', u') \left| \frac{\partial g_{mm'}(z, u)}{\partial z \partial u} \right|}{p(m) p_m(z|y) p_{mm'} q_{mm'}(z, u)}\right) \quad (7)$$

where m and m' are model indicators that determine the number of notes, z and z' are the parameter vectors of sizes m and m' respectively, $p(m)$ is the prior probability of model m , and $p_{mm'}$ is the probability of moving from model m to m' . The function $g_{mm'}(z, u) = (z', u')$ is a mapping between the parameters of the two models. This mapping may entail random decisions to generate the parameters z' of m' given the current parameters z of m . These decisions are denoted by the variable u , whose probability distribution is $q_{mm'}(z, u)$.

Since we assume that the segmentation, rhythm and pitch priors are independent, for the first two terms of the fraction in Equation 7 we have:

$$p(m) p_m(z|y) \propto p(m, s) p(\delta) p(\pi) p(y|m, z) \quad (8)$$

The terms in the right hand side of Equation 8 are respectively given by the segmentation prior, the rhythm prior, the pitch prior and the likelihood model.

During our sampling, we allow two kinds of moves that change the model dimension: merging two neighboring notes together, or splitting a single note into two. The rhythm and pitch components are sampled using Gibbs sampling without a change in model dimension, so we don't discuss the acceptance probability for those here and instead focus on the segmentation component s of z . We have two possible values for $p_{mm'}$: it is equal to p_{split} when $m' = m + 1$ and to p_{merge} when $m' = m - 1$. The two functions g_{merge} and g_{split} define the mappings between the models before and after, where the move is a merge and a split respectively:

$$g_{\text{merge}}(s_1, s_2) = (s_1 + s_2, s_1) \quad (9)$$

$$g_{\text{split}}(s, u) = (u, s - u) \quad (10)$$

In Equation 10, because we choose the splitting point randomly from all possible boundaries between the observations

currently assigned to a note, u is a random variable with uniform density $q_{\text{split}}(s, u) = \frac{1}{s-1}$. Since merging is a deterministic move, we don't have a random variable in Equation 9 so we simply have $q_{\text{merge}}(s, u) = 1$. Finally, it's easy to see from Equations 9 and 10 that both $\left| \frac{\partial g_{\text{merge}}(z, u)}{\partial z \partial u} \right|$ and $\left| \frac{\partial g_{\text{split}}(z, u)}{\partial z \partial u} \right|$ are equal to 1.

Putting everything together, we have the following two acceptance probabilities for merge and split moves:

$$\alpha_{\text{merge}}(z, z') = \min\left(1, \frac{p(z) p(y|z) p_{\text{split}} (s_k - 1)^{-1}}{p(z') p(y|z') p_{\text{merge}}}\right) \quad (11)$$

$$\alpha_{\text{split}}(z, z') = \min\left(1, \frac{p(z) p(y|z) p_{\text{merge}}}{p(z') p(y|z') p_{\text{split}} (s_k - 1)^{-1}}\right) \quad (12)$$

where s_k is either the total number of observations associated with the two notes we are merging, or it is the number of observations associated with the note we are splitting.

4.2 Generating Samples

Given a sample $C = (s_{1:K}^{(i)}, x_{1:K}^{(i)})$, new samples are generated as follows:

1. Pick an index k .

2. Resample the k th note, keeping the segmentation and all other notes the same:

$$\begin{aligned} s_j^{(i+1)} &\leftarrow s_j^{(i)} & \forall j \in [1; K] \\ x_j^{(i+1)} &\leftarrow x_j^{(i)} & \forall j \in [1; K] \setminus \{k\} \end{aligned}$$

Sample a value for $x_k^{(i+1)}$ from $p(x_k | s_{1:K}, x_{-k}, y_{1:N})$.

Let $z = (s_{1:K}^{(i+1)}, x_{1:K}^{(i+1)})$. It is accepted automatically as a sample since it was generated by Gibbs sampling.

3. Change segmentation in one of two ways:

a. With probability p_{merge} , merge the k th note with its successor:

$$\begin{aligned} K' &\leftarrow K - 1 \\ s_k^{(i+2)} &\leftarrow s_k^{(i)} + s_{k+1}^{(i)} \\ \{s_j^{(i+2)}, x_j^{(i+2)}\} &\leftarrow \{s_j^{(i)}, x_j^{(i)}\} & \forall j \in [1; k-1] \\ \{s_j^{(i+2)}, x_j^{(i+2)}\} &\leftarrow \{s_{j+1}^{(i)}, x_{j+1}^{(i)}\} & \forall j \in [k+1; K'] \end{aligned}$$

Pick a value for $x_k^{(i+2)}$ using Gibbs sampling.

b. With probability $p_{\text{split}} = 1 - p_{\text{merge}}$, split the k th note into two at r , an integer chosen uniformly at random from $[1, s_k - 1]$ (only possible if $s_k > 1$).

$$\begin{aligned} K' &\leftarrow K + 1 \\ s_k^{(i+2)} &\leftarrow r \\ s_{k+1}^{(i+2)} &\leftarrow s_k^{(i)} - r \\ \{s_j^{(i+2)}, x_j^{(i+2)}\} &\leftarrow \{s_j^{(i)}, x_j^{(i)}\} & \forall j \in [1; k-1] \\ \{s_j^{(i+2)}, x_j^{(i+2)}\} &\leftarrow \{s_{j-1}^{(i)}, x_{j-1}^{(i)}\} & \forall j \in [k+2; K'] \end{aligned}$$

Jointly sample values for $x_k^{(i+2)}$ and $x_{k+1}^{(i+2)}$ from $p(x_k, x_{k+1} | s_{1:K}, x_{-(k:k+1)}, y_{1:N})$.

4. Let $z' = (s_{1:K'}^{(i+2)}, x_{1:K'}^{(i+2)})$. It is a candidate sample that has a different segmentation than z . The acceptance probability of z' as the next sample is given by one of Equations 11 and 12, decided by whether it corresponds to a merge or a split move.

In step 1 above, although we could pick a random k , processing the events sequentially ($k = 1, \dots, K$) allows computational savings. We can factorize the likelihood as:

$$p(y_{1:N} | s_{1:K}, x_{1:K}) = p(y_{1:N_{k+1}} | s_{1:k}, x_{1:k}) \times \int_{h_{k:K}} p(y_{1+N_{k+1}:N} | s_{k+1:K}, x_{k+1:K}, h_{k:K}) dh_{k:K} \quad (13)$$

where h denotes the hidden variables ω , λ , and δ . The first factor is computed incrementally as new samples are taken for $k = 1$ to K . The second factor depends only on the future observations, and can be efficiently computed in advance for all k s by a backward filtering pass.

Given $(s_{1:K}, x_{1:K})$, the relationships between the continuous variables of our model are linear, and the additive change or noise variables are all Gaussian. So, it is a linear dynamical system where the integration over the hidden variables (in Equation 13) can be computed analytically using Kalman filtering. All the necessary probabilities can be represented by Gaussian potentials, over which multiplication, conditioning and marginalization are three basic operations.

A problem with this individual sampling of x_k s is that sometimes the neighboring notes will be such that it is very unlikely to sample a good value for either one with the current assignment of the other. To overcome such deadlocks, we randomly sample some score pairs (x_k, x_{k+1}) jointly in Step 2. Steps 3 and 4 are not performed for joint samples. Ideally, one would like to sample all events jointly given a segmentation, but this would result in an exponential sampling space.

We define an MCMC iteration as one sequence of samplings from $k = 1$ to K . After some iterations, we randomly propose a global sampling step. We have two global operations: one that shifts all the pitches in the sample by ± 1 , and one that halves or doubles all durations. If the result of halving produces a duration not in $\mathcal{R}$, we randomly select the closest element in $\mathcal{R} \cup \{0\}$. When a zero duration is assigned to an observation, it is grouped with one of its neighbors. In the experiments we used 20 chains with 250 iterations per chain. Each chain starts with an initial sample that is obtained by matching the observations as closely as possible: If the segmentation prior of an observation is less than 0.5, we group it with its successor. The durations are rounded to the closest duration in $\mathcal{R}$, and the frequencies are rounded to the closest pitch in $\mathcal{P}$. Starting with this sample, as opposed to a random one, allows us to get to a good transcription faster.

5 Experiments

5.1 Training and Evaluation Data

A corpus of monophonic vocal music was used to evaluate the system and its various components. The corpus consisted of two parts. One part contained both scores and audio data, and was used to evaluate the system. The test data had a total duration of 60 minutes, composed of 55 audio segments containing a total of 3606 notes. All pieces were by the Renaissance composer G.P. Palestrina (16th century), and sung by the same singer. The second part of the corpus consisted only of scores without audio data. This part was used to learn the rhythm and pitch priors. There were 78 pieces by Palestrina in this part, none of which appeared in the first part. Two

audio fragments were removed from the first part for the purpose of tuning the eight variance parameters of the graphical model and the two sigmoid parameters of the segmentation prior. These were not used in the evaluation.

5.2 Evaluation Metric

Given the true score $x_{1:K_T}^T$ and segmentation $s_{1:K_T}^T$ for observations $y_{1:N}$, we evaluate the system's result $x_{1:K}^*$ and $s_{1:K}^*$ as follows:

Segmentation: We count the number of correct onset/no-onset decisions over the number of possible segmentation locations ($N - 1$).

Musical Score: Since the system's segmentation might be different from the reference segmentation, comparing $x_{1:K}^*$ directly to $x_{1:K_T}^T$ is not very meaningful. We instead evaluate the score by looking at the number of rhythm and pitch errors remaining once the segmentation errors are fixed. For instance if a half note is transcribed as two successive quarter notes of the same pitch, we consider this to be a segmentation error (extra onset), and not a rhythm transcription error. Similarly, if two successive quarter notes of the same pitch are transcribed as a half note, this is only a segmentation error (missed onset). If, on the other hand, two successive quarter notes of different pitches are transcribed as a half note, there is a pitch error in addition to the segmentation error. This is because even after correcting the missed onset by splitting the half note in two, our transcription would have a pitch different from the real score.

Returning to the extra onset case, if a half note is transcribed as two separate quarter notes of different pitches, then when the segmentation error is corrected the two quarter notes are replaced by a half note with a single pitch. We need to choose one of the pitches of the two quarter notes as the pitch of the half note. There may or may not be a pitch error in addition to the segmentation error, depending on which pitch is chosen for the half note. We choose to go with the pitch whose related observation is longer in duration. If this is equal to the real pitch, there is no pitch error. This generalizes naturally to cases where a single true note is transcribed as multiple notes.

For the evaluation of durations, if a note is transcribed as multiple notes (extra onsets), we compare the sum of these durations to the reference duration. If multiple notes are transcribed as a single note (missed onset), we compare the duration of this note to the sum of the reference durations. We can extend this idea to evaluate cases of misplaced onsets, where the computed transcription contains an onset in a different place from the reference transcription. In both the reference and computed transcriptions, we only keep the onsets that appear in both, and compute durations for these new segments by summing the durations of their constituents. These are then compared to evaluate the rhythm transcription.

5.3 Results

We present the results for the evaluation of our system in Tables 1 and 2. Table 1 contains the results obtained by the sampling process described in Section 4. This process generates samples from the posterior distribution of the transcription.

Table 1: Results

	Seg.	Pitch	Dur.	Comb.	Time
All	87.28	86.11	85.42	86.26	82.25
All-R	87.63	95.08	49.81	71.42	25.60
All-P	86.85	86.67	86.29	86.60	75.44
All-SG	83.30	87.01	82.65	84.28	82.88
All-ST	86.98	83.21	85.36	85.16	67.48
All-J	86.11	81.52	79.14	82.16	99.67
None	71.27	93.81	36.60	57.68	9.10
All-RG	65.19	92.38	62.09	70.97	71.17
None-RG	62.15	92.27	19.61	38.50	4.93

Table 2: Results (Non-Reversible)

	Seg.	Pitch	Dur.	Comb.	Time
All	87.08	89.60	87.15	87.93	81.02
All-R	89.32	93.76	50.06	71.71	32.44
All-P	87.53	88.27	87.68	87.83	69.55
All-SG	83.08	87.15	83.16	84.42	82.17
All-ST	86.98	88.22	87.47	87.55	66.82
All-J	85.24	84.36	80.26	83.23	99.93
None	75.69	91.56	46.12	65.48	8.85
All-RG	65.19	92.38	62.09	70.97	71.17
None-RG	62.15	92.27	19.61	38.50	4.93

However, we are mainly interested in the mode of this distribution, so we do not really need to ensure that our sampling has the posterior as its stationary distribution. We could employ any search process over the space of transcriptions, and output the one with the highest posterior. Since we know that our input is over-segmented, a simple modification is to bias our sampler towards merging segments. We implement this by removing the $(s_k - 1)^{-1}$ factors from Equations 11 and 12. Other approaches such as simulated annealing can also be used to generate samples concentrated around the modes. However, we were able to obtain slightly higher posteriors with the same number of iterations using this simpler modification. The results using the modification are presented in Table 2, and we use that table to discuss our results. It is important to keep in mind that both Tables 1 and 2 use samples to approximate the MAP transcription. It is likely that the exact MAP configurations would have higher posteriors, with possibly higher or lower evaluation scores. However, given the high number of iterations, the values are still representative of the performance of their respective models.

In both Tables 1 and 2, the first three columns contain the success rates for the segmentation, pitch transcription and rhythm quantization, using the metric defined in Section 5.2. The next column contains the harmonic mean¹ of these three as a combined metric. The last column gives the average time per iteration (in msecs) for a 24.92 seconds input signal. The first row (All) refers to the system with all of the priors described in Section 3. The next three rows show the results when either prior is removed: rhythm (All-R), pitch (All-P), or segmentation (All-SG). We then present results where we don't include the stealing process given in Equation 3 (All-

ST), and where we don't allow joint sampling (All-J). The next row contains the results when none of these five components is used (None). The final two rows (All-RG and None-RG) show the results when we don't allow regrouping of observations, so that every observation is a separate segment. This is what we would get if we did not reason about segmentation, and instead assumed that the onset detector is perfect. Notice that these two rows are the same in the two tables since the segmentation is fixed.

The most instructive comparison is between All and All-RG. We see that not only does All-RG do significantly worse on segmentation, which is expected, its score for duration is also much lower. It seems to be the case that errors in segmentation spill over, causing significant errors in rhythm as well. Surprisingly, however, All-RG does a little better on pitch transcription. This may be because it is not subject to missed onset errors, which as we saw earlier always results in a pitch error when two notes of different pitches are transcribed as a single note. Comparison of None and None-RG is similar. Again, the system that allows regrouping greatly outperforms the one that does not on segmentation and rhythm, but does slightly worse on pitch.

Now we consider the effect of individual components of the model. We see that the rhythm prior greatly improves the performance on duration, while slightly hurting the segmentation and pitch scores. It could be that without a rhythm prior, the segmentation prior and observed pitches have a greater effect on the segmentation, and the pitches can match the observations more closely. Overall, however, the small negative effect on pitch and segmentation is well compensated for by the improvement in duration. The pitch prior has only a very small effect. It leads to a slight improvement in pitch, and a slight degrading of the segmentation and duration scores. The segmentation prior and joint sampling lead to an improvement in all three scores. The stealing process has little effect, increasing the pitch score and slightly decreasing the other two scores. It might be more useful in other styles of music.

In Figure 3, we show the transcription results for a short audio segment. The true transcription is given first, followed by the transcriptions returned by each of the systems in Table 2. The errors are indicated by the arrows above them. The complete system (All) is able to transcribe the piece with no mistakes. We see that by removing the rhythm prior (All-R), we get many stylistic errors: the notated durations of the first, forth and fifth errors are very unlikely, and they induce an unlikely rhythmic pattern. Removing the segmentation prior (All-SG) results in segmentation errors, which are often accompanied by pitch and/or rhythm mistakes. We see this in the second error, where the pitch has been transcribed incorrectly once the note is split into two. The stealing prior provides flexibility to transcribe local expressive or motor deviations correctly. Without it (All-ST), we are unable to correctly transcribe the first note at the beginning of the fourth measure: its elongated performance requires a too large deviation in the global tempo to be transcribed correctly. In the bottom two rows of Figure 3, we clearly see that the two systems without reasoning about the segmentation have many mistakes, and the quality of transcription is really poor, especially for the

¹ $H(x, y, z) = \frac{3xyz}{xy + xz + yz}$ is similar to F-measure for 3 numbers.



Figure 3: Transcription example

last row. Even in the All-RG system that includes rhythm and pitch priors, we can see that segmentation errors result in many additional pitch and rhythm errors. So, we clearly see the benefits of reasoning about the segmentation in this example.

6 Related Work

Transcription of music signals to scores is a difficult problem. Most related work consists of addressing subproblems such as pitch tracking, onset detection, beat tracking, level 1-to-2 (audio to MIDI or piano-roll), and level 2-to-3 (MIDI to score).

There are a number of related works on the level 1-to-2 problem. Most systems transform the signal into a time-frequency representation, followed by heuristic methods to segment the signal and label segments with pitch or chord names (see [Klapuri, 2004] for a more detailed literature re-

view). Sterian (1999) proposes heuristics to pick peaks from the time-frequency image. He then uses a Kalman filter to extract partial tracks and searches over hypotheses for the association of the tracks to notes. Marolt (2004) presents a connectionist approach: oscillator networks for partial tracking and neural networks for note recognition. Raphael (2002a) computes a set of features from each frame of the time-frequency image, and takes these as observations from an HMM. Cemgil *et al.* (2004) propose a graphical model whose observation is directly the time domain signal.

Beat tracking is another problem related to rhythm transcription. Dixon (2001) proposes a rule-based system to estimate beats from a sequence of onset times. Goto and Murakoa (1998) take an audio signal as input and use DSP methods to construct and select from multiple beat hypotheses. We consider beats to lie somewhere between levels 2 and 3: given perfect onset times, the beats could be used to assign discrete

durations to observed events. However, onset detection accuracy is not crucial for beat trackers. On the contrary, poor onset detection seems to help beat tracking by filtering out less salient onsets. This is clearly undesirable for a transcription system.

There have been systems employing graphical models for level 2-to-3 problem: Given a list of onset times, Cemgil and Kappen (2003) use MCMC methods to sample most likely note durations. Raphael (2002b) proposes an exact method for the joint estimate of the tempo and note durations. He introduces a “thinning” operation that removes a value from consideration for a discrete variable, if that variable is guaranteed not to have that value in the MAP configuration. This seems to keep the search space from growing exponentially. However, both of these systems assume that their list of onset times is accurate, so they have a one-to-one mapping between observations and score events. In our case, the number of mappings alone is exponential, which is why we chose approximate methods.

We believe that exploiting the advances in DSP to get to level 2 (even a noisy one), and then using graphical models for level 2-to-3 is appropriate. Level 2 data is much more instrument independent, and it is much shorter. However, without perfect onset detectors in sight, level 2 data will inevitably have errors, and our experiments have shown that these errors are magnified in the subsequent analysis to get to level 3. The most important contribution of our system is its ability to accept and reason about noisy input. To the best of our knowledge, it is the first music transcription system to go from level 1 to level 3 without assuming a perfect level 2.

7 Discussion and Conclusion

We have presented a transcription system that takes a music signal as input and returns its musical score. We use a noisy onset detector as our front-end, which computes the input to a family of graphical models. Each regrouping of this input corresponds to a specific model from this family. Our inference procedure jointly searches over models and scores that may have produced the observed sound events. Our results show that this is a very well-suited approach to the problem of music transcription.

The constraint on our onset detection noise is that we can have extra onsets but not many missed onsets, since these will be reflected as merged notes in the final score. This is a reasonable requirement, as most detectors can be tuned to reduce missed onset at the cost of increased extra onsets. We can therefore use a variety of onset detectors as our front-end. Also, in theory we could apply our system to individual frames, although the inference would take much longer.

In our models, the pitch and rhythm variables are conditionally independent given the segmentation. This keeps the sampling over the two spaces separate. A more sophisticated model could combine the rhythm and pitch processes as well as the priors. Also, the rhythm and pitch priors we learn are style-specific due to the homogeneity of the training data. This makes sense, as a trained musician would transcribe the same audio signal differently under different stylistic assumptions. Ultimately, we would like to have rhythm priors for

various styles and make the choice of style a variable that needs to be optimized as well.

References

- [Cemgil and Kappen, 2003] Ali Taylan Cemgil and Bert Kappen. Monte Carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18:45–81, 2003.
- [Cemgil et al., 2004] Ali Taylan Cemgil, Bert Kappen, and David Barber. A generative model for music transcription. *IEEE Transactions on Speech and Audio Processing*, 2004.
- [Dixon, 2001] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of the New Music Research*, 30(1):39–58, 2001.
- [Gilks et al., 1996] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996.
- [Goto and Muraoka, 1998] Masataka Goto and Yoichi Muraoka. An audio-based real-time beat tracking system and its applications. In *Proceedings of the International Computer Music Conference*, San Francisco, California, 1998.
- [Green, 1995] Peter J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.
- [Kapanci and Pfeffer, 2004] Emir Kapanci and Avi Pfeffer. A hierarchical approach to onset detection. In *Proceedings of the International Computer Music Conference*, Miami, Florida, 2004.
- [Klapuri, 2004] Anssi Klapuri. *Signal Processing Methods for Automatic Transcription of Music*. PhD thesis, Tampere University of Technology, Tampere, Finland, 2004.
- [Marolt, 2004] M. Marolt. A connectionist approach to transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449, 2004.
- [Pasula and Russell, 2001] Hanna Pasula and Stuart Russell. Approximate inference for first-order probabilistic languages. In *Proceedings of the International Joint Conference of Artificial Intelligence*, Seattle, Washington, 2001.
- [Platt, 1999] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Samola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 185–208. MIT Press, Cambridge, MA, 1999.
- [Raphael, 2002a] Chris Raphael. Automatic transcription of piano music. In *Proceedings of the International Symposium on Music Information Retrieval*, Paris, France, 2002.
- [Raphael, 2002b] Chris Raphael. A hybrid graphical model for rhythmic parsing. *Artificial Intelligence*, 137(1-2):217–238, 2002.
- [Sterian, 1999] Andrew Sterian. *Model-Based Segmentation of Time-Frequency Images for Music Transcription*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 1999.

A Hybrid Discriminative/Generative Approach for Modeling Human Activities

Jonathan Lester¹, Tanzeem Choudhury², Nicky Kern³, Gaetano Borriello^{2,4} and Blake Hannaford¹

¹Department of Electrical Engineering, University of Washington, Seattle WA 98195, USA

²Intel Research Seattle, Seattle, WA 98105, USA

³Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany

⁴Department of Computer Science, University of Washington, Seattle WA 98195, USA

Abstract

Accurate recognition and tracking of human activities is an important goal of ubiquitous computing. Recent advances in the development of multi-modal wearable sensors enable us to gather rich datasets of human activities. However, the problem of automatically identifying the most useful features for modeling such activities remains largely unsolved. In this paper we present a hybrid approach to recognizing activities, which combines boosting to discriminatively select useful features and learn an ensemble of static classifiers to recognize different activities, with hidden Markov models (HMMs) to capture the temporal regularities and smoothness of activities. We tested the activity recognition system using over 12 hours of wearable-sensor data collected by volunteers in natural unconstrained environments. The models succeeded in identifying a small set of maximally informative features, and were able to identify ten different human activities with an accuracy of 95%.

1 Introduction

The task of modeling human activities from body-worn sensors has received increasing attention in recent years, especially in the ubiquitous computing (Ubicomp) field [Bao and Intille, 2004; Lukowicz *et al.*, 2004; Patterson *et al.*, 2003]. Although originally most of the research in activity recognition was done using vision sensors [Gavrila, 1999; Pentland, 1996], it has increasingly become dominated by various types of wearable sensors, like accelerometers and audio. A fertile application domain for activity recognition is in the health care arena, especially in elder care support, long-term health-monitoring, and assisting those with cognitive disorders. In addition, activity recognition is an important component for modeling higher level human behavior, tracking routines, rituals, and social interactions.

The majority of research using wearable devices, has concentrated on using multiple sensors of a single modality,

typically accelerometers on several locations on the body [Bao and Intille, 2004; Kern *et al.*, 2003]. The placement of sensors in multiple pre-defined locations can be quite obtrusive and is one of the limitations of such an approach. While the ultimate goal is to embed these devices into clothing, this technology is far from being commercially available and widely accepted. As a result, a single sensing device that can be integrated into existing mobile platforms, such as a cell phone, will be more appealing to users and is likely to garner greater user acceptance. Work by [Bao and Intille, 2004] has shown that an appropriate sensor subset (two locations), does not effect the recognition scores significantly (by less than 5%) compared to a system with five sensors; whereas the use of a single sensor reduced the average accuracy by 35%. Our hypothesis is that incorporating multiple sensor modalities will offset the information lost by using a single sensing device. Furthermore, multiple modalities will be better suited to record the rich perceptual cues that are present in the environment, cues that a single modality often fails to capture. Multiple modalities have already shown promise in earlier activity recognition experiments [Lukowicz *et al.*, 2002].

To capture the diverse cues from movement, sound, light, etc., about ongoing activities, we have built a very small sensing unit (2.53 sq. in.) that includes eight different sensors: accelerometer, audio, IR/visible light, high-frequency light, barometric pressure, humidity, temperature, and compass. Using these sensors, we have collected a large annotated dataset of various human activities from two volunteers over a period of six weeks. We compute over six hundred different features from these eight sensor modalities, which attempt to capture various attributes of the raw signal.

Often in activity recognition, the choice of sensors and the features derived from them are driven by human intuition and by what is easily available, rather than by performance or practicality. Using the right features is crucial for recognition. We are working towards developing a framework that allows us to systematically identify modalities and features that are most useful for machine recognition and discrimination of natural human activities.

In the end, we want models that accurately recognize and track a variety of activities and a system that is lightweight enough to run on devices like cell phones, which many people already carry. Thus, minimizing the computation cost of our recognition system is also an important goal.

The two main approaches that are used for classification in machine learning are: (i) generative techniques that model the underlying distributions of the classes and (ii) discriminative techniques that only focus on learning the class boundaries [Rubinstein and Hastie, 1997]. Both of these approaches have been used extensively in the vision and the wearable sensing communities for recognizing various human behavior and activities. The work presented in this paper is a hybrid approach that combines the two techniques. First, a modified version of AdaBoost proposed by [Viola and Jones, 2001], is used to automatically select the best features and to learn an ensemble of discriminative static classifiers for the activities we wish to recognize. Second, the classification margins from the static classifiers are used to compute the posterior probabilities, which are then used as inputs into HMM models. The discriminative classifiers are tuned to make different activities more distinguishable from each other, while the HMM layer on top of the static classification stage ensures temporal smoothness and allows us to continuously track the activities.

The rest of the paper is organized as follows: Section 2 provides an overview of the activity recognition system. Section 3 presents the feature selection and discriminative classifier training method. Section 4 describes how the results from the classifiers are combined with HMMs. Section 5 describes our experimental results and the performance of the system, and Section 6 discusses our conclusions and possible future directions

2 Activity Recognition System Overview

The first problem we address is the systematic identification of modalities and features that are well suited for accurate recognition of natural human activities. The second problem we tackle is how these features can be effectively used to develop models that accurately recognize and track various activities. Below we give a brief overview of the different components in our activity recognition system.

Sensing and Feature Extraction

Using a shoulder mounted multi-sensor board (Figure 1(A)), we collect approximately 18,000 samples of data per second. To reduce the dimensionality and to bring out details in the data we compute a total of 651 features; which include linear and mel-scale FFT frequency coefficients, cepstral coefficients, spectral entropy, band pass filter coefficients, integrals, mean and variances. We combine the features from various sensors to produce a 651 dimensional feature vector at 4Hz. However, since we have sensors with different sampling rates, there can be multiple instances of a feature within the 0.25 second window; that operate on

different portions of the data. Furthermore, when calculating some features (e.g. the integral features) we incorporate a longer time window that varies from several seconds to as long as a minute. We restrict the time windows to only use data from the past, so that our system functions without any latency.

Feature Selection and Discriminative Activity Models

Earlier work has shown that discriminative methods often outperform generative models in classification tasks [Ng and Jordan, 2002]. Additionally, techniques such as bagging and boosting that combine a set of weak classifiers can further improve accuracy, without over-fitting to the training data [Schapire, 1999]. [Viola and Jones, 2001] have shown that boosting can be used not only as a method for combining classifiers but also as a method for selecting discriminative features. We use their proposed approach to select only a fraction of the total features, and to train very simple ensemble classifiers to recognize a broad set of activities.

Capturing Temporal Regularities

The activities people perform have certain natural regularities and temporal smoothness, e.g. people do not abruptly switch back and forth between walking and driving a car; thus, the recent history can help in predicting the present. Using a sequence of posterior probabilities computed from the instantaneous classifiers, we train Hidden Markov Models (HMMs) that significantly improve the performance and smoothness of our recognition system. By incorporating the static classification results we overcome the weakness of HMMs as effective classifiers [Jaakkola and Haussler, 1999].

3 Selecting the Right Features

Given a rich set of sensor data and features, our classifiers will work best if we select the right features that enable the classifiers to discriminate well between classes, and if we remove features that are not useful or which might even confuse the classifiers. Although it might be possible to hand pick the optimal features for certain activities, this is not a viable solution when the number of activities become large or when the sensor signals are not intuitive. In such scenarios, automatic techniques for finding the right set of features become increasingly important. A practical activity-recognition system will use a minimal number of features and the simplest possible models needed for high accuracy.

3.1 Feature Selection and Activity Classification using Boosted Decision Stumps

In this paper, we assume that people engage in N different type of activities. Given the set of activities $\mathbf{A} = \{A^1, \dots, A^N\}$, we also assume that we have a set of training data for each of those activities. Each sample in the training set consists of a feature vector $\mathbf{f} = \{f_1, \dots, f_K\}$ extracted from the sensors. For each activity A^i we are

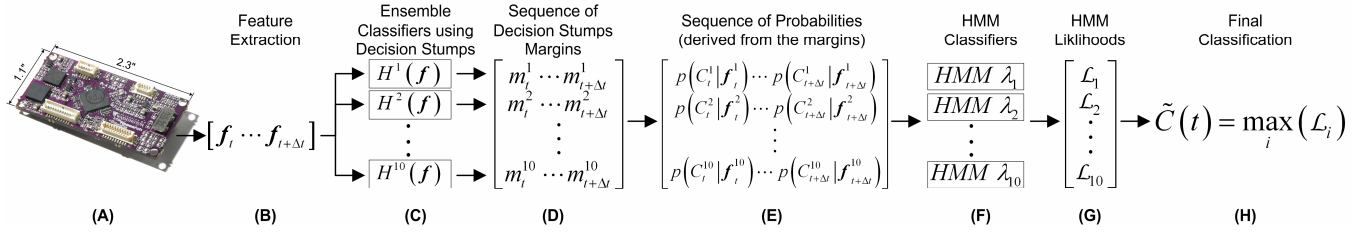


Figure 1: Flow diagram describing the classification system presented in this paper. (A) Our sensor board records a sequence of raw sensor recordings (B) from which we compute our feature vector. We pick the top fifty features per class, from our feature vector, and (C) supply them as inputs to our ensemble of decision stumps classifier. (D) Each decision stumps classifier outputs a margin at time t . (E) This sequence of margins can then be converted to probabilities by fitting them to a sigmoid. (F) The sequence of probabilities is then supplied to ten HMM classifiers (G) each of which outputs a likelihood. (H) The class with the highest likelihood is the classified class.

interested in finding a ranking of the feature set $\mathbf{R}^i = \{r_1^i, \dots, r_k^i\}$ based on their usefulness in recognizing activity A^i . Moreover, we want to find a cut-off point τ^i for the ranked feature set such that adding features beyond τ_i does not significantly improve the accuracy of the classifier C^i , i.e. $\Delta(\text{error}(C^i(f_1^i, \dots, f_{\tau^i}^i)), \text{error}(C^i(f_1^i, \dots, f_k^i))) \leq \epsilon$. The reason behind estimating τ^i is that, if $\tau^i \ll N$ then we can reduce the computational complexity of our classifiers by not extracting the less useful features. Since our final goal is to have the classifiers run on devices that users carry or wear, the computational costs of the classifiers are critical.

For each activity A^i , we iteratively train an ensemble of weak binary classifiers $H^i = \{h_1^i, \dots, h_N^i\}$ (Figure 1(C)) and obtain a ranking $\mathbf{R}^i = \{r_1^i, \dots, r_k^i\}$ for the features using the variation of the AdaBoost algorithm proposed by [Viola and Jones, 2001]. The weak classifiers are constrained to use only one feature, and at each iteration m of boosting we select the feature and the associated weak learner $h_m^i(f_m)$ that minimizes the training error $\epsilon_m^i(f_m)$ on the weighted data. The error $\epsilon_m^i(f_m)$ is used to re-weight the data for the next iteration and to compute the weight α_m^i for $h_m^i(f_m)$. At the end of this process, we have a ranking for the features based on how useful each feature is in discriminating A^i from the other activities A^j ($j \neq i$), and we also have a set of weak classifiers $h_m^i(f_m)$ and weights for those classifiers α_m^i . The final output is a weighted combination of the weak classifiers, and by estimating the error of C^i as a function of the number of features used, we can also find τ^i for C^i . So, for a given data point, the prediction of C^i is

$$H^i(\mathbf{f}) = \text{sign}(\sum_{m=1}^{\tau^i} \alpha_m^i h_m^i(f_m))$$

Each classifier C^i uses the top τ^i features, which is a fraction of the total number of features available, i.e. $\mathbf{f}^i = (f_1^i, \dots, f_{\tau^i}^i)$.

We tried two different weak classifiers in our system: (i) a discriminative decision-stump and (ii) a generative naïve Bayes model (conditional probability distributions are modeled using histograms that have 100 bins/dimension) as our weak classifier. In our experiments, the decision stump consistently outperformed the naïve Bayes classifier, so we

only use results from the decision-stump based classifier in the later sections. The decision stump finds the optimal threshold θ_m^i for each feature f_m that minimizes the weighted error such that $h_m^i(f_m) = 1$ if $f_m > \theta_m^i$ and $h_m^i(f_m) = -1$ otherwise.

For the boosted static classifiers, the classification margin for a data point can reflect the confidence in that prediction [Schapire *et al.*, 1997]. The margin of an example is the weighted fraction of the weak classifiers votes assigned to the correct class.

$$m^i(\mathbf{f}^i) = \frac{\sum_{m=1}^{\tau^i} \alpha_m^i h_m^i(f_m^i)}{\sum_{t=1}^{\tau^i} \alpha_m^i}$$

$$H^i(\mathbf{f}^i) = \text{sign}(m^i(\mathbf{f}^i))$$

However, constructing classifiers to output a posterior probability is very useful, especially if we want to combine the results of the multiple classifiers later on. One method of computing posterior probability directly is to fit a sigmoid function to the output of the ensemble classifier [Platt, 1999] (Figure 1(E)). In our case, the posterior probabilities are derived as follows -

$$p(C^i | \mathbf{f}^i) = \frac{e^{\phi m^i(\mathbf{f}^i)}}{e^{\phi m^i(\mathbf{f}^i)} + 1}, \text{ where } \phi \text{ is a constant}$$

A static classifier predicts the label for each data point independently. Most of the time this independence assumption is clearly invalid and the prediction of previous data points can help with the current classification. A temporal model that uses the confidence of the predictions of the classifiers C^i s instead of the raw features $\mathbf{f}^i$ is likely to have a greater impact on the performance. The ability to recognize activities in continuous time chunks would also allow us to learn how people transition between activities and thereby allow us to learn more about people's behavior and activity patterns. In the next section, we describe how we combine the confidence values of the static classifiers to build time-series models of the activities.

4 Incorporating Prediction History using Hidden Markov Models

HMMs have been successfully used in modeling different types of time-series data, e.g. in speech recognition, gesture

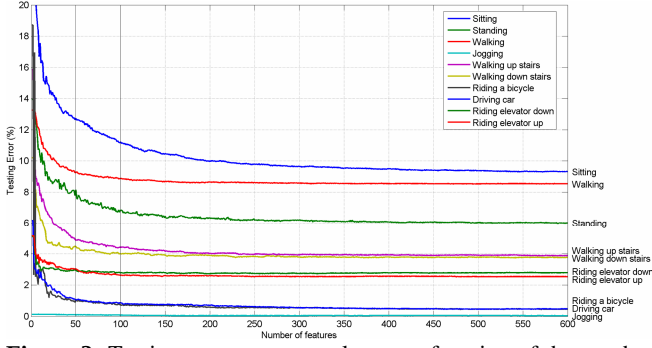


Figure 2: Testing error rates per class as a function of the number of features selected. After 50 features are selected most of the testing errors for the classes have leveled off. The data graphed here is averaged from several smaller feature selection runs.

tracking etc. We use HMMs to capture the temporal dynamics; but instead of directly using the raw features selected in the previous section, we trained our HMMs using the posterior probabilities of the static classifiers. The advantage of using the posterior probabilities is that we can take advantage of the results from the discriminatively trained classifier, as well as reduce the complexity of the HMMs. The earlier work of [Jaakkola and Haussler, 1999] has also shown the benefits of combining generative models into discriminative classifiers by deriving kernel functions from probability models. [Clarkson and Pentland, 1999; Oliver *et al.*, 2002] have used the output of an HMM model as input to another HMM. [Yin *et al.*, 2004] have used the output of static classifiers directly into HMMs for speech reading application; however they do not compute the margin or class posterior probability of these classifiers, which can be more effective than the raw outputs [Platt, 1999].

For each activity, a new HMM model λ_i (Figure 1(F)) is learned using a sequence of examples rather than individual instances. We construct a new input feature space based on the posterior class probabilities, which is

$$\tilde{\mathbf{f}} = \begin{bmatrix} p(C^1 | \mathbf{f}^1) \\ \vdots \\ p(C^N | \mathbf{f}^N) \end{bmatrix}$$

Given a set of observations $\{\tilde{\mathbf{f}}_1, \tilde{\mathbf{f}}_2, \dots, \tilde{\mathbf{f}}_T\}$ for each activity, we learn the parameters of the HMM λ_i using the standard expectation-maximization (EM) method. During testing, we have a continuous sequence S , which we use to compute the likelihood value $L_i(t)$ for λ_i at time t using a sliding window of duration Δt (Figure 1(G)) –

$$L_i(t) = P(\tilde{\mathbf{f}}_t, \tilde{\mathbf{f}}_{t+1}, \dots, \tilde{\mathbf{f}}_{t+\Delta t} | \lambda_i, S)$$

The final segmentation and classification is based on the HMM that has the highest likelihood value, i.e. $\tilde{C}_i(t) = \max_i L_i(t)$ (Figure 1(H)).

Accelerometer	37.7%
Ambient Light (IR-Vis)	2.5%
Audio	23.9%
Barometric Pressure	12.9%
Digital Compass	2.1%
Hi-Freq Vis Light	3.3%
IR Light	3.6%
Relative Humidity	4.1%
Temp. from Barometer	3.2%
Temp. from Relative Humidity	3.2%
Visible Light	3.3%

Table 2: The percentage of features from the top 50 that originated from the different sensors, averaged across all activity classes.

Alternatively, we could have trained the various states of a single HMM to recognize the different activity classes and to learn the transition characteristic between activities. We choose not to do that here as our activities are primitive and a single transition statistic is not very meaningful. However, we believe that the output of these HMMs could be used to train a single dynamic model of more complex behavior where the transition statistics would also be more informative.

5 Experiments

To validate our approach, we recorded 12 hours of data consisting of a large number of activities (such as sitting, walking, jogging, riding a bike, driving a car, etc.) using the wearable multi-sensor board. The dataset was collected across multiple days, by two volunteers (who are not the researchers) in various indoor and outdoor locations. The recordings were done in long stretches (one hour on average), where the duration of the activities themselves ranged from seconds (e.g. *entering a building*), to hours (e.g. *driving a car*). The volunteers were asked to go about performing a series of activities naturally without any specific constraints on the order, for example “go to building X and walk around inside”. To capture day-to-day variations in the activities we collected multiple instances of the various activities over the course of six weeks. On average we have about an hour of data per activity and around 100 instances per activity.

Feature Selection

For the feature selection stage, we selected 80% of the total data available for each class for training. Based on the training examples we derived a ranking R^i for the features for each activity individually using the boosted decision stump procedure described in section 3.1. Figure 2 shows the testing error as a function of the number of features used for classification. From the results we see that classification error tapers off at around $\tau^i = 50$ features for most classes. If we were to pick more features beyond the top 50 our performance only improves slightly with the testing error only improving $<1\%$ for 600 features. The practical advantage of features selection is that we can significantly reduce the computational burden on our resource constrained devices, without drastically affecting the

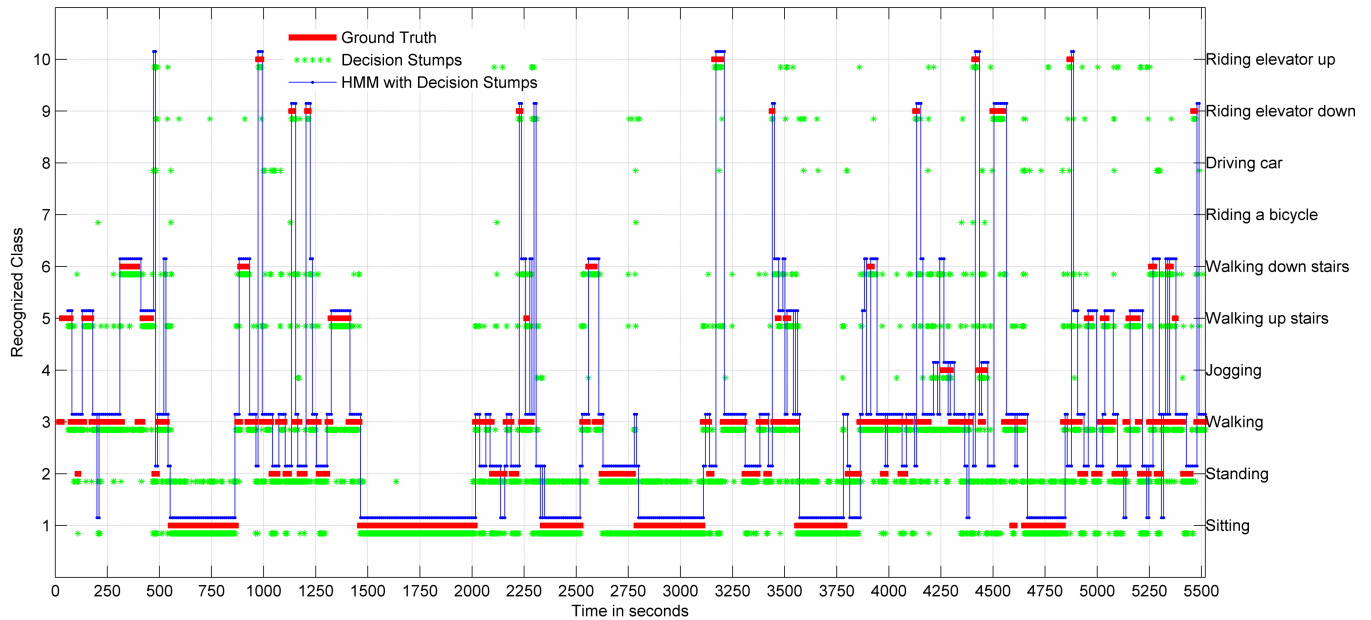


Figure 3: Output of the static decision stumps classifiers (at 4Hz), and the HMM classifiers (trained with output probabilities of the static classifiers) for a continuous 90 minute segment of the data. The results are overlaid on top of the ground truth which was obtained by annotating video recorded from a webcam worn by our volunteers. The video was only used for determining ground truth and not as an additional sensor input.

performance. Moreover, by performing boosting (re-weighting the data and selecting discriminatory features successively based on the error), we perform much better than taking a non-boosted (no re-weighting) approach to selecting the best 50 features. The accuracy ((true positive + true negative) / (total # of examples)) for the boosted features selection was on average ~11% higher than the non-boosted method. Table 2 lists the contribution of the different sensors to the final classifier. The majority of the top 50 features came from the accelerometer, audio and barometric pressure sensors. Barometric pressure data was useful in distinguishing activities that involved floor transitions (e.g. walking up/down stairs, elevator up/down); the sensor is sensitive enough to pick up pressure differences between a single floor.

Static Classification Results

Using the top 50 features we tested the performance of the ensemble classifier for two different weak classifiers – (i) decision stump (discriminative) and (ii) naïve Bayes (generative). The total duration of our test dataset was five and a half hours. The decision stumps outperformed the naïve Bayes classifiers by a large percentage. Table 3 shows the precision (true positive/(true positive + false positive)) and recall (true positive/(true positive + false negative)) numbers for the 10 activities in the dataset using the ensemble of decision stumps. Table 5 lists the average precision and recall numbers for the naïve Bayes as well as decision stump classifiers.

Continuous classification results

Although the decision stumps results from Table 3 are quite good on their own, Figure 3 illustrates the classification errors encountered for a continuous trace. The majority of the trace tends to be correctly classified by the decision stumps; but, with some scattered misclassifications. The addition of the HMM layer on top of the static classifier helps to smooth these classification errors as shown by the line in Figure 3.

The parameters of the HMMs were trained using 20 example scenes (on average 30 minutes of scenes) for each class. Each HMM had two hidden states and Gaussian observation probabilities. Classification was performed using a 15 second sliding window with 5 second overlap. Table 4 shows the sliding window performance results for the HMM, using the posterior probabilities as inputs, tested with concatenated test scenes. The overall accuracy in this case was 95%. It is interesting to note that the points in Figure 3 where the HMM and ground truth differ appear to be somewhat natural and realistic; for example, classifying a region without any ground truth between walking and sitting as standing. In fact, the HMM output reveals some deficiencies in the ground truth. For example, some segments whose ground truth was marked as walking are in fact standing (as determined by post-analysis of the video by the experimenters), and are correctly recognized as standing by the HMM.

To compare the performance to a more standard HMM approach, we trained a new set of HMMs that used the top 50 raw features as inputs rather than the output of the static classifiers. The performance of these HMMs was

		Classified Activity (by Decision Stumps)									
		Sitting	Standing	Walking	Jogging	Walking up stairs	Walking down stairs	Riding a bicycle	Driving car	Riding elevator down	Riding elevator up
Precision	Labeled Activities										
	Sitting	90.9%	43.3%	1.1%	0.3%	2.6%	2.7%	7.2%	10.2%	9.0%	5.6%
	Standing	7.1%	44.9%	0.3%		0.9%	0.3%	1.8%	0.7%	1.5%	1.5%
	Walking	1.2%	8.7%	95.1%	1.3%	21.1%	12.9%	5.4%	1.3%	0.8%	0.7%
	Jogging	0.0%		0.1%	98.3%		0.1%				
	Walking up stairs	0.0%	0.1%	1.9%		73.6%	0.7%	0.1%	0.0%		0.2%
	Walking down stairs		0.0%	1.4%	0.1%	1.0%	83.0%	0.1%		0.5%	
	Riding a bicycle	0.1%	0.1%	0.2%				85.3%			
	Driving car	0.5%	0.0%	0.0%		0.2%		0.1%	87.7%	0.1%	0.2%
Recall	Labeled Activities										
	Sitting	86.6%	10.0%	0.8%	0.0%	0.1%	0.1%	0.8%	1.3%	0.2%	0.1%
	Standing	38.2%	58.2%	1.3%		0.3%	0.1%	1.0%	0.5%	0.2%	0.2%
	Walking	1.6%	2.7%	92.6%	0.0%	1.5%	0.7%	0.7%	0.2%	0.0%	0.0%
	Jogging	0.1%		1.8%	97.7%		0.3%				
	Walking up stairs	0.1%	0.2%	26.1%		72.8%	0.6%	0.1%	0.1%		0.1%
	Walking down stairs		0.1%	22.7%	0.1%	1.2%	75.4%	0.3%		0.3%	
	Riding a bicycle	0.8%	0.3%	1.3%				97.6%			
	Driving car	4.1%	0.0%	0.1%		0.1%		0.1%	95.6%	0.0%	0.1%
		3.4%	14.4%			0.2%	0.2%		0.1%	81.3%	0.3%
		3.2%	10.0%	0.1%		1.0%			0.4%	0.4%	84.8%

Table 3: Precision and recall numbers for the decision stumps classifier. A randomly selected 20% set of data was set aside and used for the test results here. Overall accuracy is 91%.

		Classified Activity (by HMM)									
		Sitting	Standing	Walking	Jogging	Walking up stairs	Walking down stairs	Riding a bicycle	Driving car	Riding elevator down	Riding elevator up
Precision	Labeled Activities										
	Sitting	89.8%	38.5%	0.5%				0.4%	33.4%		
	Standing	10.1%	50.8%	1.4%							
	Walking	0.1%	7.4%	97.7%		5.2%	2.5%				
	Jogging				100.0%						
	Walking up stairs					94.8%					
	Walking down stairs			0.5%			97.5%				
	Riding a bicycle		3.3%					99.6%			
	Driving car								66.6%		
Recall	Labeled Activities										
	Sitting	87.5%	3.7%	0.1%				0.1%	8.6%		
	Standing	65.6%	32.8%	1.6%							
	Walking	0.4%	4.0%	93.8%		1.3%	0.4%				
	Jogging				100.0%						
	Walking up stairs					100.0%					
	Walking down stairs			2.5%			97.5%				
	Riding a bicycle		1.7%					98.3%			
	Driving car								100.0%		
										100.0%	
											100.0%

Table 4: Precision and recall numbers for the HMM classifier, using posterior probabilities as inputs. Overall accuracy is 95%.

	Naïve Bayes	Decision Stumps	HMM with Decision Stumps	HMM with Raw Features
Precision	92%	98%	99%	54%
Recall	45%	84%	91%	40%

Table 5: Overall precision and recall numbers for the various generative and discriminative classifiers, used in evaluating our system.

significantly worse, even worse than the static classifier, demonstrating the importance of discriminative classifiers in distinguishing between activities (see table 5 for a comparison of the overall precision and recall numbers for the various classifiers used in our experiments). We recognize that the modeling of more complex activities may require a generative model of personal behavior. However, we believe that discriminative classifiers that map sensor data into primitive activity classes will reduce a large amount of the sensor noise and allow us to learn complex behaviors more effectively.

6 Conclusion

The problem of recognizing human activities from sensor data presents diverse statistical challenges: different classes of actions need to be actively distinguished from each other; and a model needs to incorporate the fact that people's actions are extended over time. In the present study, we approached these problems by combining the discriminative power of an ensemble of decision stumps with the generative and temporal powers of HMMs.

As our presented results have shown, the combination of discriminative and generative classifiers is more effective than either of the classifiers on their own. Not only does the HMM on-top of the discriminative classifier perform better than the discriminative classifier on its own, but it also produces very smooth and accurate outputs as Figure 3 shows.

Feature selection plays an important role in our system not only in improving the performance of our classifier but also in creating a practical system. Our selection of the best 50 features for our top classes leaves us with 242 features from our original 651. This reduces the number of features necessary by more than 60%, which is a significant computational saving. This savings could be further improved upon by optimizing the calculation of the features to take advantage of the new subset, for example there are efficient techniques to obtain FFT coefficients only for required sub-bands [Goertzel, 1958]. In addition, this subset could be further reduced by allowing the feature selection process to determine a more optimal stopping point.

Thus, our work lays out several directions in which the automatic recognition of human actions can be pursued further. First, multi-modal wearable sensors provide a cheap, lightweight and unobtrusive means of obtaining richly detailed data in unconstrained environments, and over long periods of time. Second, the very richness of such sensor readings, and the mass of data collected, demand that suitable preprocessing and data-reduction techniques be applied. We found that by selecting the most informative features, computational costs could be cut by more than 60%, although still greater savings are probably attainable. Third, the multi-faceted nature of human activities presents opportunities for multiple machine-learning approaches to be combined, with the complementary strengths of different approaches (in this instance, boosted decision stumps and HMMs) meeting different aspects of the computational challenge. Our initial studies, presented here, have yielded high recognition rates, suggesting that this is a fruitful approach. Our future work will focus on incorporating and building on the techniques presented here to recognize more complex behavior patterns (e.g. *cooking*, *cleaning*, etc.) Further development of these ideas will, we hope, lead to activity recognition systems that can move from beyond the research lab out into the real world, offering applications in areas as diverse as smart rooms, ethnography, and health care for both the young and aging population.

Acknowledgments

The authors would like to thank Dieter Fox for the helpful comments and discussions. We also thank the two undergraduate students who collected many hours of data that was used our experiments.

References

- [Bao and Intille, 2004] L. Bao and S. Intille. Activity Recognition from User-Annotated Acceleration Data. *Proc. Pervasive*, 1-17, Vienna, Austria, 2004.
- [Clarkson and Pentland, 1999] B. Clarkson and A. Pentland. Unsupervised Clustering of Ambulatory Audio and Video. *Proceedings of the International Conference of Acoustics, Speech and Signal Processing*, Phoenix, AZ, 1999.
- [Gavrila, 1999] G. Gavrila. The visual analysis of human movement: A survey. *Computer vision and Image Understanding* 75(1), 1999.
- [Goertzel, 1958] G. Goertzel. An Algorithm for the Evaluation of Finite Trigonometric Series. *American Mathematical Monthly* 65, 1958.
- [Jaakkola and Haussler, 1999] T. Jaakkola and Haussler. Exploiting generative models in discriminative classifiers. *In Advances in Neural Information Processing Systems*, 1999.
- [Kern et al., 2003] N. Kern, B. Schiele, et al. Multi-Sensor Activity Context Detection for Wearable Computing. *Proc. EUSAI, LNCS*, 220-232, Eindhoven, The Netherlands, 2003.
- [Lukowicz et al., 2002] Paul Lukowicz, H. Junker, et al. WearNET: A Distributed Multi-sensor System for Context Aware Wearables. *Proceedings of the 4th international conference on Ubiquitous Computing*. Göteborg, Sweden, Springer-Verlag: 361-370, 2002.
- [Lukowicz et al., 2004] Paul Lukowicz, Jamie A. Ward, et al. Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. *Pervasive Computing: Proceedings of the 2nd International Conference*, 2004.
- [Ng and Jordan, 2002] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *In Advances in Neural Information Processing Systems*, 2002.
- [Oliver et al., 2002] N. Oliver, E. Horvitz, et al. Layered representations for human activity recognition. *Fourth IEEE Int. Conf. on Multimodal Interfaces*, 2002.
- [Patterson et al., 2003] D. Patterson, L. Liao, et al. Inferring high-level behavior from. *In the Proceedings of the Fifth Annual on Ubiquitous Computing*, Seattle, Washington, USA, 2003.
- [Pentland, 1996] A. Pentland. Smart Rooms. *Scientific American* 274(4):68-76, 1996.
- [Platt, 1999] J. Platt. Probabilities for SV Machines. *Advances in Large Margin Classifiers*. A. Smola, P. Bartlett, D. Scholkopf and D. Schuurmans, MIT Press, 1999.
- [Rubinstein and Hastie, 1997] Y. D. Rubinstein and T. Hastie. Discriminative vs. informative learning. *In the Proceedings of Knowledge Discovery and Data Mining*, 49-53, 1997.
- [Schapire, 1999] R. E. Schapire. A brief introduction to boosting. *Sixteenth International Joint Conference on Artificial Intelligence*, 1401-1405, 1999.
- [Schapire et al., 1997] Robert E. Schapire, Yoav Freund, et al. Boosting the margin: a new explanation for the effectiveness of voting methods, 322--330, 1997.
- [Viola and Jones, 2001] Paul Viola and Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *Computer Vision and Pattern Recognition*, 2001.
- [Yin et al., 2004] P. Yin, I. Essa, et al. Asymmetrically Boosted HMM for Speech Reading. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, pp II755-761, Washington DC, USA, 2004.