

AUTOMATED REASONING

Logical Filtering

Eyal Amir and Stuart Russell

Computer Science Division, University of California at Berkeley
Berkeley, CA 94720-1776, USA
{eyal,russell}@cs.berkeley.edu

Abstract

Filtering denotes any method whereby an agent updates its *belief state*—its knowledge of the state of the world—from a sequence of actions and observations. In *logical filtering*, the belief state is a logical formula describing possible world states and the agent has a (possibly nondeterministic) logical model of its environment and sensors. This paper presents efficient logical filtering algorithms that maintain a compact belief state representation indefinitely, for a broad range of environment classes including nondeterministic, partially observable STRIPS environments and environments in which actions *permute* the state space. Efficient filtering is also possible when the belief state is represented using prime implicants, or when it is approximated by a logically weaker formula.

1 Introduction

Any agent operating in a partially observable environment must perform computations that distinguish among the *a priori* possible current states of the world on the basis of past observations and actions. These computations may operate directly on a representation of the action–observation sequence (e.g., [Winslett, 1990; Kautz *et al.*, 1996]); they may reduce queries about the current state to queries about the initial state (e.g., [Reiter, 2001]); or, they may update the *belief state* (the agent’s knowledge about the state of the world) after each action and observation. This latter approach, called *filtering* or *recursive state estimation* in the control theory literature, is particularly useful with unbounded sequences of actions and observations.

The main computational difficulties are 1) the time needed to update the belief state, and 2) the space required to represent it. These depend on the nature of the *transition model*, which describes how the environment evolves over time, the *observation model*, which describes the way in which the environment generates observations, and the family of representations used to denote belief states. Early work, beginning with Gauss, assumed *stochastic* models. For example, the *Kalman filter* [Kalman, 1960] is a ubiquitous device that maintains a multivariate Gaussian belief state over n variables, assuming linear–Gaussian transition and observation

model. Crucially, the $O(n^3)$ update cost and the $O(n^2)$ space requirement *do not depend on the length of the observation sequence*; hence, a Kalman filter can run indefinitely. In this paper, we are interested in developing analogous results in the context of logical representations.

We adopt a simple logical language (Section 2) for describing the transition and observation models; the observations and the belief state itself are also logical formulae. The initial state may be only partially known; the transition model, which allows for actions by the agent itself, may be nondeterministic; and the observation model may be nondeterministic and *partial*, in the sense that the agent may not be able to observe the actual state.

Even when we restrict ourselves to propositional logic, it is clear that the general filtering problem is nontrivial (we prove it is computationally hard), because there are exponentially many possible states. We identify several classes of models that allow efficient filtering with respect to the belief-state representation size. Our primary method is based on decomposition theorems showing that 1) filtering distributes over disjunction in the belief state formula, and 2) filtering distributes over conjunction and negation if the actions are *permutations* of the state space. Such actions serve as one-to-one mappings between states, for those states in which they can be applied. We obtain efficient, exact algorithms for DNF belief states and for NNF (Negation Normal Form - all negations are in front of atoms) and CNF belief states with permuting actions. In other cases, we obtain efficient algorithms for *approximate filtering*.

In another class of dynamic systems, we can filter efficiently if the belief state is represented in CNF that includes all its prime implicants. Finally, we show that STRIPS models (possibly with nondeterministic effects of actions) also admit efficient filtering. The STRIPS assumption, that every action has no conditional effects and that an effect’s preconditions are the preconditions for the action’s execution, is key to this efficiency.

With respect to maintaining a compact representation, we show that properties similar to those mentioned above allow us to filter k -CNF formulae (CNF with clauses of at most k literals, when k is fixed) such that the result is represented in k -CNF (for the same fixed k). Thus, the belief state is maintained in $O(n^k)$ space indefinitely. In particular, we show mild conditions under which a compact belief state can be

maintained in nondeterministic STRIPS domains and in permutation domains. Finally, we show that DNF belief states remain compact if the effects of actions are deterministic and guaranteed to hold. These results are the first analogues, in the logical arena, of the desirable properties possessed by Kalman filters for continuous variables.

Ours is by no means the first work on filtering in a logical context. Early on, it was pointed out that filtering is easy for deterministic systems with a known initial state [Fikes *et al.*, 1972; Lin and Reiter, 1997]. Filtering in nondeterministic domains is more difficult. In particular, the related problem of temporal projection is coNP-hard when the initial state is not fully known, or when actions have nondeterministic effects [Liberatore, 1997] (see also Section 3.3).

Traditionally, computational approaches for filtering take one of three approaches: 1) enumerate the world states possible in every belief state and update each of those states separately, together generating the updated belief state [Ferraris and Giunchiglia, 2000; Cimatti and Roveri, 2000], 2) list the sequence of actions and observations and prove queries on the updated belief state [Reiter, 2001; Sandewall, 1994], or 3) approximate the belief state representation [Son and Baral, 2001; Williams and Nayak, 1996].

The first two approaches cannot be used when there are too many possible worlds (e.g., when the domain includes more than a few dozens of fluents and there are more than 2^{40} possible states) or when the sequence of actions is long (e.g., more than 100 actions). Examples include robot localization, tracking of objects and their relationships, and data mining. The third approach gives rise to many mistakes that are sometimes dangerous, and requires an approximation that fits the given problem (if one exists). Many domains of 100 fluents or less are still computationally infeasible for it.

2 Logical Filtering

In this section we define logical filtering using a transition model and action semantics that are compatible with the *standard semantics* belief update operator of [Winslett, 1990]. (To avoid confusion, this is different from another operator presented in the same publication, *PMA*, that applies a *non-monotonic* approach to formalize minimal change.) This operator is simple and allows us to examine computational properties easily. It can represent any logical transition system, and specifications in other action languages can be compiled into it [Winslett, 1990; Doherty *et al.*, 1998].

In what follows, for a set of propositional formulae, Ψ , $L(\Psi)$ is the signature of Ψ , i.e., the set of propositional symbols that appear in Ψ . $\mathcal{L}(\Psi)$ is the language of Ψ , i.e., the set of formulae built with $L(\Psi)$. Similarly, $\mathcal{L}(L)$ is the language of L , for a set of symbols L .

A transition system is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, where

- $\mathcal{P}$ is a finite set of propositional fluents;
- $\mathcal{S} \subseteq \text{Pow}(\mathcal{P})$ is the set of world states;
- $\mathcal{A}$ is a finite set of actions;
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation.

The intuition for this transition system description is that $\mathcal{P}$ is the set of features that are available for us in the world, every element in $\mathcal{S}$ is a *world state* (i.e., a subset of $\mathcal{P}$, containing

propositions that are true in this world state), $\mathcal{A}$ is the set of actions in the system and $\mathcal{R}(s, a, s')$ means that state s' is a possible result of action a in state s .

A *belief state* is a set of world states $\sigma \subseteq \mathcal{S}$. Performing an action a in a belief state σ results in a belief state that includes all the world states that may result from a in a world state in σ . We do not introduce *observations* in this transition model. Instead, we assume that observations are given to us (if at all) as logical sentences after performing an action.

A logical nondeterministic domain description D is a finite set of statements of the following kinds: *value propositions* of the form “**initially** F ” describe the initial state and *effect rules* of the form “ a **causes** F **if** G ” describe the effects of actions, for F and G being *state formulae* (propositional combinations of fluent names). We say that F is the *head* and G is the *tail* of those rules.

For a domain description D we define $\mathcal{P}_D, \mathcal{A}_D$ to be the set of propositional fluents and actions mentioned in D , respectively. For a domain description D we define a transition relation $\mathcal{R}_D(s, a, s')$ as follows.

- A fluent $f \in \mathcal{P}_D$ is *possibly affected* by action a in state s , if there is a rule “ a **causes** F **if** G ” in D such that G is true in s and $f \in L(F)$.
- Let $I(a, s)$ denote the set of fluents in $\mathcal{P}_D$ that are *not* possibly affected by action a in state s .
- Let $F(a, s)$ be a set of all the heads of activated effect rules in s (i.e., if “ a **causes** F **if** G ” is activated in s , then $F \in F(a, s)$). We consider the case of $F(a, s) = \emptyset$ (no activated effect rules) as $F(a, s) \equiv \text{TRUE}$.
- Define (recalling that world states are sets of fluents)

$$\mathcal{R}_D = \left\{ \langle s, a, s' \rangle \mid \begin{array}{l} (s' \cap I(a, s)) = (s \cap I(a, s)) \\ \text{and } F(a, s) \text{ is true in } s' \end{array} \right\}$$

When there is no confusion, we write $\mathcal{R}$ for $\mathcal{R}_D$.

If action a has an effect of *FALSE* in s , then it cannot execute.

In partially observable domains, we update our knowledge as a result of executing an action and collecting observations in the resulting state. The following definition of filtering assumes that σ is a set of world states. We use our transition operator $\mathcal{R}$ to define the resulting belief state from each action. An observation o is a formula in our language.

Definition 2.1 (Logical Filtering Semantics). Let $\sigma \subseteq \mathcal{S}$ be a belief state. The filtering of a sequence of actions and observations $\langle a_1, o_1, \dots, a_t, o_t \rangle$ is defined as follows:

1. $\text{Filter}[\epsilon](\sigma) = \sigma$;
2. $\text{Filter}[a](\sigma) = \{s' \mid \langle s, a, s' \rangle \in \mathcal{R}, s \in \sigma\}$;
3. $\text{Filter}[o](\sigma) = \{s \in \sigma \mid o \text{ is true in } s\}$;
4. $\text{Filter}[\langle a_i, o_i, \dots, a_t, o_t \rangle](\sigma) =$
 $\text{Filter}[\langle a_{i+1}, o_{i+1}, \dots, a_t, o_t \rangle]$
 $(\text{Filter}[o_i](\text{Filter}[a_i](\sigma)))$.

We call Step 2 progression with a and Step 3 filtering with o .

For example, consider a robot that is in charge of cleaning a room. It can execute an action $a = \text{fetch}(\text{broom}, \text{closet})$ which has the single effect rule “ a **causes** $\text{has}(\text{broom}) \wedge \neg \text{in}(\text{broom}, \text{closet})$ **if** $\text{in}(\text{broom}, \text{closet})$ ”. Assume that the robot’s belief state is $\sigma = \text{Pow}(\mathcal{P})$ (i.e., it considers all states possible). Then, $\text{Filter}[a](\sigma) = \{s \in$

$Pow(\mathcal{P}) \mid \neg in(broom, closet)$ is true in s }, i.e., after performing the action a we consider all worlds that satisfy $\neg in(broom, closet)$ possible. This is because if we are initially in a state in which $in(broom, closet)$, then the resulting state is one in which $\neg in(broom, closet)$, and if we are initially in a state in which $\neg in(broom, closet)$, then we stay in the same state (thus, still satisfying $\neg in(broom, closet)$). Call this resulting belief state σ' . Now, if an observation $o = has(broom)$ is received, then $Filter[o](\sigma')$ is exactly the set of worlds that satisfy $\neg in(broom, closet)$ and $has(broom)$.

3 Filtering Logical Formulae

Approaches to filtering actions and observations that at any stage enumerate the states in a belief state do not scale to large domains. An alternative approach is to perform logical progression in a form similar to the one described by [Lin and Reiter, 1997; McIlraith, 1998]. The difference is that now we wish to do so (efficiently) in the context of nondeterministic actions and observations.

In this section we present a straightforward algorithm that filters belief state formulae directly, but does so in worst-case exponential time. This algorithm serves as a starting point for Section 4, where we propose efficient algorithms. We also present distribution properties of filtering over the logical connectives $\wedge, \vee, \neg$, and examine the theoretical limitations of formula filtering. These will guide us in Section 4, and allow us to present classes of systems that are not subject to those limitations and can be tracked in polynomial time and a compact fashion indefinitely.

3.1 Zeroth-Order Filtering Algorithm

In the rest of the paper we assume that a fixed set of fluents persists for action a in all states in which it has an effect. $Eff(a)$ is the set of fluents possibly affected by a , and $\overline{Eff(a)}$ is $\mathcal{P} \setminus Eff(a)$. Some of the following notation assumes an implicit action, when this causes no confusion.

We represent a belief state σ as a logical formula φ such that a state is in σ iff it satisfies φ . The filtering of a belief state formula with an action and an observation is a formula representing the consequences of our effect rules and observation on states that satisfy our initial formula. We specify this filtering process formally using the following notation.

For a set of effect rules $r_1, \dots, r_l$ for action a , each of the form “ a causes F_i if G_i ”, write $F'_i = F_{i[\mathcal{P}/\mathcal{P}]}$, for $\mathcal{P}' = \{f'_1, \dots, f'_n\}$ a set of new symbols for fluents, $[\mathcal{P}/\mathcal{P}']$ a shorthand for $[f_1/f'_1, \dots, f_n/f'_n]$, and $[f/f']$ means that we replace all instances of symbol f in the formula by instances of symbol f' . Here, we view $\mathcal{P}$ as the set of fluents in some time t , and $\mathcal{P}'$ as the same fluents in time $t + 1$. We add the following set of rules for action a :

$$\begin{aligned} \mathcal{C} = \{ & \text{“}a \text{ causes } p \text{ if } p\text{”, “}a \text{ causes } \neg p \text{ if } \neg p\text{”} \mid p \notin Eff(a)\} \\ & \cup \{ \text{“}a \text{ causes } p \text{ if } p \wedge \bar{G}\text{”} \mid p \in Eff(a)\} \\ & \cup \{ \text{“}a \text{ causes } \neg p \text{ if } \neg p \wedge \bar{G}\text{”} \mid p \in Eff(a)\} \end{aligned}$$

where

$$\bar{G} = \neg G_1 \wedge \dots \wedge \neg G_l, \quad (1)$$

the assertion that no precondition of a holds. This has a similar effect to adding frame axioms to a set of effect axioms in

an action language. We let $r_1, \dots, r_m$ be the complete set of rules for a and call the new rules, $\mathcal{C} = \{r_{l+1}, \dots, r_m\}$, *completion rules* for a .

We filter a belief-state formula as follows. (We reuse $Filter\cdot$ for filtering a belief-state formula.) Let $Cn(\Psi)$ be the set of logical consequences of Ψ (i.e., formulae ψ such that $\Psi \models \psi$), and $Cn^{\mathcal{L}}(\Psi)$ be $Cn(\Psi) \cap \mathcal{L}$, 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. $Filter[a](\varphi) = Cn^{\mathcal{P}'}(\varphi \wedge \bigwedge_{i \leq m} ((\varphi \Rightarrow G_i) \Rightarrow F'_i))_{[\mathcal{P}'/\mathcal{P}]}$;
2. $Filter[o](\varphi) = \varphi \wedge o$.

For example, consider action $a = fetch(broom, closet)$ which has the single effect rule “ a causes $has(broom) \wedge \neg in(broom, closet)$ if $in(broom, closet)$ ”, and consider a belief state formula $\varphi = TRUE$ (i.e., we consider all states possible). Then, $Filter[a](\varphi) \equiv \neg in(broom, closet)$, i.e., after performing the action a we consider all worlds that satisfy $\neg in(broom, closet)$ possible (similar to our example in Section 2). Call this resulting belief state formula φ' . Now, if an observation $o = has(broom)$ is received, then $Filter[o](\varphi') \equiv \neg in(broom, closet) \wedge has(broom)$.

The following generalizes a theorem presented in [Doherty et al., 1998] by allowing conditional and inconsistent effects.

Theorem 3.1. *If φ is a belief state formula, and a an action, then*

$$Filter[a](\{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\}) = \{s \in \mathcal{S} \mid s \text{ satisfies } Filter[a](\varphi)\}$$

Our zeroth-order algorithm computes $Filter[(a_1, o_1, \dots, a_t, o_t)](\varphi)$ by iterating the application of filtering of a belief-state formula with an action and an observation. It sets $\varphi_0 = \varphi$ and $\varphi_i = Filter[o_i](Filter[a_i](\varphi_{i-1}))$ recursively for $i > 0$ using the two equalities defined above. This algorithm is correct, as shown by Theorem 3.1. It can be implemented using a consequence finder in a restricted language (e.g., [Simon and del Val, 2001]).

From here on, when we say *filtering* we refer to filtering of a belief-state formula, unless otherwise mentioned.

3.2 Distribution Properties and Permutation

We can decompose the filtering of a formula φ along logical connectives once we establish several distribution properties.

Theorem 3.2 (Distribution over Connectives). *Let a be an action, and let φ, ψ be formulae. Then,*

1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $\models Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$
3. $\models Filter[a](\neg \varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$

We can say something stronger for actions that act as *permutations* on the states in $\mathcal{S}$ in which they can be executed.

Definition 3.3 (Permuting Actions). *Action a is permuting, if for every state s' there is at most one s such that $\mathcal{R}(s, a, s')$.*

Domains that include only permuting actions are called *permutation domain*.

For example, consider $a = \text{fetch}(\text{broom}, \text{closet})$ from above, and assume that we add a second effect rule “ a **causes** FALSE if $\neg \text{in}(\text{broom}, \text{closet})$ ”. Thus, a is not executable unless its first rule’s precondition holds. Then, the action is a one-to-one mapping between states, when this mapping is defined (it is not defined when a state maps to no resulting state). If this second rule is not added, then the action is not one-to-one because it maps two different states (in the first we already have the broom and in the second the broom is in the closet) to the same state.

In the same spirit, an action $\text{pickUp}(A, B)$ that picks up block A from the top of block B is one-to-one when it is possible because we can find a single previous state for every resulting state. The same holds for $\text{putDown}(A, C)$. Other natural examples include *turning a row* in a Rubik’s cube, *flipping a light switch*, and *buying a gallon of gas*. In contrast, turning on the light, setting a Rubik’s cube to a particular configuration, and filling up the gas tank are *not permutation actions*. Notice that we allow *different* actions to map different states to the same state (e.g., accelerating by 5MPH when driving 40MPH results in the same state as when decelerating by 5MPH when driving 50MPH).

Theorem 3.4 (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. $\text{Filter}[a](\neg\varphi) \equiv \neg\text{Filter}[a](\varphi) \wedge \text{Filter}[a](\text{TRUE})$

Proof Sketch. Space permits inclusion only of the most interesting part, i.e., that of “ $\Leftarrow$ ” of 2.

Let s' be a world state that satisfies $\text{Filter}[a](\varphi) \wedge \text{Filter}[a](\psi)$. Then, it satisfies both of $\text{Filter}[a](\varphi)$, $\text{Filter}[a](\psi)$. For $\text{Filter}[a](\varphi)$ there is a state s such that $R_D(s, a, s')$ and s satisfies φ . Similarly, for $\text{Filter}[a](\psi)$ there is a state s_1 such that $R_D(s_1, a, s')$ and s_1 satisfies ψ . However, since a acts as a one-to-one mapping from $\mathcal{S}$ to $\mathcal{S}$, there is only one state in $\mathcal{S}$ that a maps to s' . Thus, $s = s_1$, and s satisfies ψ . Thus, s satisfies $\varphi \wedge \psi$ and s' satisfies $\text{Filter}[a](\varphi \wedge \psi)$. From Theorem 3.1 it follows that $\models \text{Filter}[a](\varphi \wedge \psi) \Leftarrow \text{Filter}[a](\varphi) \wedge \text{Filter}[a](\psi)$. $\square$

3.3 Limitations for Compact Representation

It may be argued that filtering may require only polynomial space, if we limit ourselves to initial belief states that are represented compactly and to actions whose effects and preconditions are represented compactly. In Theorem 3.5 we show the contrary. That is, for every general-purpose representation of belief states there is a dynamic system, an initial belief state, and a sequence of actions after which our belief state representation is exponential in the representation of the initial belief state.

P/poly is a *nonstandard* computational complexity class that includes problems that can be answered in polynomial time, if we are given a polynomial-length *hint* that depends only on the length of the input problem. It is considered likely that $\text{NP} \cap \text{co-NP} \not\subseteq P/\text{poly}$. Assume so, and let $f(\sigma)$ ($\sigma \subseteq 2^S$) encode belief states using n propositional symbols.

Theorem 3.5. *There is dynamic system D with n fluents, belief state σ_0 , and action sequence $a_1, \dots, a_n$ such that, for all*

$i \leq n$, $\sigma_i = \text{Filter}[a_i](\sigma_{i-1})$, and $|f(\sigma_n)| > \text{poly}(|f(\sigma_0)| \cdot |D| \cdot n)$. ($|D|$ is the representation size of D .)

The proof of this theorem reduces the problem of representing the belief state after performing an action to that of representing a *Craig Interpolant*. It uses the following.

Theorem 3.6 ([Boppa and Sipser, 1990]). *Assume that for every propositional implication $A \models B$ there is a Craig interpolant C such that $A \models C$ and $C \models B$ and $|C| \leq \text{poly}(|A| + |B|)$. Then $\text{NP} \cap \text{co-NP} \subseteq P/\text{poly}$.*

4 Efficient and Indefinitely Compact Filtering

In this section we present the main contribution of this paper, namely, a polynomial-time algorithm that computes logical filtering exactly for a significant class of transition systems. For some special cases we present simpler algorithms that are even more efficient. For systems that do not fall within this class our algorithm gives an approximation to the filtering. Also, we show that we can keep the representation of the filtered belief state compact indefinitely for a class of dynamic systems. This class includes nondeterministic STRIPS systems and some systems whose actions are permuting.

The theorems that we consider most important are the following three. Let $|\varphi|$ be the number of literals in φ ’s representation, let $\#rules(a)$ be the number of effect rules defining a , and let $\bar{G}_a = \bar{G}$ be the precondition of a as defined in (1). Assume that the effect rules that define a given action have identical sets of affected propositional symbols (however, different actions may have different such sets).

Theorem 4.1 (Efficiency). *Given NNF formula φ , action a , and observation o , the filtering algorithm in Figure 1 returns the filtering of φ with a and o , if a is permuting. If a is not permuting, then the algorithm returns an logically weaker formula than the filtering. It does so in time $O(|\varphi| \cdot 2^{\#rules(a) + L(\bar{G}_a)})$.*

Theorem 4.2 (Compact Filtering). *Given k -CNF formula φ (fixed k), deterministic action a , and observation o in k -CNF, the filtering of φ with a and o is in k -CNF, if a is permuting, $\text{Filter}[a](\text{TRUE})$ is in k -CNF, and for every literal ℓ in φ , $\text{Filter}[a](\ell) \equiv \text{Filter}[a](\text{TRUE}) \wedge T$, where T is a conjunction of literals.*

If a is not permuting, then the algorithm in Figure 1 still returns a formula in k -CNF.

The conditions on action a in the last theorem hold, e.g., for actions whose every defining rule has a precondition that is a single clause (e.g., a literal). It also holds for actions which are defined by at most two rules, and actions that affect all the literals that appear in their preconditions.

Finally, for nondeterministic STRIPS actions (actions that have no conditional effects) we get efficiency and compact representation. Let k -PI-CNF denote the class of k -CNF formulae that include all their prime implicates (i.e., φ in k -PI-CNF iff φ in k -CNF and for every clause C , if $\varphi \models C$, then there is a clause C' in φ such that C' subsumes C). Every formula can be represented in k -PI-CNF for some k .

Theorem 4.3 (STRIPS Actions). *Given k -PI-CNF formula φ (fixed k), STRIPS action a with effect in k -PI-CNF, and*

observation o in 2-CNF, the filtering of φ with a and o is computed in time $O(|\varphi| \cdot k + 2^{|L(\bar{G}_a)|})$. The result is in k -PI-CNF, if after observing o we know if a succeeded or failed.

4.1 Closed-Form Solution and Factored Filtering

Our zeroth-order filtering algorithm uses consequence finding tools which do not scale to large domains. The following always holds and suggests a different reasoning procedure.

$$\text{Filter}[a](\varphi) \equiv \bigwedge_{i_1, \dots, i_u \leq m, \varphi \models G_{i_1} \vee \dots \vee G_{i_u}} (F_{i_1} \vee \dots \vee F_{i_u}). \quad (2)$$

We can compute $\text{Filter}[a](\varphi)$ by testing queries of the form $\varphi \models G_{i_1} \vee \dots \vee G_{i_u}$ (instead of applying consequence finding). On the other hand, it requires an exponential number in m of such tests (recall, m is the number of rules, including the completion rules). Since $m > 2n$ (there are two completion rules for each of the n domain features), this is worse computationally than the method of enumerating all the states.

In what follows we use the intuition that we need only few rules if φ includes only a small subset of $\mathcal{P}$, the fluent symbols of our domain. In general, φ may include many fluent symbols because we may know many things about many different parts of our domain. Nevertheless, if we can decompose φ into small parts that can be filtered separately, then each of the parts includes only a small subset of $\mathcal{P}$, and filtering each of the parts separately becomes easy.

Assume that we order the rules of a such that $r_1, \dots, r_t$ ($t \leq l$) satisfy $L(F_i) \cap L(G_i) = \emptyset$ ($r_i = \text{"}a \text{ causes } F_i \text{ if } G_i\text{"}$, $i \leq t$), and $r_{t+1}, \dots, r_l$ satisfy $L(F_i) \cap L(G_i) \neq \emptyset$. Furthermore, let r_{m+1} be the additional rule " $a \text{ causes } \bar{G} \text{ if } \bar{G}$ " (recall, m is the number of rules of a , including completion rules). We define $\mathcal{B}$ to be

$$\mathcal{B} = \bigwedge_{i \leq t} (\neg G_i \vee F_i) \wedge \bigwedge_{i_1, \dots, i_u \in \{t+1, \dots, l, m+1\}} (\tilde{G}_{i_1, \dots, i_u} \vee \bigvee_{f \leq u} F_{i_f}) \quad (3)$$

where $\tilde{G}_{i_1, \dots, i_u} \equiv Cn^{\text{Eff}(a)}(\bigwedge_{f \leq u} \neg G_{i_f})$.

$\mathcal{B}$ is a term that is always implied by $\text{Filter}[a](\text{TRUE})$, i.e., the progression of zero knowledge with the action a . The first set of conjuncts of $\mathcal{B}$ is the result of applying a rule " $a \text{ causes } F_i \text{ if } G_i$ " whose preconditions are not affected by executing a . Even when we know nothing before performing a , we will know that either the effect occurred or the precondition did not hold and still does not hold. The second set of conjuncts applies a similar intuition for the case of effect rules that may affect the truth value of their original preconditions.

Define $\mathcal{C}(L)$ to be the set of completion rules of a for fluents in L , i.e., $\mathcal{C}(L) = \{i > l \mid \text{the effect of } r_i \in \mathcal{C} \text{ is in } L\}$.

We can now state the fundamental theorem of this Section. It holds for all domains expressed using our action language.

Theorem 4.4 (Closed-Form Representation). *If φ is a belief state formula and $\{r_1, \dots, r_l\}$ is the set of effect rules for action a , each of the form " $a \text{ causes } F_i \text{ if } G_i$ ", then*

$$\text{Filter}[a](\varphi) \equiv \bigwedge_{i_1, \dots, i_u \in \{1, \dots, l, m+1\} \cup \mathcal{C}(L(\varphi)), \varphi \models G_{i_1} \vee \dots \vee G_{i_u}} (F_{i_1} \vee \dots \vee F_{i_u}) \bigwedge \mathcal{B} \quad (4)$$

The intuition for equation (4) is that progressing φ with an action a can be computed by looking at all the possible combination of preconditions of effect rules and completion rules for $L(\varphi)$. If we can prove that $G_1 \vee G_2$ holds from φ , then we can conclude that $F_1 \vee F_2$ holds in the result of executing a . The conclusions that are not accounted for with this intuition are the effects that we infer from the completion rules in $\mathcal{C}(L(G_1, \dots, G_l))$ together with the effect rules for a . Those conclusions are summarized in $\mathcal{B}$, which is independent of φ .

PROCEDURE NNF-Filter ($\langle a_i, o_i \rangle_{0 \leq i \leq t}, \varphi$) $\forall i, a_i$ an action, o_i an NNF observation, φ a belief-state formula. 1. If $t = 0$, return φ . 2. Set $\mathcal{B}$ as in equation (3) for a_t but in NNF form. 3. Return $o_t \wedge \mathcal{B} \wedge \text{NNF-ProgressStep}(a_t, \text{NNF-Filter}(\langle a_i, o_i \rangle_{0 \leq i \leq t-1}, \varphi))$.
PROCEDURE NNF-ProgressStep (a, φ) a an action, φ a belief-state formula, $r_1, \dots, r_l, r_{m+1}$ rules for a . 1. If φ is a literal, then return the NNF form of $\bigwedge \{\bigvee_{i \in I} F_i \mid I \subseteq \{1, \dots, l, m+1\} \cup \mathcal{C}(L(\varphi)), \varphi \models \bigvee_{i \in I} G_i\}$. 2. If $\varphi = \varphi_1 \vee \varphi_2$, then return $\text{NNF-ProgressStep}(a, \varphi_1) \vee \text{NNF-ProgressStep}(a, \varphi_2)$. 3. It must be that $\varphi = \varphi_1 \wedge \varphi_2$. Return $\text{NNF-ProgressStep}(a, \varphi_1) \wedge \text{NNF-ProgressStep}(a, \varphi_2)$.

Figure 1: Filtering an NNF formula.

Our NNF filtering algorithm is presented in Figure 1. It is much faster than our zeroth-order algorithm, and it relies on Theorems 4.4, 3.2, and 3.4. In the following, if ψ is a logically weaker formula than our filtering, we say that it is a *safe approximation* for filtering. We denote effects by F_i and preconditions by G_i . For an action a , set $t = |\bigcup_{i \leq l} L(G_i)|$.

Corollary 4.5 (NNF Filtering). *Let φ be a formula in NNF with h literals, and let a be an action with l effect rules. Then, NNF-Filter safely approximates $\text{Filter}[a](\varphi)$ in time $O(h \cdot 2^{l+t})$. If a is permuting, then this computation is exact.*

Extended Example

As an example, consider our room-cleaning robot from above, and the action $a = \text{fetch}(\text{broom}, \text{closet})$, with the slightly more elaborate effect rule " $a \text{ causes } \text{has}(\text{broom}) \wedge \neg \text{in}(\text{broom}, \text{closet}) \text{ if } \text{in}(\text{broom}, \text{closet}) \wedge \neg \text{locked}(\text{closet})$ ". Assume that our robot has its belief state represented by $\neg \text{locked}(\text{closet}) \wedge (\text{in}(\text{broom}, \text{closet}) \vee \text{in}(\text{broom}, \text{shed}))$.

NNF-Filter filters each of the literals separately with a and combines the results. For a given literal, filtering is mostly done in step 1 of NNF-ProgressStep. We describe how this is done for each of the literals in our robot's belief state.

NNF-ProgressStep($a, \neg \text{locked}(\text{closet})$) tests in step 1 all possible entailments of the form $\neg \text{locked}(\text{closet}) \models \bigvee_{i \in I} G_i$, where I includes elements from the effect rule of a , the frame rule for $\bar{G}$ (rule r_{m+1}), and the completion rule for $\neg \text{locked}(\text{closet})$. For brevity, denote the effect of a by F_1 , and the precondition for that rule by G_1 . There is only one effect rule defining a , so in this case $\bar{G} = \neg G_1$.

In that step, one of our tests finds that

$$\neg \text{locked}(\text{closet}) \models G_1 \vee \bar{G}$$

(because $G_1 \vee \bar{G} = G_1 \vee \neg G_1 \equiv \text{TRUE}$). This makes us include $F_1 \vee \bar{G}$ in the filtering. In the same step, another test finds that $\neg \text{locked}(\text{closet}) \models \neg \text{locked}(\text{closet})$ (trivially true). This makes us include $\neg \text{locked}(\text{closet})$ in the filtering. The conjunction of the two is logically equivalent to $\neg \text{in}(\text{broom}, \text{closet}) \wedge \neg \text{locked}(\text{closet})$. This is the result of filtering $\neg \text{locked}(\text{closet})$ with a .

Similarly, for $\text{in}(\text{broom}, \text{closet})$, one of our tests in step 1 of NNF-ProgressStep finds that

$$\text{in}(\text{broom}, \text{closet}) \models G_1 \vee (\bar{G} \wedge \text{in}(\text{broom}, \text{closet}))$$

(because $G_1 \vee (\bar{G} \wedge \text{in}(\text{broom}, \text{closet})) \equiv G_1 \vee \text{in}(\text{broom}, \text{closet})$, and the latter follows trivially from $\text{in}(\text{broom}, \text{closet})$). As before, we also find that $\text{in}(\text{broom}, \text{closet}) \models G_1 \vee \bar{G}$. We find that the filtering of $\text{in}(\text{broom}, \text{closet})$ with a is $(\text{in}(\text{broom}, \text{closet}) \vee \text{has}(\text{broom})) \wedge (\neg \text{in}(\text{broom}, \text{closet}) \vee \text{locked}(\text{closet}))$. Notice that this belief state implies $\text{has}(\text{broom}) \vee \text{locked}(\text{closet})$.

For the filtering of the last literal, $\text{in}(\text{broom}, \text{shed})$, we find its filtering to be equivalent to $\text{in}(\text{broom}, \text{shed}) \wedge (\neg \text{in}(\text{broom}, \text{closet}) \vee \text{locked}(\text{closet}))$. (Notice, that we do not know that the broom is not in the closet because initially we did not know that the broom *cannot* be in more than one place; we made this choice to simplify this example.)

Now, we recurse back in the algorithm and combine the filtered formulae. We find that the filtering of $\text{in}(\text{broom}, \text{closet}) \vee \text{in}(\text{broom}, \text{shed})$ is equivalent to $(\text{in}(\text{broom}, \text{shed}) \vee \text{in}(\text{broom}, \text{closet}) \vee \text{has}(\text{broom})) \wedge (\neg \text{in}(\text{broom}, \text{closet}) \vee \text{locked}(\text{closet}))$. Notice that this belief state implies $\text{in}(\text{broom}, \text{shed}) \vee \text{has}(\text{broom}) \vee \text{locked}(\text{closet})$.

Finally, for the original belief state, $\neg \text{locked}(\text{closet}) \wedge (\text{in}(\text{broom}, \text{closet}) \vee \text{in}(\text{broom}, \text{shed}))$, we conjoin the last result with the filtering of $\neg \text{locked}(\text{closet})$ and get

$$(\text{in}(\text{broom}, \text{shed}) \vee \text{has}(\text{broom})) \wedge \neg \text{in}(\text{broom}, \text{closet}) \wedge \neg \text{locked}(\text{closet})$$

4.2 DNF and CNF Belief States

If φ is in DNF (a disjunction of conjunctions), then we can limit the size of the resulting formula. We write $D \wedge a \models \psi$ to say that action a has effect rules in D that logically imply ψ in a state in which a is executed (e.g., $\neg \bar{G}$ may be a tautology).

Corollary 4.6 (Iterating DNF Filtering). *Let φ be in DNF with h literals and s disjuncts, and a an action with l effect rules with $\{F_i\}_{i \leq l}$ in DNF with d disjuncts total, and $\{G_i\}_{i \leq l}$ in CNF, each with c conjuncts. Then, $\text{Filter}[a](\varphi)$ in DNF is computed exactly in time $O(h \cdot 2^{l+t})$ with at most $\max(2s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts. If $D \wedge a \models \neg \bar{G}$, then it has at most $\max(s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts.*

Thus, when a is a *deterministic* action (every rule's effect is a conjunction of literals) with a single effect rule that is always guaranteed to succeed, then the number of disjuncts in the formula does not grow as the filtering progresses.

For CNF formulae we can find a more significant class of actions that allow us to maintain compact representation. We show that under some conditions every k -CNF formula is filtered into a k -CNF formula (fixed k). This implies that the

belief state representation is no larger than $(2n)^k$, which is manageable for small fixed k 's.

The main observation that we use is that a clause of k literals may give rise to a larger clause after filtering, only if one of the following holds: (a) the filtering of TRUE includes a clause of more than k literals; or (b) the filtering of a literal includes a clause of 2 or more literals that is not subsumed by $\text{Filter}[a](\text{TRUE})$. The first case can occur when we do not know whether the action succeeded or not, and which rules applied if it did. In that case, we know that after the action one of the effects holds or no precondition holds (this yields a formula which may include many disjunctions). The second case can occur when the precondition of a rule includes a conjunction of literals. When we filter a single literal we may get a disjunction in the result (of the form *the effect holds, or the rest of the precondition does not*). When we filter a clause, this may cause the filtering to include a larger clause.

The following theorem describes sufficient conditions for filtering a k -CNF formula into k -CNF, thus keeping the representation compact (k is fixed).

Theorem 4.7 (Filtering a k -CNF Clause). *Let C be a k -CNF clause, and action a have l effect rules, all deterministic. Assume that $\mathcal{B}$ is in k -CNF, and that whenever $f \models \bigvee_{i \in I} G_i$ for literal f and $I \subset \{1, \dots, l, m+1\}$, $|I| \geq 2$, then $D \wedge a \models \bigvee_{i \in I} G_i$ or $f \models G_i$ for some $i \in I$. Then, $\text{Filter}[a](C)$ is in k -CNF and can be computed in time $O(2^{l+t})$.*

Note that $\mathcal{B}$ is in k -CNF for an action a when $|\bigcup_{i \leq l} L(G_i) \cup \text{Eff}(a)| \leq k$ or $|\bigcup_{i \leq l} L(G_i) \setminus \text{Eff}(a)| + l \leq k$ or $l = 1$ and $|L(G_1)| - \min(1, |L(F_1) \setminus L(G_1)|) \leq k$.

Consequently, filtering with actions that are permuting maintains a k -CNF if $\models \bigvee_{i \in I} G_i$ whenever $f \models \bigvee_{i \in I} G_i$ for some $|I| \geq 2$. An example of such an action is *flipping a switch* (if the light is on, it will get turned off, and vice versa). One of the preconditions always holds. Another example is moving a block (whichever it is) from the top of a stack. For literal f , if $f \models \text{top}(A) \vee \text{top}(B)$, then the disjunction is implied by $D \wedge a$ or a single precondition follows from f .

Corollary 4.8 (Iterating CNF Filtering). *If φ is in k -CNF, then the assumptions of Theorem 4.7 imply that $\text{Filter}[a](\varphi)$ is approximated safely in time $O(|\varphi| \cdot 2^{l+t})$, with a result in k -CNF. If a is permuting, then this computation is exact.*

4.3 Prime-Implicate Belief States

It turns out that a form of distribution over conjunction holds for all actions if the belief state is represented as the conjunction of all of its *prime implicates* (formulae we call *prime implicate belief states* (PI-CNF)). In this form we can distribute the computation to the conjuncts and conjoin the result of filtering small subgroups of them separately. More precisely,

$$\text{Filter}[a](\varphi) \equiv \bigwedge_{j_1, \dots, j_z \leq s} \text{Filter}[a](\bigwedge_{g \leq z} C_{j_g}) \quad (5)$$

for z a number that depends on the representation of the preconditions of a and on the number of rules defining a .

Theorem 4.9 (Filtering Prime Implicates). *Let φ be in k -PI-CNF (PI-CNF and k -CNF), and let action a have l effect rules with effects in k -CNF and preconditions in d -CNF, with*

PROCEDURE PI-Filter ($\langle a_i, o_i \rangle_{0 \leq i \leq t}, \varphi$) $\forall i, a_i$ an action, o_i an observation, φ a belief-state formula. 1. If $t = 0$, return φ . 2. Return the PI-CNF form of $o_t \wedge \text{PI-ProgressStep}(a_t, \text{PI-Filter}(\langle a_i, o_i \rangle_{0 \leq i \leq (t-1)}, \varphi))$.
PROCEDURE PI-ProgressStep (a, φ) a an action, φ a belief-state formula. $r_1, \dots, r_l, r_{m+1}$ rules for a . 1. Let $res \leftarrow \text{TRUE}$. 2. For every $i_1, \dots, i_j \in \{1, \dots, l, m+1\}$, and $f_1, \dots, f_u$ literals in $\text{Pre}(a) \setminus \text{Eff}(a)$, do (a) Let $\bigwedge_{g \leq z} \hat{C}_g$ be the CNF representation of $\bigvee_{h \leq j} G_{i_h}$. (b) For every $g \leq z$, nondeterministically choose a clause C_g in φ whose restriction to $\text{Pre}(a)$ subsumes $\hat{C}_g$. (c) If there is a clause for every $g \leq z$, then set $res \leftarrow res \wedge F$, where F is $\bigvee_{h \leq j} (F_{i_h} \vee (C_{i_h} \cap \text{Eff}(a)))$. 3. Return res .

Figure 2: Filtering a Prime-Implicate CNF formula.

each G_i of at most c clauses. Then, equation (5) holds for $z = c^l \cdot (d^c + 1)$, and **PI-Filter** (Figure 2) computes $\text{Filter}[a](\varphi)$ exactly in time $O(2^{l \cdot |\text{Pre}(a) \cup \text{Eff}(a)|} \cdot (s^z + z))$. If $D \wedge a \models \neg \bar{G}$, then $z = c^l$, with $O(2^l \cdot (s^z + z))$ time.

Corollary 4.10 (Maintaining k -PI-CNF). *Let φ be a k -PI-CNF formula, and let action a have l effect rules, all deterministic. Assume that G_i is a disjunction of literals, for all $i \leq l$. Also, assume that $D \wedge a \models \neg \bar{G}$. Then, $\text{Filter}[a](\varphi) \equiv \text{PI-ProgressStep}(a, \varphi)$, and the latter is in k -CNF. If $k = 2$, then $\text{PI-Filter}(a, \varphi)$ is in k -PI-CNF.*

The conclusion for $k = 2$ uses the fact that every prime implicate of a formula in 2-CNF is a clause with at most two literals. Simple counter examples (omitted) show that k -PI-CNF cannot be maintained without these conditions.

4.4 Nondeterministic STRIPS Domains

STRIPS domains present a special case of the results that we discussed above. In such domains every action has a single rule (no conditional effects) and actions can be executed only when their preconditions hold¹. Unlike the original STRIPS, we allow nondeterministic effects, and allow belief states to be any CNF formulae in the fluents of the domain.

More precisely, every action a has exactly two effect rules, r_1, r_2 . Their preconditions are such that $G_1 \equiv \neg G_2$. Also, $F_2 \equiv \text{FALSE}$. Thus, a can be executed only when G_1 holds. Consequently, when we filter with a we assert implicitly that its preconditions held in the last world state.

The assumption that there is only one rule that determines a 's effects and otherwise the action is not executed has a dramatic effect. For $l_1, \dots, l_k$ literals we get that

$$\text{Filter}[a](l_1 \vee \dots \vee l_k) \equiv \begin{cases} T_a & \exists i \leq k \ l_i \in \mathcal{L}(\text{Eff}(a)) \\ T_a \wedge \bigvee_{i \leq k} l_i & l_1, \dots, l_k \notin \mathcal{L}(\text{Eff}(a)) \end{cases} \quad (6)$$

¹With the alternate assumption that actions have no effect unless their preconditions hold, but observations (or their absence) are guaranteed to distinguish actions' success from failure, the same results hold, except that now the compactness of representation remains only after filtering with observations.

Theorem 4.11 (Iterating STRIPS Filtering: CNF). *Let φ be in k -CNF with s clauses and a a STRIPS action. If F_1 is in k -CNF and $|L(G_1) \setminus L(F_1)| \leq t$, for some $t \leq k$, then $\text{Filter}[a](\varphi)$ can be approximated safely in time $O(s \cdot k + 2^t)$, yielding a k -CNF formula. If a is permuting, then this computation is exact.*

If our representation is PI-CNF, then we get an even stronger result, leading to the algorithm in Figure 3.

Theorem 4.12 (Factoring STRIPS filtering: PI-CNF). *Let $\bigwedge_{i \leq s} C_i$ be in PI-CNF, and let a be a STRIPS action. Then,*

$$\text{Filter}[a](\bigwedge_{i \leq s} C_i) \equiv \bigwedge_{i \leq s} \text{Filter}[a](C_i).$$

For example, for $a = \text{fetch}(\text{broom}, \text{closet})$ that has effect $F_1 = \text{has}(\text{broom}) \wedge \neg \text{in}(\text{broom}, \text{closet})$, if we know $(\text{in}(\text{broom}, \text{closet}) \vee \neg \text{locked}(\text{closet})) \wedge (\text{in}(\text{broom}, \text{shed}) \vee \text{locked}(\text{closet}))$ before applying a , then after it we know $F_1 \wedge (\text{in}(\text{broom}, \text{shed}) \vee \text{locked}(\text{closet}))$.

Corollary 4.13 (Iterating STRIPS filtering: k -PI-CNF). *Let φ be in k -PI-CNF, and let a be a STRIPS action with F_1 in k -PI-CNF, $t = |L(G_1) \setminus L(F_1)|$, and $t \leq k$. Then, $\text{STRIPS-Filter}(a, \varphi)$ computes $\text{Filter}[a](\varphi)$ exactly in time $O(|\varphi| \cdot k + 2^t)$, yielding a k -PI-CNF formula.*

This means that we can filter in practice any prime implicate belief state in any nondeterministic STRIPS domain. This filtering stays compact, with the size depending only on the PI-CNF representation of F_1 and the number of propositional symbols in G_1 but not F_1 .

Finally, every STRIPS action can be filtered efficiently, even if we drop the assumption that either the action succeeds or we observe an error. This can be done by assuming that the action succeeds, finding the filtering of that action, and then disjoining the result with the initial belief state. Nonetheless, this scheme may cause representation space explosion as filtering proceeds over multiple steps, unless some care is taken.

PROCEDURE STRIPS-Filter ($\langle a_i, o_i \rangle_{0 \leq i \leq t}, \varphi$) $\forall i, a_i$ an action, o_i an observation, φ a belief-state formula. 1. For i from 1 to t do, (a) Set $\varphi' \leftarrow \bigwedge_{C \in \varphi} \text{Filter}[a_i](C)$, where $\text{Filter}[a_i](C)$ is computed using (6), and eliminate subsumed clauses. (b) Let φ be the prime implicates of $\varphi' \wedge o_i$. 2. Return φ .
--

Figure 3: Filtering a PI-CNF formula with STRIPS actions.

We tested our STRIPS-filter algorithm in partially observable blocks-world domains. The implementation in LISP includes a random action-sequence and observation-sequence generator, and both the generator and filtering algorithm receive a description of the domain, actions and observations specified in PDDL (a plan-problem specification language).

We ran the algorithm with blocks-world domains of increasing sizes (3 to 20 blocks), yielding domains that range from tens to over a thousand of propositional features. We collected the time taken per filtering step for each of the executions and the space taken overall at every iteration, starting with zero knowledge. The results are shown in Figure 4.

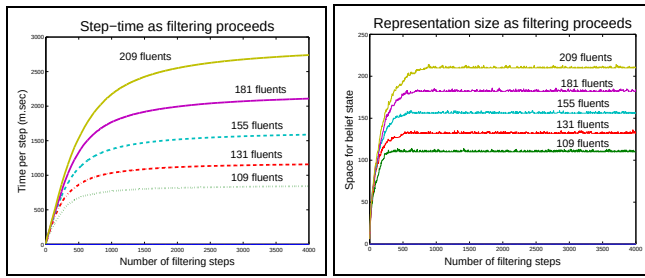


Figure 4: STRIPS-Filter in the blocks world.

4.5 Observation Model

An observation model is a theory $\mathcal{O}$ that includes *observation constraints*, axioms that describe the relationship between observed facts and other fluents. We allow $\mathcal{O}$ to include any axioms. Our model assumes that observations o are collected after an action is executed. o is a sentence made up with fluents, similar to the observation constraints. The conjunction, $\mathcal{O} \wedge o$ is then used to filter the resulting belief state. Formally,

$$\text{Filter}[o](\varphi) = \varphi \wedge o \wedge \mathcal{O}. \quad (7)$$

This allows all the results that we enumerated above to apply with this model as well. The following connects the results of Sections 3,4 to filtering with observations.

Corollary 4.14. *If $o \wedge \mathcal{O}$ is in k -CNF and $\text{Filter}[a](\varphi)$ is in k -CNF, then $\text{Filter}[a, o](\varphi)$ is in k -CNF.*

In particular, this means that our results for STRIPS domains and for NNF, CNF and DNF formulae still hold here. Thus, the representation remains compact in the cases that we indicated already, and computation remains easy in the same cases as well. Finally, for k -PI-CNF, we get the following.

Corollary 4.15 (Obs. and k -PI-CNF). *If $o \wedge \mathcal{O}$ is in 2-CNF, and φ is in k -PI-CNF, then $\text{Filter}[o](\varphi)$ is in k -PI-CNF.*

5 Conclusions

In this paper we presented the task of logical filtering and gave it a computational treatment. The results we obtained here have implications for monitoring and controlling dynamic systems. In many cases we present a closed-form computation of the filtering and in others show how to approximate this computation. In some cases we can guarantee that the size of the representation of the filtered formula can be bounded and kept small. In those cases, logical filtering can be used to control processes that run over very long periods of time. Examples of such systems are abundant and include robot motion control, natural language processing, and agents that explore partially observed worlds.

We made use of several assumptions in this paper in different contexts and with different consequences. We presented permutation domains and the certainty of existence of an effect ($D \wedge a \models \neg \bar{G}$) as characteristics of the domain that make filtering easier. We showed that the commonly used assumption that every action has a relatively small number of rules (at most polynomial in n), and that effects, preconditions and terms in the belief state typically use a small vocabulary, all

have a drastic effect on the computational effort needed for filtering and on the size of the resulting belief state.

The need to track the state of the world is a basic one, and many works have appealed to it implicitly in the past. However, the computational treatment of such tracking has been avoided so far, partially due to the absence of a developed theory of nondeterministic domains, and partially due to negative results about the general cases of this task. Nonetheless, this problem and methods for its solution have received much attention in control theory. The results we obtained here promise to find their application in this domain and may be combined with stochastic filtering techniques.

Acknowledgments

This research was supported by ONR MURI Fund N00014-01-1-0890, ONR MURI Fund N00014-00-1-0637, and NSF grant ECS-9873474. The first author thanks Xuanlong Nguyen for a stimulating discussion on Theorem 3.4.

References

- [Boppana and Sipser, 1990] R. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 757–804. Elsevier, 1990.
- [Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *JAIR*, 13:305–338, 2000.
- [Doherty et al., 1998] P. Doherty, W. Lukasiewicz, and E. Madalinska-Bugaj. The PMA and relativizing change for action update. In *Proc. KR '98*, pages 258–269, 1998.
- [Ferraris and Giunchiglia, 2000] P. Ferraris and E. Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proc. AAI '00*, pages 748–753, 2000.
- [Fikes et al., 1972] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. *AIJ*, 3:251–288, 1972.
- [Kalman, 1960] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. of ASME – J. of Basic Engineering*, 82(Ser. D):35–45, 1960.
- [Kautz et al., 1996] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proc. KR '96*, 1996.
- [Liberatore, 1997] P. Liberatore. The complexity of the language A. *ETAI*, 1:13–38, 1997.
- [Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *AIJ*, 92:131–167, 1997.
- [McIlraith, 1998] S. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. KR '98*, pages 167–177, 1998.
- [Reiter, 2001] R. Reiter. *Knowledge In Action*. MIT Press, 2001.
- [Sandewall, 1994] E. Sandewall. *Features and Fluents*. Oxford, 1994.
- [Simon and del Val, 2001] L. Simon and A. del Val. Efficient consequence-finding. In *IJCAI '01*, pages 359–365, 2001.
- [Son and Baral, 2001] T. C. Son and C. Baral. Formalizing sensing actions: A transition function based approach. *AIJ*, 125, 2001.
- [Williams and Nayak, 1996] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. AAI '96*, pages 971–978, 1996.
- [Winslett, 1990] M. Winslett. *Updating Logical Databases*. Cambridge U. Press, 1990.

A Tractability Result for Reasoning with Incomplete First-Order Knowledge Bases

Yongmei Liu and Hector J. Levesque

Department of Computer Science

University of Toronto

Toronto, ON, Canada M5S 3G4

{yliu, hector}@cs.toronto.edu

Abstract

In previous work, Levesque proposed an extension to classical databases that would allow for a certain form of incomplete first-order knowledge. Since this extension was sufficient to make full logical deduction undecidable, he also proposed an alternative reasoning scheme with desirable logical properties. He also claimed (without proof) that this reasoning could be implemented efficiently using database techniques such as projections and joins. In this paper, we substantiate this claim and show how to adapt a bottom-up database query evaluation algorithm for this purpose, thus obtaining a tractability result comparable to those that exist for databases.

1 Introduction

As argued in [Levesque, 1998], there is only one deductive technique efficient enough to be feasible on knowledge bases (KBs) of the size seemingly required for common-sense reasoning: the deduction underlying classical database query evaluation. And yet, databases by themselves are too restricted to serve as the representational scheme for common-sense reasoning, since they require, among other things, complete knowledge of the domain. Levesque proposed a generalization of databases called *proper knowledge bases*, which allow for a limited form of incomplete knowledge. Despite the limitations, however, the deduction problem for proper KBs is no longer even decidable. Levesque proposed an alternative reasoning procedure V for proper KBs that was logically sound and, when the query was in a certain normal form, logically complete. Moreover, he argued that it should be possible to implement V for very large KBs using database techniques. However, no proof was given.

In this paper, we examine proper KBs and the V procedure more closely, and we prove a tractability result for this type of logical reasoning that are comparable to those that exist for classical database query evaluation. In particular, we adapt a bottom-up database retrieval algorithm to the case of proper KBs. Thus, what we show here is that in some cases it is indeed possible to reason efficiently with incomplete knowledge in a logically sound and complete way, or at least as efficiently as we can with a database.

The rest of the paper is organized as follows. In the next section, we review proper KBs and V , prove a new property of V , *i.e.* locality, and define answers to open queries. In Section 3, we review the complexity of database query evaluation, and present a polynomial time algorithm for evaluating k -guarded formulas. In Section 4, we show how to use this algorithm to evaluate queries wrt proper KBs and hence obtain a tractability result. In Section 5, we illustrate this query evaluation method for proper KBs with some example queries. Finally in Section 6, we describe some future work.

2 Proper KBs and V

2.1 Basic definitions

We use a standard first-order logical language $\mathcal{L}$ with a countably infinite set of constants $\mathcal{C} = \{c_1, c_2, \dots\}$, no other function symbols, and a distinguished equality predicate. We have in mind that in any interpretation, $=$ should be understood as identity, that the constants will be unique names. This can be captured by a set of axioms $\mathcal{E}$, consisting of the axioms of equality and the set of formulas $\{c_i \neq c_j \mid i \neq j\}$.

In what follows, ρ ranges over atoms (excluding equality) whose arguments are distinct variables; e ranges over *ewffs*, *i.e.* quantifier-free formulas whose only predicate is equality. $\forall\alpha$ denotes the universal closure of formula α ; α_d^x denotes α with all free occurrences of x replaced by constant d . The *width* of α is the maximal number of free variables in its subformulas. The *quantifier rank* $qr(\alpha)$ of α is the depth of nesting of quantifiers in α .

Definition 2.1 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 $\forall(e \supset \neg\rho)$.

To illustrate the idea of a proper KB, imagine a scenario involving a robot security guard keeping track of the occupants of the rooms of a building. The robot can find out that someone has entered or left a room, and by going into a room, find out who the occupants are. We can express what the robot might know using a proper KB:¹

- Alice is alone in Room 1.
 $\forall xy. x = \text{alice} \wedge y = \text{room1} \supset \text{In}(x, y)$
 $\forall xy. x = \text{alice} \wedge y \neq \text{room1} \supset \neg \text{In}(x, y)$
 $\forall xy. x \neq \text{alice} \wedge y = \text{room1} \supset \neg \text{In}(x, y)$

¹We think of these as beliefs about some initial state of the world. It should be possible to generalize the notion of proper KB to deal with state change. See Section 6 for discussion.

- Bob and possibly others are in Room 2.
 $\forall xy. x = \text{bob} \wedge y = \text{room2} \supset \text{In}(x, y)$
 $\forall xy. x = \text{bob} \wedge y \neq \text{room2} \supset \neg \text{In}(x, y)$
- Carol is not in the building.
 $\forall xy. x = \text{carol} \supset \neg \text{In}(x, y)$
- Only Bob and Carol go into Room 3.
 $\forall xy. x \neq \text{bob} \wedge x \neq \text{carol} \wedge y = \text{room3} \supset \neg \text{In}(x, y)$

Note that this information cannot be expressed in a traditional database where, *e.g.* we cannot leave open the occupants of Room 2. On the other hand, facts requiring disjunction or existential quantification cannot be expressed as part of a proper KB, *e.g.* every room having at most three occupants.

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.² Levesque [1998] proposed the reasoning procedure 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. $V[\Sigma, \rho\theta] = \begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in \Sigma \\ & \text{such that } V[\Sigma, e\theta] = 1 \\ 0 & \text{if there is a } \forall(e \supset \neg\rho) \in \Sigma \\ & \text{such that } V[\Sigma, e\theta] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$
2. $V[\Sigma, t = t'] = 1$ if t is identical to t' , and 0 otherwise;
3. $V[\Sigma, \neg\alpha] = 1 - V[\Sigma, \alpha]$;
4. $V[\Sigma, \alpha \vee \beta] = \max\{V[\Sigma, \alpha], V[\Sigma, \beta]\}$;
5. $V[\Sigma, \exists x\alpha] = \max\{V[\Sigma, \alpha_d^x] \mid d \in H^+(\Sigma \cup \{\alpha\})\}$, where $H^+(S)$ is the union of the constants in S , and a constant not in S , say the lexicographically smallest one.

This V procedure always terminates, and is logically sound (*i.e.* if it returns 1 (resp. 0), then the query (resp. its negation) is logically entailed by $\mathcal{E} \cup \Sigma$). Moreover, Levesque defined a certain normal form called $\mathcal{NF}$, and proved that for queries in $\mathcal{NF}$, the V procedure is also logically complete (so that, for instance, V returns 1 iff the query is logically entailed). Finally, Levesque showed that in the propositional case, every query could be equivalently expressed in $\mathcal{NF}$, and conjectured that this was also true in the first-order case.

2.2 Invariance under renaming

The first difficulty that arises when attempting to reason with proper KBs is the fact that unlike with databases, we cannot fix the domain in advance, or even put an upper bound on its size. For example, if Σ consists of just $\forall x(x \neq a \supset P(x))$, and α_n is the sentence of $\mathcal{L}$ that says that there are at least n distinct instances of P , then $V[\Sigma, \alpha_n] = 1$ for all n . However, we can take advantage of the fact that for this Σ , all constants other than a behave the same. This will allow us to define finite versions of answers to open queries, as shown below.

Notation Let α be a formula and S a set of formulas. Let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. We use α^* to denote α with each c replaced by c^* , and use S^* to denote $\{\alpha^* \mid \alpha \in S\}$.

²The classical validity problem is undecidable even when there are no function symbols.

Lemma 2.2 Let e be an ewff, and let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. Then $\mathcal{E} \models e$ iff $\mathcal{E} \models e^*$.

Theorem 2.3 Let Σ be a proper KB and α a sentence. Let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. Then $V[\Sigma, \alpha] = V[\Sigma^*, \alpha^*]$.

Proof: The proof is by induction on the structure of α . The case for atoms uses Lemma 2.2. The cases for equalities, negations and disjunctions are trivial. Now consider $\exists x\alpha$. Let $d \in H^+(\Sigma \cup \{\alpha\})$. If d appears in Σ or α , by induction, $V[\Sigma, \alpha_d^x] = V[\Sigma^*, (\alpha_d^x)^*] = V[\Sigma^*, (\alpha^*)_{d^*}^x]$. Otherwise, d^* does not appear in Σ^* or α^* . Now let $c \in H^+(\Sigma^* \cup \{\alpha^*\})$ such that c does not appear in Σ^* or α^* . Let $*$ be the composition of $*$ and the bijection that swaps c and d^* and leaves the rest constants unchanged. Then $d^* = c$; $\Sigma^* = \Sigma^*$, $\alpha^* = \alpha^*$, since neither d^* nor c appears in Σ^* or α^* . By induction, $V[\Sigma, \alpha_d^x] = V[\Sigma^*, (\alpha^*)_{d^*}^x] = V[\Sigma^*, (\alpha^*)_c^x]$. Thus $V[\Sigma, \exists x\alpha] = \max\{V[\Sigma, \alpha_d^x] \mid d \in H^+(\Sigma \cup \{\alpha\})\} = \max\{V[\Sigma^*, (\alpha^*)_d^x] \mid d \in H^+(\Sigma^* \cup \{\alpha^*\})\} = V[\Sigma^*, (\exists x\alpha)^*]$. ■

Corollary 2.4 Let Σ be a proper KB and α a formula with a single free variable x . Let c and d be two constants that do not appear in Σ or α . Then $V[\Sigma, \alpha_c^x] = V[\Sigma, \alpha_d^x]$.

Proof: Follows from the theorem by taking $*$ as the bijection that swaps c and d and leaves the rest constants unchanged. ■

2.3 Locality

A nice property of V is its locality, *i.e.*, for any proper KB Σ and query α , the value of $V[\Sigma, \alpha]$ only depends on those sentences of Σ that mention some predicate in α .

Definition 2.5 Let Σ be a proper KB and α a query. The restriction of Σ wrt α , denoted $\Sigma|_\alpha$, is the set $\{\forall(e \supset l) \in \Sigma \mid \text{the predicate of } l \text{ appears in } \alpha\}$.

Theorem 2.6 $V[\Sigma, \alpha] = V[\Sigma|_\alpha, \alpha]$.

Proof: The proof is by induction on the structure of α . Here we only prove the case for quantifications. $V[\Sigma, \exists x\alpha] = \max\{V[\Sigma, \alpha_d^x] \mid d \in H^+(\Sigma \cup \{\alpha\})\} = (\text{by induction}) \max\{V[\Sigma|_\alpha, \alpha_d^x] \mid d \in H^+(\Sigma \cup \{\alpha\})\} = (\text{by Corollary 2.4}) \max\{V[\Sigma|_\alpha, \alpha_d^x] \mid d \in H^+(\Sigma|_\alpha \cup \{\alpha\})\} = V[\Sigma|_\alpha, \exists x\alpha]$. ■

In fact, we can go further than this: roughly speaking, we need only consider those sentences $\forall(e \supset l) \in \Sigma$ such that after substitutions dictated by e , the l would unify with some literal in the query α . For example, $\forall xy(y = b \supset P(x, y))$ is not needed to answer the query $\exists xP(x, a)$. For KBs that are large because they have facts about a large number of individuals, this optimization can reduce significantly the size of the KB fragment needed to answer the query correctly.

2.4 Answers to open queries

Definition 2.7 Let Σ be a proper KB and $\alpha(\bar{x})$ a formula with free variables. The answer to α wrt Σ is the set $\alpha(\Sigma) = \{\bar{c} \mid V[\Sigma, \alpha(\bar{c})] = 1\}$.

However, $\alpha(\Sigma)$ may well be infinite. Fortunately, we can find a finite representation for it.

Definition 2.8 Let Σ be a proper KB and α a formula. Then $H(\Sigma, \alpha)$ is the finite set of constants consisting of those mentioned in Σ or α and the r next ones, lexicographically, where r is the quantifier rank of $\forall\alpha$.

Definition 2.9 Let Σ be a proper KB and $\alpha(\bar{x})$ a formula with free variables. The finite answer to α wrt Σ is the set $\alpha_f(\Sigma) = \{\bar{c} \in H(\Sigma|\alpha, \alpha) \mid V[\Sigma, \alpha(\bar{c})] = 1\}$.

The following theorem justifies $\alpha_f(\Sigma)$ as a finite representation for $\alpha(\Sigma)$.

Theorem 2.10 Let Σ be a proper KB and $\alpha(\bar{x})$ a formula. Then for any constants $\bar{c}, \bar{c} \in \alpha(\Sigma)$ iff $\bar{c}^* \in \alpha_f(\Sigma)$, where $\bar{c}^*$ is like $\bar{c}$ except that constants not in $\Sigma|\alpha \cup \{\alpha\}$ are replaced by unique representatives not in $\Sigma|\alpha \cup \{\alpha\}$ from $H(\Sigma|\alpha, \alpha)$.

Proof: We can expand $\star$ to a bijection $*$ from $\mathcal{C}$ to $\mathcal{C}$ such that $c^* = c$ for any c that appears in $\Sigma|\alpha$ or α . Then $V[\Sigma, \alpha(\bar{c})] =$ (by locality) $V[\Sigma|\alpha, \alpha(\bar{c})] =$ (by Theorem 2.3) $V[(\Sigma|\alpha)^*, (\alpha(\bar{c}))^*]$, i.e. $V[\Sigma|\alpha, \alpha(\bar{c}^*)] = V[\Sigma, \alpha(\bar{c}^*)]$. ■

3 The complexity of database queries

3.1 An overview

The complexity of query evaluation has been one of the main pursuits of database theory. Traditionally, there are two complexity measures: combined complexity and data complexity [Vardi, 1982]. Combined complexity is measured in terms of the combined size of the database and the query. Chandra and Merlin [1977] proved that the combined complexity of conjunctive queries (queries expressed by first-order formulas that are of the form $\exists \bar{x}(\alpha_1 \wedge \dots \wedge \alpha_n)$ where α_i are atoms) is NP-complete; Vardi [1982] proved that the combined complexity of first-order queries is PSPACE-complete. However, the main factor responsible for this high complexity is the size of the query and not the size of the database. This is contrary to the situation in practice where we normally evaluate small queries against large databases. Data complexity measures the complexity of query evaluation solely in terms of the size of the database and treats the size of the query as a constant. A folk result is that first-order queries can be evaluated in time $n^{O(l)}$, where n is the size of the database and l is the size of the query. Thus the data complexity of first-order queries is in PTIME. However, such a complexity can hardly qualify as tractable even if the exponent is small, say 5.

Yannakakis [1995] was the first to suggest that parameterized complexity [Downey and Fellows, 1995] might be an appropriate complexity measure for database query evaluation. Query evaluation is fixed-parameter tractable if there is a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ and a constant c such that the problem can be solved in time $f(l) \cdot n^c$, where l and n are the size of the query and the database, respectively. However, Papadimitriou and Yannakakis [1999] proved that the parameterized complexity of conjunctive queries is W[1]-complete and thus most likely not fixed-parameter tractable. We refer the reader to [Grohe, 2002] for a survey on parameterized complexity in database theory.

Therefore database query evaluation in general is hard with respect to both combined and parameterized complexity. And yet, database queries do work quite well in practice, even for very large databases. A careful examination of the hardness results show that they often depend on queries that are somewhat atypical in applications, e.g. databases representing graphs and queries asking for the existence of cliques.

Naturally, many research efforts have gone into finding classes of queries that can be proven tractable, even in the worst cases. The earliest result of this form, due to Yannakakis [1981], showed that acyclic conjunctive queries can be evaluated in polynomial time. This result has been extended in several ways. The first extension, due to Chekuri and Rajaraman [1997], showed that conjunctive queries with bounded treewidth are tractable. Later, Gottlob *et al.* [1999] introduced the notion of hypertree width and showed that conjunctive queries with bounded hypertree width are tractable; bounded hypertree width generalizes the notions of acyclicity and bounded treewidth. Recently, Flum *et al.* [2001] generalized the notions of acyclicity and bounded treewidth from conjunctive queries to nonrecursive stratified Datalog (NRSD), which is known to have the same expressive power as all of first-order logic, and showed that acyclic and bounded treewidth NRSD are tractable. Inspired by their work, Gottlob *et al.* [2001] extended the notion of hypertree width to NRSD, and obtained a nice logical characterization of hypertree width: they showed that the k -guarded fragment of first-order logic has the same expressive power as NRSD of hypertree width at most k . Thus k -guarded first-order logic turns out to be the largest tractable class of queries so far.

3.2 An evaluation algorithm

The guarded fragment GF was introduced by Andr  ka *et al.* [1996], and has recently received a lot of attention. The idea is that any existentially quantified subformula ϕ must be conjoined with a guard, i.e. an atom containing all free variables of ϕ . Gottlob *et al.* [2001] extended GF to the k -guarded fragment GF_k where up to k atoms may jointly act as a guard, and proved that GF_k -formulas can be evaluated in time $O(l^2 n^k)$ where l and n are as before. The proof is by transforming GF_k -formulas into k -guarded NRSD programs.

In this section, we introduce the k -guarded fragment of $\mathcal{L}$, and explicitly present a polynomial algorithm for evaluating k -guarded formulas against databases and analyze its complexity. We will use this algorithm to evaluate k -guarded formulas with respect to proper KBs.

Definition 3.1 The k -guarded fragment GF_k of $\mathcal{L}$ is the smallest set of formulas such that

1. GF_k contains atoms and equalities;
2. GF_k is closed under Boolean operations;
3. GF_k contains $\exists \bar{x}(G_1 \wedge \dots \wedge G_m \wedge \phi)$, provided that the G_i are atoms or equalities, $m \leq k$, $\phi \in GF_k$, and the free variables of ϕ appear among the G_i .

Definition 3.2 The strictly k -guarded fragment SGF_k are the formulas in GF_k of the form $\exists \bar{x}(G_1 \wedge \dots \wedge G_m \wedge \phi)$. We allow the degenerate cases where $\bar{x}$ is empty or $m = 0$ or ϕ is TRUE.

Note that any k -guarded sentence is strictly k -guarded.

The evaluation algorithm below will take a strictly k -guarded formula as the query. It turns out that any formula ϕ can be transformed in linear time into an equivalent SGF_k -formula for some k , using guards of the form $x = x$, if necessary. For example, note that $R(x, y, z) \wedge S(z, u, v)$ can be formulated in SGF_2 , but $\neg R(x, y, z) \wedge \neg S(z, u, v)$ only in SGF_5 .

The evaluation algorithm will have worst-case time complexity that is exponential only in the k . For fixed k , the algorithm is polynomial, but perhaps impractical when k is large. An upper bound on k is the width of ϕ . Obviously, if ϕ has at most k distinct variables, the width of ϕ is no more than k .

Before presenting the evaluation algorithm, we first recall some basic notions of relational database theory. A database instance is simply a logical structure (or interpretation) with a finite domain. Let $\mathcal{A}$ be such a structure with domain A . Let X and Y be sets of variables. An X -relation R over A , is a finite set of mappings γ from X into A , which we write as $\{\gamma : [X \rightarrow A]\}$. If $Y \subseteq X$, the Y -projection of an X -relation R , which we write $\pi_Y(R)$, is the set $\{\gamma|Y \mid \gamma \in R\}$, where $\gamma|Y$ is the mapping restricted to Y . Let R be an X -relation and S a Y -relation. The join of R and S is the $X \cup Y$ -relation

$$R \bowtie S = \{\gamma : [X \cup Y \rightarrow A] \mid \gamma|X \in R, \gamma|Y \in S\}.$$

The join of R with the complement of S is the $X \cup Y$ -relation

$$R \bowtie^c S = \{\gamma : [X \cup Y \rightarrow A] \mid \gamma|X \in R, \gamma|Y \notin S\}.$$

Let ϕ be a formula with free variables $X = \{x_1, \dots, x_n\}$. The answer to ϕ wrt $\mathcal{A}$ is the X -relation

$$\phi(\mathcal{A}) = \{\gamma : [X \rightarrow A] \mid \mathcal{A} \models \phi(\gamma(x_1), \dots, \gamma(x_n))\}.$$

Here is the algorithm for computing answers:

Procedure Eval($\mathcal{A}, \phi$)

Input: A structure $\mathcal{A}$ such that the domain and all relations are sorted, and a formula $\phi \in \text{SGF}_k$

Output: $\phi(\mathcal{A})$

suppose $\phi(\bar{x}) = \exists \bar{y}(G_1 \wedge \dots \wedge G_m \wedge \psi)$
 return $\pi_{\bar{x}}\{\text{Eval}(\mathcal{A}, G_1(\mathcal{A}) \bowtie \dots \bowtie G_m(\mathcal{A}), \psi')\}$
 where ψ' is the result of pushing $\neg$ s in ψ so that they are in front of atoms, equalities, or existentials.

Procedure Eval($\mathcal{A}, R, \phi$) is defined recursively by:

1. If ϕ is an atom or an equality,
 then Eval($\mathcal{A}, R, \phi$) = $R \bowtie \phi(\mathcal{A})$;
2. If ϕ is negation of an atom or an equality,
 then Eval($\mathcal{A}, R, \phi$) = $R \bowtie^c \phi(\mathcal{A})$;
3. Eval($\mathcal{A}, R, (\phi \wedge \psi)$) = Eval($\mathcal{A}, R, \phi$) $\cap$ Eval($\mathcal{A}, R, \psi$);
4. Eval($\mathcal{A}, R, (\phi \vee \psi)$) = Eval($\mathcal{A}, R, \phi$) $\cup$ Eval($\mathcal{A}, R, \psi$);
5. Eval($\mathcal{A}, R, \exists \bar{y}\phi$) = $R \bowtie$ Eval($\mathcal{A}, \exists \bar{y}\phi$);
6. Eval($\mathcal{A}, R, \neg \exists \bar{y}\phi$) = $R \bowtie^c$ Eval($\mathcal{A}, \exists \bar{y}\phi$).

Theorem 3.3 Given a sorted structure $\mathcal{A}$ and a formula $\phi \in \text{SGF}_k$, $\phi(\mathcal{A})$ can be computed in time $O(\ln^k)$, where l is the size of ϕ , and n is the size of the database.

Proof: For each of the l literals and logical operators in the query, we perform either a join, a join with complement, an intersection, or a union. Note that when we perform a join or a join with complement within Eval($\mathcal{A}, R, \phi$), the attributes of the second argument is always a subset of those of the first, since for guarded formulas, the free variables in ϕ are a subset of those in R . By induction, Eval($\mathcal{A}, R, \phi$) always returns a relation that is a subset of R . Moreover, the R is obtained from joins of at most k guards, and so has size at most n^k . Assuming all relations start in sorted form, the results of all operations can be kept in sorted form, and computed in time $O(n^k)$. Thus, the time overall is $O(\ln^k)$. ■

4 The complexity of V

In this section, we consider how hard it is to compute V . Not surprisingly, V is no easier than database query evaluation. What is more significant is that under reasonable assumptions, it is not much harder either.

4.1 Intractability results

Theorem 4.1 The combined complexity of V is NP-hard for conjunctive queries, and PSPACE-hard for first-order queries. The parameterized complexity of V is W[1]-hard for conjunctive queries.

Proof: The proof is essentially the same as that for database query evaluation. For conjunctive queries, the reduction is from the clique problem. For first-order queries, the reduction is from QBF (Quantified Boolean Formula). ■

4.2 A tractability result

We now present a tractability result for V . The basic idea is to reduce V to database query evaluation and use the algorithm Eval from Section 3. To do this efficiently, we would like to use database operations as much as possible, and only reason with the formulas e in $\forall(e \supset l)$ when necessary. For each predicate P appearing in a query α , we will consider storing two relations P^+ and P^- depending on the form of Σ .

Definition 4.2 The ewff defining a predicate P denoted e_P , is the disjunction of all e such that $\forall \bar{x}(e(\bar{x}) \supset P(\bar{x})) \in \Sigma$. The ewff defining $\neg P$ denoted $e_{\neg P}$ is analogous.

Definition 4.3 We distinguish among three cases for an ewff:

1. e is relational if it is of the form
 $\bar{x} = \bar{c}_1 \vee \dots \vee \bar{x} = \bar{c}_n$ where $n \geq 0$.
2. e is co-relational if it is of the form
 $\bar{x} \neq \bar{c}_1 \wedge \dots \wedge \bar{x} \neq \bar{c}_n$ where $n > 0$.
3. e is unrestricted otherwise.

Definition 4.4 The e -size of Σ is the maximum size of an unrestricted e_P or $e_{\neg P}$ in Σ .

For each predicate P , we store P^+ as a sorted relation when e_P is relational or co-relational, and analogously for P^- using $e_{\neg P}$. We let $DB(\Sigma, C)$ (where C is a set of constants that includes all those in Σ) denote the database whose domain is C and whose relations are P^+ and P^- as above.

Note that in the case of a predicate P in Σ corresponding to a normal database-style relation, e_P will be relational and $e_{\neg P}$ will be co-relational and the two relations P^+ and P^- will be identical. A simple optimization would merge them.

Next, given a formula α we construct a database query by replacing every occurrence of a predicate P as follows: a positive³ occurrence of P is replaced by P^+ when e_P is relational, $\neg P^+$ when co-relational, and by e_P itself otherwise; a negative occurrence of P is replaced by $\neg P^-$ when $e_{\neg P}$ is relational, P^- when co-relational, and by $\neg e_{\neg P}$ itself otherwise. We let $Q(\Sigma, \alpha)$ denote the resulting database query with double negation removed.

We present examples of this transformation in the next section. Here we note the following correctness result:

³A positive (resp. negative) occurrence of P in α is one within the scope of an even (resp. odd) number of negations.

Theorem 4.5 Let Σ be a proper KB and α a formula of $\mathcal{L}$. Let $C = H(\Sigma|\alpha, \alpha)$, $\mathcal{A} = DB(\Sigma|\alpha, C)$, and $\phi = Q(\Sigma, \alpha)$. Then $\alpha_f(\Sigma) = \phi(\mathcal{A})$.

This theorem says that the (finite version of) answers to open queries in $\mathcal{L}$ correspond exactly to the answers we get for the database constructed as above. This is a consequence of locality and the following lemma:

Lemma 4.6 Let Σ be a proper KB and α a sentence of $\mathcal{L}$. Let C be any finite set of constants consisting of those in $\Sigma \cup \{\alpha\}$ and k extra ones for some $k \geq qr(\alpha)$. Let $\mathcal{A} = DB(\Sigma|\alpha, C)$. Then $V[\Sigma, \alpha] = 1$ iff $\mathcal{A} \models Q(\Sigma, \alpha)$, and $V[\Sigma, \alpha] = 0$ iff $\mathcal{A} \models Q(\Sigma, \neg\alpha)$.

Proof: The proof is by induction on the structure of α . Here we choose to prove the case for $V[\Sigma, \alpha] = 0$ which is trickier than the case for $V[\Sigma, \alpha] = 1$.

$V[\Sigma, P(\bar{c})] = 0$ iff $\mathcal{E} \models e_{\neg P}(\bar{c})$ iff $\mathcal{A} \models Q(\Sigma, \neg P(\bar{c}))$ since

$$\mathcal{E} \models e_{\neg P}(\bar{c}) \text{ iff } \begin{cases} \mathcal{A} \models P^-(\bar{c}) & \text{if } e_{\neg P} \text{ is relational} \\ \mathcal{A} \models \neg P^-(\bar{c}) & \text{if } e_{\neg P} \text{ is co-relational} \\ \mathcal{A} \models e_{\neg P}(\bar{c}) & \text{otherwise} \end{cases}$$

The case for $t = t'$ is trivial. $V[\Sigma, \neg\alpha] = 0$ iff $V[\Sigma, \alpha] = 1$ iff (by induction) $\mathcal{A} \models Q(\Sigma, \alpha)$, i.e. $Q(\Sigma, \neg\neg\alpha)$.

Consider $\alpha \vee \beta$. Let $Q(\Sigma, \neg\alpha) = \neg\phi$ and $Q(\Sigma, \neg\beta) = \neg\psi$. Then $Q(\Sigma, \neg(\alpha \vee \beta)) = \neg(\phi \vee \psi)$. $V[\Sigma, \alpha \vee \beta] = 0$ iff $V[\Sigma, \alpha] = 0$ and $V[\Sigma, \beta] = 0$ iff (by induction) $\mathcal{A} \models Q(\Sigma, \neg\alpha)$ and $\mathcal{A} \models Q(\Sigma, \neg\beta)$ iff $\mathcal{A} \models \neg\phi \wedge \neg\psi$ iff $\mathcal{A} \models \neg(\phi \vee \psi)$, i.e. $Q(\Sigma, \neg(\alpha \vee \beta))$.

Consider $\exists x\alpha$. Let $Q(\Sigma, \neg\alpha) = \neg\phi$. Then $Q(\Sigma, \neg\exists x\alpha) = \neg\exists x\phi$. $V[\Sigma, \exists x\alpha] = 0$ iff $V[\Sigma, \alpha_d^x] = 0$ for any $d \in H^+(\Sigma \cup \{\alpha\})$ iff (by Corollary 2.4 since C contains at least one constant not in $\Sigma \cup \{\alpha\}$) $V[\Sigma, \alpha_d^x] = 0$ for any $d \in C$ iff (by induction since C consists of those constants in $\Sigma \cup \{\alpha_d^x\}$ and j extra ones for some $j \geq qr(\alpha)$) $\mathcal{A} \models Q(\Sigma, \neg\alpha_d^x)$, i.e. $\neg\phi_d^x$ for any $d \in C$ iff $\mathcal{A} \models \forall x\neg\phi$ iff $\mathcal{A} \models \neg\exists x\phi$, i.e. $Q(\Sigma, \neg\exists x\alpha)$. ■

To obtain a tractability result for open queries over proper KBs, we need only ensure that the database queries we construct are k -guarded then use the Eval procedure.

Definition 4.7 Let Σ be a proper KB. The strictly k -guarded fragment $SG_k(\Sigma)$ of $\mathcal{L}$ for Σ is the same as SGF_k except that when $\exists \bar{x}(G_1 \wedge \dots \wedge G_m \wedge \phi)$ is within the scope of an even (resp. odd) number of $\neg$ s, the G_i must be equalities or atoms $P(\bar{t})$ s.t. e_P is relational (resp. $e_{\neg P}$ is co-relational) or literals $\neg P(\bar{t})$ s.t. $e_{\neg P}$ is relational (resp. e_P is co-relational).

Clearly, for any α , if $\alpha \in SG_k(\Sigma)$, then $Q(\Sigma, \alpha) \in SGF_k$. Note that every formula α with width m can be transformed into an equivalent $SG_k(\Sigma)$ -formula for some $k \leq m$, using equalities as guards if necessary.

Now we get our main complexity result:

Theorem 4.8 Let Σ be a proper KB and $\alpha \in SG_k(\Sigma)$. Then $\alpha_f(\Sigma)$ can be computed in time $O(lwn^k)$, where l is the size of α , n is the size of $\Sigma|\alpha$, and w is the e -size of $\Sigma|\alpha$.

Proof: Let $C = H(\Sigma|\alpha, \alpha)$, $\mathcal{A} = DB(\Sigma|\alpha, C)$, and $\phi = Q(\Sigma, \alpha)$. We sort C . By Theorem 4.5, $\alpha_f(\Sigma) = \text{Eval}(\mathcal{A}, \phi)$.

The size of $\mathcal{A}$ is $O(n)$, and the length of ϕ is $O(lw)$. By Theorem 3.3, $\alpha_f(\Sigma)$ can be computed in time $O(lwn^k)$. ■

As a special case of the theorem, we have:

Corollary 4.9 Let Σ be a proper KB and α a sentence in $SG_k(\Sigma)$. Then whether $V[\Sigma, \alpha] = 1$ can be decided in time $O(lwn^k)$, where l , w and n are as above.

Corollary 4.10 Let Σ be a proper KB and $\alpha \in \mathcal{NF} \cap SG_k(\Sigma)$. Then whether $\mathcal{E} \cup \Sigma \models \alpha$ can be decided in time $O(lwn^k)$, where l , w and n are as above.

Proof: Follows from previous corollary and the fact that V is logically sound and complete for queries in $\mathcal{NF}$. ■

Note that these bounds are identical to their database counterparts modulo the w factor. In most cases of interest, the w will be small since even in a large KB, we only expect to see a large e_P or $e_{\neg P}$ in database-like cases, where the e is relational or co-relational. In the case where none of the e are relational or co-relational, we get the following:

Corollary 4.11 Let Σ be a proper KB and α a formula of width k with all literals defined by an unrestricted ewff. Then $\alpha_f(\Sigma)$ can be evaluated in time $O((lw)^{k+1})$, where l is the size of α , and w is the e -size of $\Sigma|\alpha$.

Proof: We transform α into an equivalent $SG_k(\Sigma)$ -formula, using equalities as guards. The size of $\Sigma|\alpha$ is $O(lw)$. ■

5 An example

To illustrate how query evaluation would work with proper KBs, we return to the example robotic scenario mentioned in the first section, involving the predicate $In(\text{person}, \text{room})$. We can see from the example that e_{In} is given in relational form, while $e_{\neg In}$ is given in unrestricted form. To make things interesting, we assume two more predicates: One is $Mgr(\text{person}_1, \text{person}_2)$ saying that the first person is a manager and that the second is one of his or her employees; we assume that Mgr is given as a closed database predicate. The other is $Cmp(\text{person}_1, \text{person}_2)$ saying that the two persons are compatible. We assume that the robot knows that any two people are compatible except $m(> 2)$ pairs; among these m pairs, the robot only knows that two pairs are not compatible. Hence e_{Cmp} is given in co-relational form, and $e_{\neg Cmp}$ is given in relational form. Thus the corresponding database has the following relations: In^+ , Mgr^+ , Mgr^- , Cmp^+ , and Cmp^- , where $Mgr^+ = Mgr^-$, and $Cmp^- \subseteq Cmp^+$.

Here are some example queries, with the guards underlined.

- What rooms are unoccupied?
 $r = r \wedge \neg \exists x(x = x \wedge r = r \wedge \text{In}(x, r))$
- Are there rooms with more than one manager in them?
 $\exists r \exists m \exists m'(\text{In}(\underline{m, r}) \wedge \text{In}(\underline{m', r}) \wedge m \neq m' \wedge \exists x \text{Mgr}(\underline{m, x}) \wedge \exists x' \text{Mgr}(\underline{m', x'}))$
- Are there any managers whose employees are all in the same room?
 $\exists m \exists r(m = m \wedge r = r \wedge \neg \exists x(\text{Mgr}(\underline{m, x}) \wedge r = r \wedge \neg \text{In}(x, r)))$
- Is it true that any two people who are not compatible are in different rooms?
 $\neg \exists x \exists x'(\neg \text{Cmp}(\underline{x, x'}) \wedge \neg \exists r \exists r'(\text{In}(\underline{x, r}) \wedge \text{In}(\underline{x', r'}) \wedge r \neq r'))$

Observe that in all cases, the queries are strictly 2-guarded wrt the proper KB. Also, because no query contains a literal and a unifiable literal of opposite polarity, from results in [Levesque, 1998], all the queries are in $\mathcal{NF}$.

The corresponding database queries are as follows:

1. $r = r \wedge \neg \exists x (x = x \wedge r = r \wedge \neg e_{-In}(x, r))$
2. $\exists r \exists m \exists m' (In^+(m, r) \wedge In^+(m', r) \wedge m \neq m' \wedge \exists x \underline{Mgr^+(m, x)} \wedge \exists x' \underline{Mgr^+(m', x')})$
3. $\exists m \exists r (m = m \wedge r = r \wedge \neg \exists x (\underline{Mgr^-(m, x)} \wedge r = r \wedge \neg In^+(x, r)))$
4. $\neg \exists x \exists x' (\underline{Cmp^+(x, x')} \wedge \neg \exists r \exists r' (\underline{In^+(x, r)} \wedge In^+(x, r') \wedge r \neq r'))$

6 Conclusions

In this paper, we have shown how a bottom-up query evaluation procedure for databases can be used to answer queries for KBs with a certain form of incomplete knowledge. Although this procedure can be impractical for k -guarded queries where k is large, they would be impractical for databases too.

A number of questions remain to be addressed. First of all, Lakemeyer and Levesque [2002] have proposed an extension to proper KBs that allow disjunctions in the KB. It would be interesting to see how much of the database retrieval mechanism could be preserved in this case. We can also imagine other extensions to proper KBs, such as relaxing the unique name assumption over constants, or allowing a limited use of function symbols. Also, as suggested in the first section, we can imagine a dynamic scenario where at any given point what a robot or agent knows about the current situation is expressible as a proper KB. It would then be useful to amalgamate regression-based techniques for reasoning about change from [Reiter, 2001] with the database techniques considered here. Among other things, this would require determining those cases where the successor state axioms guarantee that a proper KB remains proper after an action has been performed, perhaps along the lines of [Petrick and Levesque, 2002]. It would also be interesting to investigate the relationship between proper KBs and other subsets of logic to see if the complexity results presented here can be further generalized. Two immediate candidates are datalog programs and stratified logic programs that include some form of classical negation. Finally, we note that additional optimizations can be made to our query evaluation procedure that do not change the worst-case performance, but would improve its behaviour in practice.

Acknowledgments

We would like to thank Leonid Libkin for pointing us to the relevant database literature. Financial support was gratefully received from the Natural Sciences and Engineering Research Council of Canada.

References

- [Andréka *et al.*, 1996] H. Andréka, I. Hodkinson, and I. Németi. Modal languages and bounded fragments of predicate logic. ILLC Research Report ML-96-03, University of Amsterdam, 1996.
- [Chandra and Merlin, 1977] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Conference record of the 9th annual ACM Symp. on Theory of Computing*, pages 77–90, 1977.
- [Chekuri and Rajaraman, 1997] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *Proc. 6th Int. Conf. on Database Theory*, pages 56–70, 1997.
- [Downey and Fellows, 1995] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.
- [Flum *et al.*, 2001] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. In *Proc. of the 8th Int. Conf. on Database Theory*, pages 22–38, 2001.
- [Gottlob *et al.*, 1999] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In *Proc. 18th ACM Symp. on Principles of Database Systems*, 1999.
- [Gottlob *et al.*, 2001] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width. In *Proc. of the 20th ACM Symp. on Principles of Database Systems*, 2001.
- [Grohe, 2002] M. Grohe. Parameterized complexity for the database theorist. *SIGMOD Record*, 31(4), 2002.
- [Lakemeyer and Levesque, 2002] G. Lakemeyer and H.J. Levesque. Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In *Proc. of KR-02*, pages 73–81, 2002.
- [Levesque, 1998] H.J. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. of KR-98*, pages 14–23, 1998.
- [Papadimitriou and Yannakakis, 1999] C.H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.
- [Petrick and Levesque, 2002] R. Petrick and H.J. Levesque. Knowledge equivalence in combined action theories. In *Proc. of KR-02*, pages 303–314, 2002.
- [Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
- [Vardi, 1982] M.Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th Annual ACM Symp. on Theory of Computing*, pages 137–146, 1982.
- [Yannakakis, 1981] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases*, pages 82–94, 1981.
- [Yannakakis, 1995] M. Yannakakis. Perspectives on database theory. In *Proc. of the 36th Annual Symp. on Foundations of Computer Science*, pages 224–246, 1995.

Practical Partition-Based Theorem Proving for Large Knowledge Bases

Bill MacCartney

Knowledge Systems Lab
Computer Science Dept.
Stanford University
bmac@ksl.stanford.edu

Sheila McIlraith

Knowledge Systems Lab
Computer Science Dept.
Stanford University
sam@ksl.stanford.edu

Eyal Amir

Computer Science Div.
University of California
Berkeley, CA
eyal@cs.berkeley.edu

Tomás E. Uribe

AI Center
SRI International
Menlo Park, CA
uribe@ai.sri.com

Abstract

Query answering over commonsense knowledge bases typically employs a first-order logic theorem prover. While first-order inference is intractable in general, provers can often be hand-tuned to answer queries with reasonable performance in practice. Appealing to previous theoretical work on *partition-based reasoning*, we propose resolution-based theorem proving strategies that exploit the structure of a KB to improve the efficiency of reasoning. We analyze and experimentally evaluate these strategies with a testbed based on the SNARK theorem prover. Exploiting graph-based partitioning algorithms, we show how to compute a *partition-derived ordering* for ordered resolution, with good experimental results, offering an automatic alternative to hand-crafted orderings. We further propose a new resolution strategy, *MFS resolution*, that combines partition-based message passing with focused sublanguage resolution. Our experiments show a significant reduction in the number of resolution steps when this strategy is used. Finally, we augment partition-based message passing, partition-derived ordering, and MFS by combining them with the set-of-support restriction. While these combinations are incomplete, they often produce dramatic improvements in practice. This work presents promising practical techniques for query answering with large and potentially distributed commonsense KBs.

1 Introduction

Theorem proving in first-order logic (FOL) is intractable in general. Nevertheless, first-order provers are commonly used for query answering over large knowledge bases (KBs) containing thousands of axioms, such as Cycorp's Cyc and the High Performance Knowledge Base (HPKB) [4]. To make headway in large KBs, theorem provers usually require KB-specific tuning and customization.

Partition-based reasoning (PBR) [2; 12] promises to speed up reasoning, without manual tuning, by exploiting the structure implicit in such large commonsense KBs, which typically contain loosely coupled clusters of domain knowledge.

PBR uses graph-based techniques to automatically partition a logical theory into a network of subtheories minimally connected by links of shared vocabulary. Theorem proving is performed locally in each subtheory, and relevant results are propagated between partitions to achieve globally sound and complete collaborative reasoning.

Previous work on PBR has presented a theoretical framework and made claims about the potential for improving the efficiency of reasoning in practice. In this paper we validate these claims empirically, and introduce novel FOL resolution strategies that exploit PBR techniques to improve the efficiency of reasoning.

Outline: In Section 2, we review the theoretical framework of PBR. In Section 3 we explain how a generic theorem prover may be easily augmented with PBR, and describe the experimental testbed we developed using the SNARK theorem prover [18]. Using this testbed, in Section 4 we compare the performance of the PBR message-passing algorithm (MP) to that of popular resolution strategies [3]. MP far outperforms unrestricted resolution, fares comparably to ordered resolution with a default ordering, and sometimes beats set-of-support (SOS) resolution.

In Section 5 we show how automatic partitioning [1] can induce a *partition-derived ordering* (PDO) for use with ordered resolution. Ordering can be a highly efficient resolution strategy, but its success has previously depended on hand-crafted orderings tailored to a specific KB, often through trial and error. Our PDO is competitive with hand-crafted orderings and far outperforms SNARK's default ordering. This important result will let future theorem provers incorporate efficient automatic ordering.

In Section 6, we present a novel resolution strategy, MFS resolution, and show it to be sound and complete. MFS combines MP with a *focused support* restriction employed within partitions. Focused support is a novel resolution restriction that is complete for consequence-finding in a specified sublanguage. In experiments, MFS significantly reduces the number of resolution steps required to answer queries.

Finally, in Section 7 we examine combinations of each of these strategies (MP, PDO, and MFS) with SOS. While these combinations are, in general, incomplete, they perform well in experimental testing. PDO+SOS was found to outperform every other strategy and combination examined, and its theoretical incompleteness was never encountered in practice.

This paper introduces novel, easily implemented techniques for improving the efficiency of FOL theorem proving with large structured KBs such as commonsense KBs. It is the first to report experimental results for PBR, and the first to examine combinations of MP with other strategies in a theoretical or experimental setting.

2 Background: partition-based reasoning

The PBR framework has two components: graph-based algorithms for automatic partitioning of a theory, and message-passing algorithms for reasoning with the partitioned theory. For further details see [2; 12; 1].

2.1 Automatically partitioning a theory

We say that $\{\mathcal{A}_i\}_{i \leq n}$ is a *partitioning* of a logical theory $\mathcal{A}$ if $\mathcal{A} = \bigcup_i \mathcal{A}_i$. Each $\mathcal{A}_i$ is a set of axioms called a *partition*, $L(\mathcal{A}_i)$ is its signature (the set of non-logical symbols), and $\mathcal{L}(\mathcal{A}_i)$ is its language (the set of formulae built with $L(\mathcal{A}_i)$). Partitions may share symbols and axioms.

To partition a theory, we first construct a *symbols graph* from the axioms, where each vertex represents a symbol in the theory, and two vertices are joined by an edge iff the two symbols appear together in an axiom. We then use one of several *tree decomposition* algorithms [16; 1] to generate a tree where each node corresponds to a tightly connected cluster of symbols, defining a partition $\mathcal{A}_i$ consisting of the axioms in the original theory that contain only those symbols.

The result is a *partition graph* $G : (V, E, l)$, where vertices correspond to partitions, and edges correspond to communication links between partitions, labeled with their *communication vocabulary*: the label $l(i, j)$ on the edge between $\mathcal{A}_i$ and $\mathcal{A}_j$ is the set of symbols shared by the two partitions, that is, $l(i, j) = L(\mathcal{A}_i) \cap L(\mathcal{A}_j)$. Efficient partition-based reasoning depends on finding a tree decomposition that keeps the communication languages small.

Partitioning Algorithms

A *tree decomposition* is a tree of nodes, where each node is a set of symbols and the tree satisfies the following property: if a symbol appears in two different nodes, then it appears in all the nodes and edges on the path between the two nodes. The *width* of a tree decomposition is the size (number of symbols) of its largest node (the largest set of symbols in a single node). An *optimal* tree decomposition is one which has the lowest width among all tree decompositions for the graph. This tree's width is the *treewidth* of the original graph. Tree decomposition algorithms are commonly used in a variety of AI applications, including constraint satisfaction problems and Bayes Nets.

We have performed experiments with two graph partitioning algorithms. The first algorithm is due to [16] and is based on an ordering heuristic named *min fill*. In this heuristic we iterate over the symbols in the symbols graph. At each step, we select a symbol in the graph, add all missing edges between the symbol's neighbors in the graph, and remove that symbol from the graph. The symbol is selected such that the number of added edges is minimal. The tree decomposition is extracted from this order. The experimental results reported here are based on this partitioning algorithm.

The second algorithm is due to [1] and uses a divide-and-conquer approach that is guaranteed to approximate the optimum decomposition by a factor of at most $O(\log t)$, where t is the treewidth of the symbols graph. Iteratively, the algorithm finds a vertex cut in the graph (a set of vertices whose removal separates the graph into two or more parts). This vertex cut, W , is sent recursively to each of the separated parts, and subsequent iterations are required to find a vertex cut that separates W in a *balanced* fashion (here, we may use algorithms such as [11; 9]). The tree decomposition is built from the final parts and the vertex-cuts' vertex sets recursively. In experiments, this algorithm generated partitions somewhat fewer in number and larger in size than the first algorithm, but yielded similar performance in answering queries.

2.2 Reasoning with message passing (MP)

Figure 1 displays Forward-Message-Passing (MP), a PBR algorithm proposed in [2]. It takes as input a partitioned theory $\mathcal{A}$, an associated partition graph $G = (V, E, l)$, and a query formula Q in $\mathcal{L}(\mathcal{A}_k)$, and returns YES if the query was entailed by $\mathcal{A}$. MP first directs all edges in the partition graph G toward the goal partition $\mathcal{A}_k$. Since G is a tree, this means each partition (except the goal) will have exactly one “outbound link” (leading to the next partition on the path to the goal) and an “outbound link vocabulary” L_{out} . MP then performs consequence finding independently in each partition, and propagates each derived formula over the outbound link toward the goal iff the formula's signature matches L_{out} .

PROCEDURE FORWARD-M-P (MP)($\{\mathcal{A}_i\}_{i \leq n}, G, Q$)

$\{\mathcal{A}_i\}_{i \leq n}$ a partitioning of the theory $\mathcal{A}$,
 $G : (V, E, l)$ a graph with the connections between partitions,
 Q a query in $\mathcal{L}(\mathcal{A}_k)$ (for some $k \leq n$).

1. Define $i \prec j$ iff i is on the path between j and k in G .
2. Concurrently,
 - (a) Perform consequence finding for each of the partitions $\mathcal{A}_i, i \leq n$.
 - (b) For every $(i, j) \in E$ such that $i \prec j$, for every consequence φ of $\mathcal{A}_j$ found (or φ in $\mathcal{A}_j$), if $\varphi \in \mathcal{L}(l(i, j))$, then add φ to the set of clauses of $\mathcal{A}_i$.
 - (c) If Q is proven^a in $\mathcal{A}_k$, return YES.

^aWe initially add $\neg Q$ to $\mathcal{A}_k$ and derive inconsistency.

Figure 1: A forward message-passing algorithm.

This algorithm was shown sound and complete when the partition graph corresponds to a tree decomposition, and the reasoning procedure employed in each partition is sound and complete for $\mathcal{L}$ -consequence generation, where $\mathcal{L}$ is the language defined by the outbound link vocabulary L_{out} :

Definition 2.1. Given a target language $\mathcal{L}$, a reasoning procedure $\mathcal{R}$ is complete for $\mathcal{L}$ -consequence generation if every formula in $\mathcal{L}$ logically entailed by a set of axioms S is also entailed by the set of consequences in $\mathcal{L}$ that is generated by $\mathcal{R}$ from S .

Ordered resolution is an example of a reasoning procedure that satisfies this condition [7].

3 Experimental setup

To evaluate the performance of various PBR strategies, we built an experimental platform around SNARK, a resolution-based FOL theorem prover developed by Mark Stickel at the SRI AI Center ([18]). Adding PBR capabilities to SNARK was straightforward. Three extensions were required:

- Associate a set of partitions with each clause.
- Restrict resolution to occur only within partitions.
- Ensure appropriate propagation of new clauses to neighboring partitions.

These modifications should be easy to implement in any resolution-based prover.

3.1 Implementing PBR in SNARK

In SNARK, a *row* is a data structure that stores a clause along with metadata such as the inference method used to derive it. We extend this data structure to include a (non-empty) set of partition IDs indicating the partitions in which the clause resides. A separate data structure holds a description of the partition graph: the partitions, the links that connect them, and the vocabulary for each partition and link.

Resolution is restricted to occur only within partitions: two rows may be resolved only if the intersection of their partition sets is non-empty. The resolvent inherits this intersection as its own partition set, so that it appears in the same partitions as its parents. (This restriction may be superimposed on top of other resolution restrictions, such as set-of-support.)

Finally, to implement message passing between partitions, we examine the vocabulary of every new resolvent. If it matches the outbound link language L_{out} of any partition in its partition set, the partition set is expanded to include the adjacent partition reached by that link.

To generate the initial partitioning, we first load the KB axioms into SNARK and convert them to clausal form. The clauses are passed to an external partitioning tool, which generates the partition graph and defines the vocabulary for each partition and link, according to the particular partitioning strategy selected (see Section 2.1). The partition graph is essentially static, and can be generated just once for each KB, so that the cost of partitioning is amortized over many queries. However, the links between partitions are reoriented by MP on a per-query basis, to ensure that all communication links lead toward the goal partition.

3.2 Sample KBs, evaluation metrics

The greatest problem we encountered was obtaining large KBs of FOL suitable for use as test data. Test suites for theorem provers such as TPTP [19] are biased toward propositional logic (PL) or small problems in FOL, while the minority of test problems containing significant numbers of FOL axioms usually describe an abstract domain and employ only a small number of symbols, making them unsuitable for partitioning. The best application of partitioning will be to commonsense KBs that contain large numbers of both axioms and symbols. Unlike most mathematical theorem proving problems, these KBs exhibit structure that can be exploited by partitioning to improve the efficiency of reasoning. However,

the few such KBs publically available tend to employ proprietary extensions to FOL, making their direct comparison in a common testbed problematic.

In the end, we used a subset of the Cyc KB containing 730 axioms (and 150 symbols) concerning spatial relationships, along with a set of queries supplied by an outside source. We are continuing our efforts to remedy the paucity of experimental data by adding other large KBs. We are currently testing queries against the Suggested Upper Merged Ontology (SUMO) [13]; experimental results will be reported at the project website¹.

We collected a variety of statistics on each KB partitioning task and each query trial, including runtimes and elapsed times. But we are most interested in metrics that allow us to compare one trial with another by the size of the search space explored. Processor runtime is an imperfect proxy, since it depends greatly on implementation details that are not our primary concern. Thus, in the remainder of this paper, we focus on *number of resolution steps* as the best measure of the quantity of work required for each trial.

4 Baseline evaluation of MP

As a starting point for evaluating the effectiveness of MP, we compare it with two common resolution restriction strategies: *set-of-support* (SOS) and *ordered resolution*. SOS places the negated query into a designated “set of support” and allows only resolutions involving at least one clause from the set of support, to which newly derived clauses are added. In ordered resolution, a global ordering among predicates is given (by the user), and clauses are resolved upon only on their highest literals (in the predicate ordering).

To assess the effectiveness of a resolution strategy, we examine how much work is required to answer a query using the strategy, relative to using *no* strategy (that is, allowing any possible resolution). As described in Section 3.2, we use the number of resolution steps as a measure of the work done in answering a query. The absolute number of resolution steps is typically in the thousands, but since this number may vary widely from one query to the next, we must normalize in order to make meaningful comparisons across queries. Thus, for each query and strategy, we report the number of resolution steps required to answer the query as a percent of the number of resolution steps required using no strategy.

Results for a representative selection of queries are shown in the first three histograms of Figure 2. Relative to using no strategy, MP reduces the numbers of steps required to answer most queries by one-third to two-thirds.

However, MP is significantly outperformed by SOS on many queries (though not all). Why? This “vanilla” version of MP operates only at the global level: it restricts resolution *between* partitions, but allows unrestricted resolution *within* partitions, equivalent to using no strategy at the local level. In the following sections, we show how to complement MP with smart local strategies for enhanced performance.

The trials for ordered resolution used a default (arbitrary) ordering, so the unimpressive results are no surprise. In Sec-

¹<http://www.ksl.stanford.edu/projects/Partitioning>

tion 5, we introduce a mechanism for automatically inducing efficient orderings from a partition structure.

5 Partition-derived ordering

A major obstacle to the effective use of ordered resolution is finding a good predicate ordering, which will minimize inference. SNARK provides a default ordering, but as we saw in Section 4, it is undistinguished compared to MP and SOS, though better than using no strategy. Efficient ordered resolution is usually achieved by arduous hand-crafting of a predicate ordering by a theorem-proving expert.

In this section, we propose to use automatic partitioning, and in particular PBR tree-decomposition algorithms, to automatically produce a predicate ordering. This is an important contribution as it paves the way to improving theorem-proving software for ordered resolution, providing more effective ordered resolution for the non-expert, and relieving the expert from arduous hand-coding of predicate ordering.

Figure 4 describes the P2O algorithm for inducing a symbol ordering from a computed partitioning. It takes as input a partition graph $G : (V, E, l)$ for partitioned theory $\mathcal{A} : \bigcup_i \mathcal{A}_i$, and a query Q in $\mathcal{L}(\mathcal{A}_k)$, and outputs $Ord(\mathcal{A})$, a symbol ordering on $\mathcal{A}$. We call this ordering a *partition-derived ordering*. When combined with ordered resolution, we call the strategy PDO. Recall that ordered resolution uses an ordering on predicate symbols, so PDO in fact uses $Ord(\mathcal{A})$ restricted to predicate symbols and ignores the ordering imposed on constant and function symbols.

The underlying intuition is as follows: in MP, we perform resolution and message passing starting with the partitions farthest away from the goal partition. All predicates are resolved *except* those in the outbound link language, which are resolved in the next partition downstream, unless they continue to appear in subsequent outbound link languages. In this way, the ordering of partitions along paths toward the goal determines an ordering among predicates resolved.

PROCEDURE P2O(G, Q)

$G : (V, E, l)$ is a partition graph for the partitioned theory $\mathcal{A} : \{\mathcal{A}_i\}_{i \leq n}$ and Q is a query in $\mathcal{L}(\mathcal{A}_k)$ ($k \leq n$).

1. Define $j \prec i$ iff j is on the path between i and k in G .
2. Let $\mathcal{L}_{out_i} = \mathcal{L}(l(i, j))$ for j such that $(i, j) \in E$ and $j \prec i$ (there is at most one such j ; if there is none, then let $\mathcal{L}_{out_i} = \emptyset$). $\mathcal{L}_{out_i}$ is the output language of partition $\mathcal{A}_i$.
3. Let $\mathcal{L}_{in_i} = \bigcup_h \mathcal{L}(l(h, i))$ for h such that $(h, i) \in E$ and $i \prec h$ (there may be none or multiple such h 's). $\mathcal{L}_{in_i}$ is the input language of partition $\mathcal{A}_i$.
4. Initialize $Ord(\mathcal{A}) = \text{empty list}$.
5. Traverse the tree from furthest leaves to root following $\prec$, in a decreasing order, for every $i \leq n$.
 $Ord(\mathcal{A}) = \text{append}(Ord(\mathcal{A}), \mathcal{L}_{in_i} \setminus \mathcal{L}_{out_i}, \mathcal{L}(\mathcal{A}_i) \setminus \mathcal{L}_{out_i})$.
6. $Ord(\mathcal{A}) = \text{append}(Ord(\mathcal{A}), \mathcal{L}(\mathcal{A}_k), \cdot)$.

Figure 4: An algorithm to compute a predicate ordering.

Theorem 5.1. *Let $\mathcal{A} = \bigcup \{\mathcal{A}_i\}_{i \leq n}$ be a partitioning of $\mathcal{A}$ based on predicates only (so all function and constant symbols appear in all partitions), let G be a tree decomposition*

corresponding to $\mathcal{A}$, and let Q be a query. Then, ordered resolution with order $Ord(\mathcal{A})$ is equivalent to MP, where inference within each partition uses ordered resolution with an order compatible with $Ord(\mathcal{A})$.

Proof. For every $p \in L(\mathcal{A})$, let $P(p)$ be the $\prec$ -smallest partition, i , such that $p \in \tilde{L}(i)$. There is exactly one such partition; otherwise there would be two that are not connected, contradicting the fact that G is a tree decomposition (it guarantees that if a symbol appears in two partitions then it appears on the edges on the path between them).

Define $<_0$, an order between symbols, by $p <_0 q$ iff p precedes q in $Ord(\mathcal{A})$ (equivalently, $P(p) \prec P(q)$). Define $<_{\mathcal{A}}$ to be a total order that agrees with $<_0$. Let $\{p_i\}_{i \leq m}$ be an enumeration of the predicate symbols of $\mathcal{A}$ according to $<_{\mathcal{A}}$. Notice that if ordered resolution with $<_{\mathcal{A}}$ resolved two clauses upon p_i , then p_i is the highest symbol in both clauses.

We show that ordered resolution is equivalent to MP by showing that ordered resolution step can be done by MP, and vice versa. Let C_1, C_2 be two clauses resolved on p by ordered resolution using the order $<_{\mathcal{A}}$. Let i_1, i_2 be such that C_1 is in partition i_1 and C_2 is in partition i_2 . Assume that MP cannot resolve the two clauses. That means that they are in different partitions and that at least one of the clauses, say C_2 , should be sent to another partition, $\mathcal{A}_j$, with $j \prec i_2$, in order for the two clauses to be resolved (j is the $\prec$ -largest partition such that $j \prec i_1$ and $j \prec i_2$).

Since C_2 was not sent to partition j , its vocabulary is not in the label of one of the edges on the path between i_2 and p in G . Let q be a symbol that appears in C_2 but is not on one of the edges in the path between partitions i_2 and j . p is the $<_{\mathcal{A}}$ -largest symbol for both C_1 and C_2 , meaning that p appears on the path between i_1, i_2 (thus it also appears on the path between i_2, j , which is a subset of the path between i_1, i_2) and that $q <_{\mathcal{A}} p$ (because q does not appear on that path (thus, it is not p) and p is the $<_{\mathcal{A}}$ -largest in C_2).

Since q is not in $\tilde{L}(j)$ (otherwise it would be on the path between i_2, j) and the path between j, i_2 includes p , it must be that $P(p) \preceq j$, and $P(p) \preceq j \prec P(q)$ implies that $p <_{\mathcal{A}} q$. This contradicts the previous observation that $q <_{\mathcal{A}} p$. Thus, q appears on the path between i_2, j and C_2 must be sent to partition j . The same holds for C_1 . Thus, MP will resolve the two clauses.

The other direction (every resolution in MP is a resolution in ordered resolution) is simpler, and follows from Theorem 5.2 below. $\square$

It is interesting to observe that we can, conversely, generate a theory partitioning from a symbol ordering, should one be given by an expert. The algorithm is as follows: Let S be a stack containing the predicate symbols in logical theory $\mathcal{A}$ in the designated order $Ord(S)$. Let Z be an empty stack. Pop a symbol s from S . Create a set of symbols that includes s , and all the symbols still in S that appear with s in some axiom in $\mathcal{A}$. If this set of symbols is contained by another set on Z , discard it. Otherwise, push it onto Z . Continue until S is empty. Then, set $G : (V, E)$ to be an empty graph, and iteratively pop sets of symbols from Z . For a set of symbols Z_i , add it to V as a neighbor (in G) to a set of symbols in

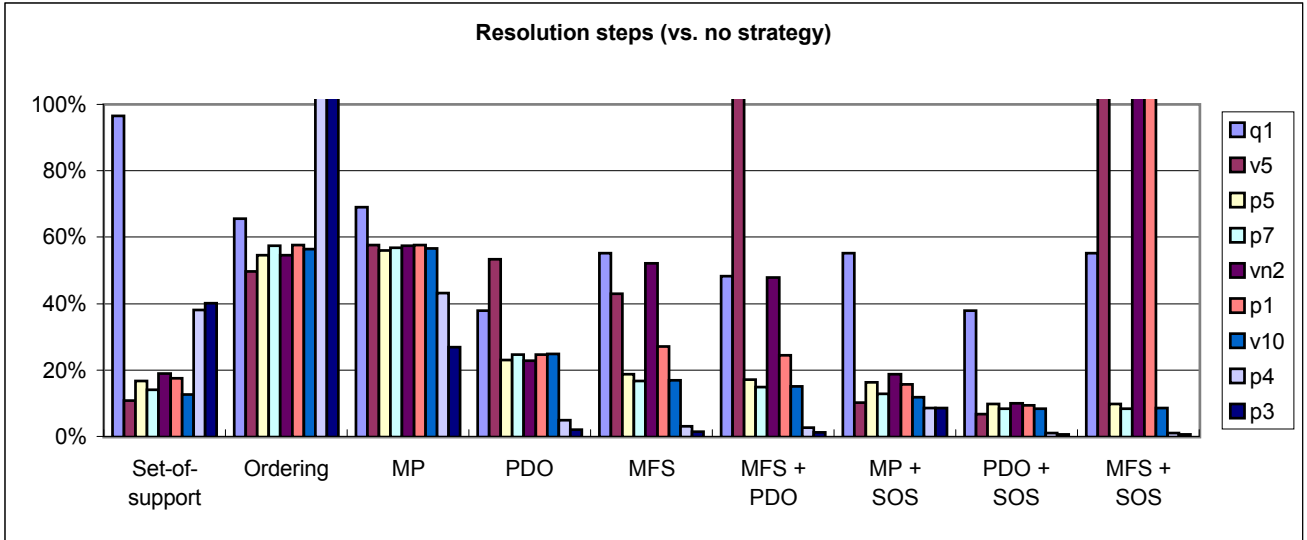


Figure 2: Performance comparison of reasoning strategies: set-of-support (SOS), ordered resolution, vanilla MP, partition-derived ordering (PDO), MP with focused support (MFS), and several combinations of strategies. For a representative selection of queries, we show the number of resolution steps required to answer the query using each strategy, expressed as a percent of the number of steps required using no strategy. Columns exceeding 100% represent timeouts.

V with which it shares the largest number of symbols. When Z is empty, create a partition $\mathcal{A}_i$ for every set of symbols $Z_i \in V$ such that $\mathcal{A}_i$ includes all the axioms from $\mathcal{A}$ that are in $\mathcal{L}(Z_i)$. Call the resulting set of partitions $\mathcal{P}(\text{Ord}(S))$, and the graph $G(\text{Ord}(S))$.

Theorem 5.2. Assume that Ord is an order on the nonlogical symbols of theory $\mathcal{A}$, and that Q is a query such that $\mathcal{L}(Q)$ are the final symbols in order Ord . Then, ordered resolution with order Ord is equivalent to MP with partitions $\mathcal{P}(\text{Ord})$ and graph $G(\text{Ord})$, where inference within each partition uses ordered resolution with an order compatible with $\text{Ord}(\mathcal{A})$.

Proof Sketch. To see that MP does not perform more resolutions than ordered resolution with order $\leq_{\mathcal{A}}$, we only need to notice that every partition corresponds to a *bucket* in the ordered resolution algorithm. If MP resolves C_1, C_2 on predicate P , then C_1, C_2 were together in some partition j , and were resolved on their highest predicate. Thus, C_1, C_2 will also be resolved by ordered resolution with order $\leq_{\mathcal{A}}$. $\square$

Proposition 5.3. *PDO is sound and complete.*

In our experimental evaluation, illustrated in Figure 2, PDO made a strong showing. Its performance was comparable with that of SOS for most queries; for a few queries, PDO was up to two orders of magnitude more efficient than SOS. Unsurprisingly, PDO invariably outperformed ordered resolution with SNARK’s default ordering. PDO also outperformed vanilla MP, which by itself does not focus reasoning at the local level. When MP is augmented with focused support (the MFS combination), PDO’s comparative advantage largely disappears. In Section 7, we will see that the perfor-

mance of PDO can be enhanced even further by combining it with other strategies.

6 A new restriction strategy: MFS resolution

MP operates at the global level to focus reasoning by restricting between-partition resolution and passing relevant results toward the goal partition. However, MP has no impact on local reasoning, that is, reasoning within partitions. In fact, this is one of the strengths of MP: individual partitions may employ heterogeneous reasoning methods, including special-purpose reasoners optimized for particular domains. (As described in Section 2.2, the global soundness and completeness of MP follow from the soundness and completeness of the reasoners used in each partition.) MP can thus be used to orchestrate collaborative reasoning between disparate systems, provided that they share a common vocabulary. Nevertheless, the absence of a local strategy limits gains in efficiency. In this section we exploit PBR to propose a new restriction strategy, *MFS resolution*, that combines global MP with a local strategy for focused sublanguage resolution.

With vanilla MP, all possible resolutions are performed within a partition, but resolvents can be propagated to other partitions only if they are in the outbound link language. We propose a local resolution strategy, *focused support*, that takes inspiration from strategies described by Slagle [17], SOL restriction [8], and SFK-resolution [7].

Definition 6.1 (Focused Support Restriction). Let $\mathcal{T}$ be a clausal theory and let L be a designated subset of $\mathcal{L}(\mathcal{T})$. Initialize S to be the set of clauses in $\mathcal{T}$ that non-exhaustively include symbols from L . Two clauses may be resolved only if one of them is in S and the resolved predicate is not in L . The

Query	no strategy	ordering	SOS	MP	PDO	MFS	MP+SOS	PDO+SOS	MFS+SOS
<i>Number of resolution steps</i>									
q1	29	19	28	20	11	16	16	11	16
v5	4,343	2,157	467	2,504	2,318	1,869	441	294	–
p5	3,575	1,948	598	1,999	826	673	585	348	350
p7	3,958	2,273	556	2,250	972	664	507	332	334
vn2	3,681	2,007	696	2,115	839	1,922	690	364	–
p1	4,182	2,410	729	2,413	1,029	1,131	653	391	–
v10	3,862	2,182	489	2,187	963	656	458	326	328
p4	–	–	13,191	15,004	1,715	1,088	2,938	382	381
p3	–	–	38,943	25,956	1,942	1,285	8,243	594	566
<i>Runtime in seconds</i>									
q1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1	0.1
v5	31.9	10.2	1.7	27.0	12.1	9.8	4.6	1.2	–
p5	24.8	9.8	2.4	18.5	3.3	3.3	5.8	1.7	2.6
p7	26.5	11.6	2.2	21.4	3.9	3.2	5.4	1.6	1.7
vn2	25.1	9.5	2.7	12.4	3.5	13.3	2.7	1.7	–
p1	27.3	14.0	2.8	20.1	4.0	8.2	6.5	1.8	–
v10	26.4	10.8	2.9	13.7	3.7	3.4	2.0	1.5	1.6
p4	–	–	217.5	454.3	9.6	7.1	32.0	1.7	1.8
p3	–	–	1,191.0	1,067.4	8.8	7.9	100.8	2.1	2.1

Figure 3: Absolute performance metrics for several reasoning strategies: no strategy, ordered resolution using a default ordering, set-of-support (SOS), vanilla MP, partition-derived ordering (PDO), MP with focused support (MFS), and combinations. For a representative selection of queries, we show the number of resolution steps and the runtime required to answer the query using each strategy. Blanks indicate timeouts.

resolvent is added to S .

When focused support is used as a local strategy in PBR, clausal theory $\mathcal{T}$ is an individual partition $\mathcal{A}_i$ from a partitioned theory, and L is the output communication language of that partition $\mathcal{A}_i$.

Focused support is sound and complete for generating consequences in the restricted vocabulary of L . The proof relies on the following lemma:

Lemma 6.2 ([17]). *If P is an ordering of the predicate symbols in a finite, unsatisfiable set U of clauses and if M is a model, then there is a semantic P -deduction of the empty clause from U .*

Roughly, P -deduction resolves clauses C_1, C_2 on predicate R only if C_1 satisfies M , C_2 does not satisfy M , and R is the largest among the predicates in C_2 , in some preset order.

Theorem 6.3. *Focused support is sound and complete for L -consequence generation (c.f. Definition 2.1).*

Proof. Let $\mathcal{A}$ be a clausal theory and $D \in \mathcal{L}(L)$ a clause such that $\mathcal{A} \models D$. We show that focused support eventually generates D as a result of $\mathcal{A}$. Assume that $\mathcal{A}$ is consistent. Thus, there is a model M that satisfies $\mathcal{A}$. Let S be as in the definition of the focused support strategy for L .

Lemma 6.2 says that there is a P -resolution of $\{\}$ from $\mathcal{A}$ for the model M and for every ordering P of the predicate symbols. Since everything D resolves with is already in S , we can remove $\neg D$ from the resolution graph of this proof and yield D from the P -deduction of $\mathcal{A}$. This shows that there is a P -deduction of D from $\mathcal{A}$.

To show that there is a focused support deduction of D from $\mathcal{A}$ we still need to show that restricting resolution to occur only on non- L literals does not prevent proving D . Choose P to be an order such that the symbols not in L come before those in L . Sentences that are not in S have no symbols from L . When we resolve sentences from S with those not in S , then clearly the focused support condition holds. We move all the clauses of S that are completely in $\mathcal{L}(L)$ (i.e., have no symbols outside L) into a distinguished set, $\bar{S}$. Now, when we resolve two sentences from $\bar{S}$, both of them have symbols that are not in L . Those symbols take precedence over the ones from L , so the focused support condition is held. If the resolvent has only literals from L , then we put it in $\bar{S}$. Otherwise, we put it in S .

Finally, assume that $\bar{S} \not\models D$. Then, $\bar{S} \cup \{\neg D\}$ is consistent. However, $\bar{S} \cup S \models D$ (as shown above with the P -deduction), so $\bar{S} \cup S \cup \{\neg D\}$ is not consistent. However, for sentences in S , $\bar{S}$ resolution with our semantic support condition is equivalent to using ordered resolution with the order only specifying that literals in L are resolved after literals not in L . Since this is known to be complete for L -consequence finding [20; 7], we know that $\bar{S} \models D$, and the proof is done. $\square$

MFS resolution is the combination of MP with the focused support restriction within partitions. As discussed in Section 2.2, MP is sound and complete provided that each partition uses a reasoning procedure that is sound and complete for L -consequence finding. Since focused support is such a procedure, MFS resolution is also sound and complete:

Corollary 6.4 (Soundness and Completeness of MFS). *MP with the focused support restriction (MFS resolution) is sound and complete.*

The relative performance of MFS, as depicted in Figure 2, is encouraging. For most queries, the performance of MFS is comparable to that of SOS; and for a minority of queries MFS is one or even two orders of magnitude more efficient.

We also examined the performance of MFS combined with PDO (MFS+PDO), which was slightly superior to that of MFS alone. However, unlike MFS, MFS+PDO is incomplete. Section 7 discusses the incomplete strategy MFS+SOS, which exhibited even better performance.

Theorem 6.5 (Incompleteness of MFS+PDO). *PDO with the focused support restriction (MFS+PDO) is incomplete.*

The incompleteness of MFS+PDO can be seen by noticing that focused support is stricter than each of SOS and semantic resolution. When we add PDO to this restriction, we eliminate resolutions with any clause that does not share predicates with the clauses in the negated query. Interestingly, MFS+PDO is very close to being complete, in the following sense. A close examination of the proof of Theorem 6.3 reveals that we can apply a predicate order restriction to clauses that do not contain predicates from L and still maintain completeness for L -consequence finding. Thus, a simple modification of MFS+PDO, allowing clauses in the focused support of a partition to resolve on any non- L literal (without considerations of ordering) is complete.

7 Combinations of strategies

Preceding sections examined three reasoning strategies that are shown to be complete: MP, PDO, and MFS. In this section we examine combinations of these strategies with SOS. Though none of these combinations is complete, they have some interesting theoretical properties, and in practice showed the best performance of all strategies evaluated. Results of experimental evaluation are summarized in Figure 5.

To combine MP with SOS, we maintain a global set of support, which initially contains only the negated query, and we allow resolution between two clauses only if they are in the same partition and at least one of them is in the support. Because messages are sent only toward the goal partition in MP, never away, resolution will occur only in the goal partition unless we place the negated query in every partition whose language includes it. In either case, SOS resolution will occur in partitions in which the negated query appears.

Clearly, the MP+SOS combination is not complete. Information crucial to proving the query may reside in partitions that do not participate in the computation. However, the algorithm determines the clauses that are most relevant to proving the query, in a simple and dynamic way. This sort of inference is used extensively with large KBs, where the selection of the KB fragments needed to answer the query is done by hand. Our automatic selection mechanism may suffice for many applications, and our experiments bear this out.

The argument given for the incompleteness of MP+SOS also shows that PDO+SOS is incomplete. This follows from the relationship between MP and ordered resolution presented

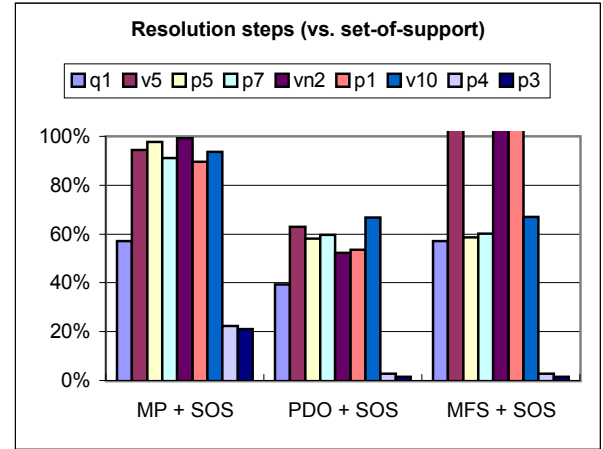


Figure 5: Performance comparison of three combinations of reasoning strategies. In contrast to Figure 2, the vertical axis is normalized relative to SOS.

in Theorem 5.2. When the PDO predicate ordering is determined by the same partitioning being used in MP, then PDO+SOS restricts resolution to only those clauses that include predicates that appear in the support. Despite its incompleteness, PDO+SOS performed extremely well in experimental evaluation. Incompleteness was never encountered in practice, and the PDO+SOS combination answered every query in fewer steps than any other strategy or combination of strategies.

The combination of MFS with SOS (MFS+SOS) is incomplete as well, since MFS is a restricted form of MP, and MP+SOS is incomplete. This incompleteness was revealed in experimental testing, when the MFS+SOS combination failed to answer a few queries. However, when it did not fail, MFS+SOS was highly efficient, answering most queries in as few steps as PDO+SOS.

8 Summary and related work

We propose, analyze, and experimentally evaluate a variety of strategies that improve the efficiency of resolution-based FOL theorem provers. We also propose an algorithm to automatically induce predicate orderings for ordered resolution. Experimental results confirm significant performance improvements using our techniques. This work makes promising practical contributions to the theorem proving community, presented in a manner designed to facilitate replication. (For software and data, see <http://www.ksl.stanford.edu/projects/Partitioning>.) Our results are particularly significant for query answering with commonsense KBs, whose size, structure, and sometimes distributed nature make them well-suited to our structure-based efficiency improvements.

There is a diversity of related work. Our algorithm to induce a predicate ordering is the first to use graphical structure, and in particular treewidth approximation algorithms, with FOL resolution. Current automated approaches to ordering

predicates use properties of those predicates, such as their arity (e.g., the Knuth-Bendix method [10] when applied to a uniform weight function, as in [21]), or generate an arbitrary default ordering (e.g., lexical, as in [18]). Our work is most significantly distinguished from work on CSPs (e.g., [6]) and propositional reasoning (e.g., [5; 15]) in that we partition (and subsequently order) a graph that includes all the nonlogical symbols in the theory, whereas propositional methods order nodes in a graph that correspond to propositional symbols and ordering is often dynamic.

Our *focused support* restriction (on which MFS is based) resembles SFK resolution [7] and SOL resolution [8] in its computation of resolvents in a target language. However, in contrast to SFK resolution, our target language is not closed under subsumption. Further, in [7] there is no clear way to determine a predicate ordering. MFS resolution combines focused support with predicate ordering imposed by MP, generating a significant improvement in performance.

Finally, there has been little experimental work studying the behavior of theorem proving strategies on large KBs, and none on commonsense KBs. The success rate of leading theorem provers, such as SPASS, Otter, Setheo, Protein and 3TAP, in formal verification problems with hundreds of axioms, is shown in [14] to depend strongly on how good they are at finding the few relevant axioms needed in the proofs. Our work presents a principled method to successfully elicit such a set of relevant axioms, the PDO+SOS strategy being most notably successful.

In the future, we plan to continue our experimental evaluation with additional KBs. We also hope to demonstrate the use of MP for collaborative theorem proving among distributed KBs and diverse (perhaps special-purpose) reasoners. Finally, we note that PBR naturally enables parallelization of the FOL theorem proving task. The benefits of partition-driven parallelism can be evaluated with little further implementation effort.

Acknowledgements

We thank Mark Stickel, Vinay Chaudhri, Adam Pease, and Richard Waldinger for their assistance, feedback and comments. This research was supported in part by DARPA grant N66001-97-C-8554-P00004, NAVY grant N66001-00-C-8027, DARPA grant N66001-00-C-8018, ONR MURI Fund N00014-01-1-0890, ONR MURI Fund N00014-00-1-0637, and NSF grant ECS-9873474.

References

- [1] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proc. UAI '01*, pages 7–15. MK, 2001.
- [2] E. Amir and S. McIlraith. Partition-based logical reasoning. In *Proc. KR '2000*, pages 389–400. MK, 2000.
- [3] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
- [4] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Gunning, and M. Burke. The DARPA high-performance knowledge bases project. *AI Magazine*, 19(4):25–49, 1998.
- [5] A. Darwiche. Utilizing knowledge-based semantics in graph-based algorithms. In *Proc. AAAI '96*, pages 607–613, 1996.
- [6] R. Dechter and J. Pearl. Tree clustering schemes for constraint processing. In *Proc. AAAI '88*, 1988.
- [7] A. del Val. A new method for consequence finding and compilation in restricted language. In *Proc. AAAI '99*, pages 259–264. AAAI Press/MIT Press, 1999.
- [8] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2-3):301–353, Aug. 1992.
- [9] G. Karypis. Multilevel algorithms for multi-constraint hypergraph partitioning. Technical Report #99-034, University of Minnesota, Department of Computer Science, Minneapolis, MN, 1999.
- [10] D. E. Knuth and P. B. Bendix. *Simple word problems in universal algebras*, pages 263–297. Pergamon Press, Oxford, 1970.
- [11] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. of the ACM*, 46(6):787–832, 1999.
- [12] S. McIlraith and E. Amir. Theorem proving with structured theories. In *IJCAI '01*, pages 624–631. MK, 2001.
- [13] I. Niles and A. Pease. Towards a standard upper ontology. In *2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 17–19, Oct. 2001. <http://ontology.teknowledge.com>.
- [14] W. Reif and G. Schellhorn. Theorem proving in large theories. In *Proc. FTP'97*, pages 119–124, 1997.
- [15] I. Rish and R. Dechter. Resolution versus search: two strategies for SAT. *J. of Automated Reasoning*, 24(1-2):225–275, 2000.
- [16] D. J. Rose. Triangulated graphs and the elimination process. *J. of Discrete Mathematics*, 7:317–322, 1974.
- [17] J. R. Slagle. Automatic theorem proving with renamable and semantic resolution. *J. of the ACM*, 14(4):687–697, 1967.
- [18] M. E. Stickel, R. J. Waldinger, and V. K. Chaudhri. A guide to SNARK. Technical report, SRI International, 2000.
- [19] G. Sutcliffe and C. B. Suttner. The TPTP problem library: CNF release v1.2.1. *J. of Automated Reasoning*, 21(2):177–203, 1998. <http://www.cs.miami.edu/~tptp>.
- [20] P. Tison. Generalization of consensus theory and application to the minimization of boolean functions. *IEEE Transactions on Electronic Computers*, EC-16:446–456, 1967.
- [21] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobald, and D. Topic. SPASS version 2.0. In *Proc. CADE-18*, pages 275–279. Springer, 2002.

BELIEF REVISION AND UPDATE

On the Revision of Probabilistic Beliefs using Uncertain Evidence

Hei Chan and Adnan Darwiche
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{hei,darwiche}@cs.ucla.edu

Abstract

We revisit the problem of revising probabilistic beliefs using uncertain evidence, and report results on four major issues relating to this problem: How to specify uncertain evidence? How to revise a distribution? Should, and do, iterated belief revisions commute? And how to provide guarantees on the amount of belief change induced by a revision? Our discussion is focused on two main methods for probabilistic revision: Jeffrey’s rule of probability kinematics and Pearl’s method of virtual evidence, where we analyze and unify these methods from the perspective of the questions posed above.

1 Introduction

We consider in this paper the problem of revising beliefs given uncertain evidence, where beliefs are represented using a probability distribution. There are two main methods for revising probabilistic beliefs in this case. The first method is known as *Jeffrey’s rule* and is based on the principle of *probability kinematics*, which can be viewed as a principle for minimizing belief change [Jeffrey, 1965]. The second method is called *virtual evidence* and is proposed by Pearl in the context of belief networks—even though it can be easily generalized to arbitrary probability distributions—and is based on reducing uncertain evidence into certain evidence on some virtual event [Pearl, 1988]. We analyze both of these methods in this paper with respect to the following four questions:

1. How should one specify uncertain evidence?
2. How should one revise a probability distribution?
3. Should, and do, iterated belief revisions commute?
4. What guarantees can be offered on the amount of belief change induced by a particular revision?

Our main findings can be summarized as follows. First, we show that Jeffrey’s rule and Pearl’s method both revise beliefs using the principle of probability kinematics. Jeffrey’s rule explicitly commits to this principle, while Pearl’s method is based on a different principle. Yet, we show that Pearl’s method implies the principle of probability kinematics, leading to the same revision method as that of Jeffrey’s. The difference between Jeffrey’s rule and Pearl’s method is in the

way uncertain evidence is specified. Jeffrey requires uncertain evidence to be specified in terms of the *effect* it has on beliefs once accepted, which is a function of both evidence strength and beliefs held before the evidence is obtained. Pearl, on the other hand, requires uncertain evidence to be specified in terms of its *strength* only. Despite this difference, we show that one can easily translate between the two methods of specifying evidence and provide the method for carrying out this translation.

The multiplicity of methods for specifying evidence also raises an important question: how should informal statements about evidence be captured formally using available methods? For example, what should the following statement translate to: “Seeing these clouds, I believe there is an 80% chance that it will rain?” We will discuss this problem of interpreting informal evidential statements later, where we emphasize its subtlety.

As to the question of iterated belief revision: It is well known that Jeffrey’s rule does not commute; hence, the order in which evidence is incorporated matters [Diaconis & Zabell, 1982]. This has long been perceived as a problem, until clarified recently by the work of Wagner who observed that Jeffrey’s method of specifying evidence is dependent on what is believed before the evidence is obtained and, hence, should not be commutative to start with [Wagner, 2002]. Wagner proposed a method for specifying evidence, based on the notion of *Bayes factor*, and argued that this method specifies only the strength of evidence, and is independent of the beliefs held when attaining evidence. Wagner argued that when evidence is specified in that particular way, iterated revisions should commute. He even showed that combining this method for specifying evidence with the principle of probability kinematics leads to a revision rule that commutes. We actually show that Pearl’s method of virtual evidence is specifying evidence according to Bayes factor, exactly as proposed by Wagner and, hence, corresponds exactly to the proposal he calls for. Therefore, the results we discuss in this paper unify the two main methods for probabilistic belief revision proposed by Jeffrey and Pearl, and show that differences between them amount to a difference in the protocol for specifying uncertain evidence.

Our last set of results relate to the problem of providing guarantees on the amount of belief change induced by a revision. We have recently proposed a distance measure for

bounding belief changes, and showed how one can use it to provide such guarantees [Chan & Darwiche, 2002]. We show in this paper how this distance measure can be computed when one distribution is obtained from another using the principle of probability kinematics. We then show how the guarantees provided by this measure can be realized when applying either Jeffrey’s rule or Pearl’s method, since they both are performing revision based on the principle of probability kinematics.

2 Probability Kinematics and Jeffrey’s Rule

Suppose that we have two probability distributions Pr and Pr' which disagree on the probabilities they assign to a set of mutually exclusive and exhaustive events $\gamma_1, \dots, \gamma_n$, yet:

$$Pr(\alpha \mid \gamma_i) = Pr'(\alpha \mid \gamma_i), \quad (1)$$

for $i = 1, \dots, n$, and for every event α in the probability space. We say here that Pr' is obtained from Pr by *probability kinematics* on $\gamma_1, \dots, \gamma_n$. This concept was proposed by Jeffrey [1965] to capture the notion that even though Pr and Pr' disagree on the probabilities of events γ_i , they agree on their relevance to every event α .

Consider now the problem of revising a probability distribution Pr given uncertain evidence relating to a set of mutually exclusive and exhaustive events $\gamma_1, \dots, \gamma_n$. One method of specifying uncertain evidence is through the *effect* that it would have on beliefs once accepted. That is, we can say that the evidence is such that the probability of γ_i becomes q_i once the evidence is accepted. If we adopt this method of evidence specification, we conclude that there is *only one* distribution Pr' such that:

- $Pr'(\gamma_i) = q_i$ for $i = 1, \dots, n$.
- Pr' is obtained from Pr by probability kinematics on $\gamma_1, \dots, \gamma_n$.

Moreover, this specific distribution is given by:

$$Pr'(\omega) \stackrel{\text{def}}{=} Pr(\omega) \frac{q_i}{Pr(\gamma_i)}, \quad \text{if } \omega \models \gamma_i, \quad (2)$$

where ω is an atomic event, also known as a world, and $\models$ is the logical entailment relationship. This is exactly the distribution that Jeffrey suggests and, hence, this method of revision is known as *Jeffrey’s rule*. We stress here that we are drawing a distinction between the principle of probability kinematics and Jeffrey’s rule, which are often considered synonymous. Specifically, Jeffrey’s rule arises from a combination of two proposals: (1) the principle of probability kinematics, and (2) the specification of uncertain evidence using a posterior distribution. It is possible for one to combine the principle of probability kinematics with other methods for specifying evidence as we discuss later.

It is not hard to show that the above distribution Pr' is indeed obtained from Pr by probability kinematics on $\gamma_1, \dots, \gamma_n$, as it satisfies Equation 1:

$$\begin{aligned} Pr'(\alpha \mid \gamma_i) &= \frac{Pr'(\alpha, \gamma_i)}{Pr'(\gamma_i)} \\ &= \frac{\sum_{\omega \models \alpha, \gamma_i} Pr'(\omega)}{q_i} \end{aligned}$$

$$\begin{aligned} &= \frac{\sum_{\omega \models \alpha, \gamma_i} Pr(\omega) \frac{q_i}{Pr(\gamma_i)}}{q_i} \\ &= \frac{\sum_{\omega \models \alpha, \gamma_i} Pr(\omega)}{Pr(\gamma_i)} \\ &= \frac{Pr(\alpha, \gamma_i)}{Pr(\gamma_i)} \\ &= Pr(\alpha \mid \gamma_i). \end{aligned}$$

Therefore, for any event α , its probability under the new distribution Pr' is:

$$\begin{aligned} Pr'(\alpha) &= \sum_i Pr'(\alpha \mid \gamma_i) Pr'(\gamma_i) \\ &= \sum_i Pr(\alpha \mid \gamma_i) q_i \\ &= \sum_i q_i \frac{Pr(\alpha, \gamma_i)}{Pr(\gamma_i)}, \end{aligned}$$

which is the closed form for Jeffrey’s rule. We now show an example of using Jeffrey’s rule.

Example 1 (Due to Jeffrey) Assume that we are given a piece of cloth, where its color can be one of: green (c_g), blue (c_b), or violet (c_v). We want to know whether, in the next day, the cloth will be sold (s), or not sold ($\bar{s}$). Our original state of belief is given by the distribution Pr :

$$\begin{aligned} Pr(s, c_g) &= .12, & Pr(s, c_b) &= .12, & Pr(s, c_v) &= .32, \\ Pr(\bar{s}, c_g) &= .18, & Pr(\bar{s}, c_b) &= .18, & Pr(\bar{s}, c_v) &= .08. \end{aligned}$$

Therefore, our original state of belief on the color of the cloth (c_g, c_b, c_v) is given by the distribution $(.3, .3, .4)$. Assume that we now inspect the cloth by candlelight, and we want to revise our state of belief on the color of the cloth to the new distribution $(.7, .25, .05)$ using Jeffrey’s rule. If we apply Jeffrey’s rule (Equation 2), we get the new distribution Pr' :

$$\begin{aligned} Pr'(s, c_g) &= .28, & Pr'(s, c_b) &= .10, & Pr'(s, c_v) &= .04, \\ Pr'(\bar{s}, c_g) &= .42, & Pr'(\bar{s}, c_b) &= .15, & Pr'(\bar{s}, c_v) &= .01. \end{aligned}$$

3 Virtual Evidence and Pearl’s Method

The problem of revising a probability distribution under uncertain evidence can be approached from a different perspective than that of probability kinematics. Specifically, when we have uncertain evidence about some mutually exclusive and exhaustive events $\gamma_1, \dots, \gamma_n$, we can interpret that evidence as *hard evidence* on some *virtual event* η , where the relevance of $\gamma_1, \dots, \gamma_n$ to η is uncertain. It is assumed that the virtual event η depends only on the events $\gamma_1, \dots, \gamma_n$ and, therefore, is independent of any other event α given γ_i :

$$Pr(\eta \mid \gamma_i, \alpha) = Pr(\eta \mid \gamma_i). \quad (3)$$

According to this approach, the uncertainty regarding evidence on $\gamma_1, \dots, \gamma_n$ is recast as uncertainty in the relevance of $\gamma_1, \dots, \gamma_n$ to the virtual event η . Specifically, the uncertainty is recast as the likelihood of γ_i given virtual evidence η : $Pr(\eta \mid \gamma_i)$, for $i = 1, \dots, n$.

We next show that the new distribution obtained after accepting the uncertain evidence on $\gamma_1, \dots, \gamma_n$, $Pr(\cdot | \eta)$, is:

$$Pr(\omega | \eta) = Pr(\omega) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}, \text{ if } \omega \models \gamma_i, \quad (4)$$

where $\lambda_1, \dots, \lambda_n$ are ratios chosen such that:

$$\lambda_1 : \dots : \lambda_n = Pr(\eta | \gamma_1) : \dots : Pr(\eta | \gamma_n).$$

Hence, the specific likelihoods $Pr(\eta | \gamma_i)$ are not important here, but their ratios are. This is why this method usually specifies uncertain evidence using a set of likelihood ratios $\lambda_1, \dots, \lambda_n$ [Pearl, 1988]. The derivation of Equation 4 is based on the assumptions underlying the method of virtual evidence given by Equation 3:

$$\begin{aligned} Pr(\omega | \eta) &= \frac{Pr(\omega, \eta)}{Pr(\eta)} \\ &= \frac{Pr(\eta | \omega) Pr(\omega)}{\sum_j Pr(\eta | \gamma_j) Pr(\gamma_j)} \\ &= \frac{Pr(\eta | \gamma_i, \omega) Pr(\omega)}{\sum_j Pr(\eta | \gamma_j) Pr(\gamma_j)} \\ &= \frac{Pr(\eta | \gamma_i) Pr(\omega)}{\sum_j Pr(\eta | \gamma_j) Pr(\gamma_j)} \\ &= Pr(\omega) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}, \text{ if } \omega \models \gamma_i. \end{aligned}$$

The last step is based on $Pr(\eta | \gamma_i) = k\lambda_i$, where k is a constant. For any event α , the new probability after accommodating the virtual evidence is:

$$\begin{aligned} Pr(\alpha | \eta) &= \sum_i Pr(\alpha, \gamma_i | \eta) \\ &= \sum_i Pr(\alpha, \gamma_i) \frac{\lambda_i}{\sum_j \lambda_j Pr(\gamma_j)} \\ &= \frac{\sum_i \lambda_i Pr(\alpha, \gamma_i)}{\sum_j \lambda_j Pr(\gamma_j)}, \end{aligned}$$

which is the closed form for Pearl's method.

The above revision method is a generalization of the method of virtual evidence proposed by Pearl [1988] in the context of belief networks. A belief network is a graphical probabilistic model, composed of two parts: a directed acyclic graph where nodes represent variables, and a set of conditional probability tables (CPTs), one for each variable [Pearl, 1988; Jensen, 2001]. The CPT for variable X with parents $\mathbf{U}$ defines a set of conditional probabilities of the form $Pr(x | \mathbf{u})$, where x is a value of variable X , and $\mathbf{u}$ is an instantiation of parents $\mathbf{U}$. Suppose now that we have some virtual evidence bearing on variable Y , which has values $y_1, \dots, y_n$. This virtual evidence is represented in the belief network by adding a dummy node Z and a directed edge $Y \rightarrow Z$, where one value of Z , say z , corresponds to the virtual event η . This ensures the assumption of Equation 3, that virtual event z is independent of every other event α given every y_i , i.e., $Pr(z | y_i, \alpha) = Pr(z | y_i)$, which follows from the independence semantics of belief networks [Pearl, 1988].

The uncertainty of evidence is quantified by the likelihood ratios: $Pr(z | y_1) : \dots : Pr(z | y_n) = \lambda_1 : \dots : \lambda_n$, which are specified in the CPT of variable Z . Finally, we incorporate the presence of the virtual event z by adding the observation $Z = z$ to the rest of evidence in the belief network. We now show a simple example.

Example 2 (Due to Pearl) We are given a belief network with two variables: A represents whether the alarm of Mr. Holmes' house goes off, and B represents whether there is a burglary. To represent the influence between the two variables, there is a directed edge $B \rightarrow A$. The CPTs of A and B are given by: $Pr(a | b) = .95$, meaning the alarm goes off if there is a burglary with probability .95; $Pr(a | \bar{b}) = .01$, meaning the alarm goes off if there is no burglary with probability .01; and $Pr(b) = 10^{-4}$, meaning on any given day, there is a burglary on any given house with probability 10^{-4} .

One day, Mr. Holmes' receives a call from his neighbor, Mrs. Gibbons, saying she may have heard the alarm of his house going off. Since Mrs. Gibbons suffers from a hearing problem, Mr. Holmes concludes that there is an 80% chance that Mrs. Gibbons did hear the alarm going off. According to the method of virtual evidence, this uncertain evidence can be specified by the virtual event η and the likelihood ratio: $Pr(\eta | a) : Pr(\eta | \bar{a}) = 4 : 1$. To incorporate the virtual evidence into the belief network, we add the variable Z , and the directed edge $A \rightarrow Z$, and specify the CPT of Z such that $Pr(z | a) : Pr(z | \bar{a}) = 4 : 1$. For example, we can assign $Pr(z | a) = .4$ and $Pr(z | \bar{a}) = .1$. After incorporating the virtual evidence by adding the observation $Z = z$ to the evidence, we can easily compute the answers to queries in the belief network. For example, the probability that there is a burglary at Mr. Holmes' house is now $Pr(b | z) \approx 3.85 \times 10^{-4}$.

4 Comparing the Revision Methods

From the illustrations of the two belief revision methods, Jeffrey's rule and Pearl's method of virtual evidence, we can see that a belief revision method can be broken into two parts: a formal method of specifying uncertain evidence, and a principle of belief revision that commits to a unique distribution among many which satisfy the uncertain evidence.

4.1 Pearl's method and Probability Kinematics

We now show that Pearl's method, like Jeffrey's rule, also obeys the principle of probability kinematics; what they differ in is how uncertain evidence is specified.

Suppose that a probability distribution Pr was revised using the method of virtual evidence, with likelihood ratios $\lambda_1, \dots, \lambda_n$ bearing on events $\gamma_1, \dots, \gamma_n$, obtaining the new distribution $Pr(\cdot | \eta)$. We can easily see that the revision satisfies the principle of probability kinematics, i.e. Equation 1:

$$\begin{aligned} Pr(\alpha | \gamma_i, \eta) &= \frac{Pr(\alpha, \gamma_i | \eta)}{Pr(\gamma_i | \eta)} \\ &= \frac{Pr(\alpha, \gamma_i) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}}{Pr(\gamma_i) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}} \end{aligned}$$

$$\begin{aligned}
&= \frac{Pr(\alpha, \gamma_i)}{Pr(\gamma_i)} \\
&= Pr(\alpha \mid \gamma_i).
\end{aligned}$$

Therefore, both Jeffrey's rule and Pearl's method obey the principle of probability kinematics for belief revision.

4.2 From Pearl's Method to Jeffrey's Rule

With the previous result, we now show how we can easily translate between the two methods of specifying uncertain evidence. For example, to translate from Pearl's method to Jeffrey's rule, we note that the new probabilities of events $\gamma_1, \dots, \gamma_n$ after applying virtual evidence, with likelihood ratios $\lambda_1, \dots, \lambda_n$, are given by:

$$\begin{aligned}
Pr(\gamma_i \mid \eta) &= \sum_{\omega \models \gamma_i} Pr(\omega \mid \eta) \\
&= \sum_{\omega \models \gamma_i} Pr(\omega) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j} \\
&= Pr(\gamma_i) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}.
\end{aligned}$$

Suppose instead that we want to revise the distribution Pr using Jeffrey's rule, assuming that after accepting uncertain evidence, the probability of γ_i becomes:

$$q_i = Pr(\gamma_i) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}.$$

Substituting the above probability q_i in Jeffrey's rule (Equation 2), we get:

$$\begin{aligned}
Pr'(\omega) &= Pr(\omega) \frac{Pr(\gamma_i) \sum_j \lambda_j}{Pr(\gamma_i)} \\
&= Pr(\omega) \frac{\lambda_i}{\sum_j Pr(\gamma_j) \lambda_j}, \text{ if } \omega \models \gamma_i,
\end{aligned}$$

which is exactly the distribution obtained by the method of virtual evidence (Equation 4). We now illustrate this translation by revisiting Example 2.

Example 3 In Example 2, we applied the method of virtual evidence on Pr , by specifying the virtual evidence: $Pr(\eta \mid a) : Pr(\eta \mid \bar{a}) = \lambda_a : \lambda_{\bar{a}} = 4 : 1$. The original probabilities of a and $\bar{a}$ are given by: $Pr(a) = .010094$, and $Pr(\bar{a}) = .989906$. After applying the virtual evidence, the new probabilities of a and $\bar{a}$ are given by:

$$\begin{aligned}
Pr(a \mid \eta) &= Pr(a) \frac{\lambda_a}{Pr(a) \lambda_a + Pr(\bar{a}) \lambda_{\bar{a}}} \approx .039189; \\
Pr(\bar{a} \mid \eta) &= Pr(\bar{a}) \frac{\lambda_{\bar{a}}}{Pr(a) \lambda_a + Pr(\bar{a}) \lambda_{\bar{a}}} \approx .960811.
\end{aligned}$$

Alternatively, we can apply Jeffrey's rule to obtain the new distribution Pr' such that $Pr'(a) = Pr(a \mid \eta) \approx .039189$ and $Pr'(\bar{a}) = Pr(\bar{a} \mid \eta) \approx .960811$, and Pr' will be the same to the distribution $Pr(\cdot \mid \eta)$ obtained by virtual evidence.

4.3 From Jeffrey's Rule to Pearl's Method

We can also easily translate from Jeffrey's rule to Pearl's method. Specifically, the new probabilities of events $\gamma_1, \dots, \gamma_n$ after applying Jeffrey's rule are given by:

$$Pr'(\gamma_i) = q_i.$$

Suppose instead that we want to revise the distribution Pr using Pearl's method. We can do this by applying the method of virtual evidence, with likelihood ratios $\lambda_1, \dots, \lambda_n$, such that:

$$\lambda_1 : \dots : \lambda_n = \frac{q_1}{Pr(\gamma_1)} : \dots : \frac{q_n}{Pr(\gamma_n)}.$$

Therefore, $\lambda_i = kq_i/Pr(\gamma_i)$, where k is a constant. Substituting λ_i in Pearl's method (Equation 4), we get:

$$\begin{aligned}
Pr(\omega \mid \eta) &= Pr(\omega) \frac{\frac{kq_i}{Pr(\gamma_i)}}{\sum_j Pr(\gamma_j) \frac{kq_j}{Pr(\gamma_j)}} \\
&= Pr(\omega) \frac{q_i}{Pr(\gamma_i)}, \text{ if } \omega \models \gamma_i,
\end{aligned}$$

because $\sum_j q_j = 1$. This is exactly the distribution obtained by Jeffrey's rule (Equation 2). We now illustrate this translation by revisiting Example 1.

Example 4 In Example 1, we applied Jeffrey's rule on the original distribution Pr to obtain new distribution Pr' . We can also do this by applying the method of virtual evidence. We can interpret the inspection of the cloth by candlelight as virtual evidence η , where $Pr(\eta \mid c_g) : Pr(\eta \mid c_b) : Pr(\eta \mid c_v) = \lambda_g : \lambda_b : \lambda_v = 7 : 2.5 : .375$.¹ The probability distribution of the color of the cloth after applying virtual evidence is therefore given by:

$$\begin{aligned}
Pr(c_g \mid \eta) &= \frac{Pr(c_g) \lambda_g}{Pr(c_g) \lambda_g + Pr(c_b) \lambda_b + Pr(c_v) \lambda_v} = .7; \\
Pr(c_b \mid \eta) &= \frac{Pr(c_b) \lambda_b}{Pr(c_g) \lambda_g + Pr(c_b) \lambda_b + Pr(c_v) \lambda_v} = .25; \\
Pr(c_v \mid \eta) &= \frac{Pr(c_v) \lambda_v}{Pr(c_g) \lambda_g + Pr(c_b) \lambda_b + Pr(c_v) \lambda_v} = .05.
\end{aligned}$$

We can easily verify that this probability distribution $Pr(\cdot \mid \eta)$ obtained by virtual evidence is the same as the distribution Pr' obtained by Jeffrey's rule.

5 Interpreting Evidential Statements

The evidence specification protocols adopted by Jeffrey's rule and Pearl's method have been discussed by Pearl [2001], in relation to the problem of formally interpreting evidential statements. Consider the following statement as an example:

Looking at this evidence, I am willing to bet 2:1 that David is not the killer.

This statement can be formally interpreted using either protocol. For example, if α denotes the event "David is not the killer," this statement can be interpreted in two ways:

¹The λ 's are chosen such that $\lambda_i = kPr'(c_i)/Pr(c_i)$, where k is a constant.

1. After accepting the evidence, the probability that David is not the killer becomes twice the probability that David is the killer: $Pr'(\alpha) = 2/3$ and $Pr'(\bar{\alpha}) = 1/3$.
2. The probability that I will see this evidence η given that David is not the killer is twice the probability that I will see it given that David is the killer: $Pr(\eta | \alpha) : Pr(\eta | \bar{\alpha}) = 2 : 1$.

The first interpretation translates directly into a formal piece of evidence, Jeffrey's style, and can be characterized as an "All things considered" interpretation since it is a statement about the agent's final beliefs, which are a function of both his prior beliefs and the evidence [Pearl, 2001]. On the other hand, the second interpretation translates directly into a formal piece of evidence, Pearl's style, and can be characterized as a "Nothing else considered" interpretation since it is a statement about the evidence only [Pearl, 2001].

The two interpretations can lead to contradictory conclusions about the evidence. For example, if we use the "Nothing else considered" approach to interpret our statement, we will conclude that the evidence is against David being the killer. However, if we use the "All things considered" interpretation, it is not clear whether the evidence is for or against, unless we know the original probability that David is the killer. If, for example, David is one of four suspects who are equally likely to be the killer, we will originally have: $Pr(\alpha) = 3/4$. Therefore, this evidence has actually increased the probability that David is the killer! Because of this, Pearl argues for the "Nothing else considered" interpretation, as it provides a summary of the evidence and the evidence alone, and discusses how people tend to use betting odds to quantify their beliefs even when they are based on the evidence only [Pearl, 2001].

Example 2 provides another opportunity to illustrate the subtlety involved in interpreting evidential statements. The evidential statement in this case is "Mr. Holmes concludes that there is an 80% chance that Mrs. Gibbons did hear the alarm going off." Interpreting this statement using the "All things considered" approach gives us the conclusion that $Pr'(a) : Pr'(\bar{a}) = 4 : 1$, where a denotes the event that the alarm has gone off. This interpretation assumes that the 4 : 1 ratio applies to the posterior beliefs in a and $\bar{a}$, after Mr. Holmes has accommodated the evidence provided by Mrs. Gibson. However, in Example 2, this statement was given a "Nothing else considered" interpretation, as by Pearl [1988, Page 44-47], where the 4 : 1 ratio is taken as a quantification of the evidence strength. That is, the statement is interpreted as $Pr(\eta | a) : Pr(\eta | \bar{a}) = 4 : 1$, where η stands for the evidence. In fact, the two interpretations will lead to two different probability distributions and, hence, give us different answers to further probabilistic queries. For example, if we use the "All things considered" approach in interpreting this evidential statement, the probability of having a burglary will be $Pr'(b) = 7.53 \times 10^{-3}$, which is much larger than the probability we get using the "Nothing else considered" approach in Example 2, which is 3.85×10^{-4} .

From the discussions above, the formal interpretation of evidential statements appears to be a non-trivial task, which can be sensitive to context and communication protocols. Re-

gardless of how this is accomplished though, we need to stress that the process of mapping an informal evidential statement into a revised probability distribution involves three distinct elements:

1. One must adopt a formal method for specifying evidence.
2. One must interpret the informal evidential statement, by translating it into a formal piece of evidence.
3. One must apply a revision, by mapping the original probability distribution and formal piece of evidence into a new distribution.

Our main point here is that Jeffrey's rule and Pearl's method employs the same belief revision principle, i.e. probability kinematics. Moreover, although they adopt different, formal methods for specifying evidence, one can translate between the two methods of specification. Finally, one may take the view that Jeffrey's rule and Pearl's method constitute commitments to how informal evidential statements should be interpreted, making the first and second elements the same, but we do not take this view here. Instead, we regard the issue of interpretation as a third, orthogonal dimension which is best addressed independently.

6 Commutativity of Iterated Revisions

We now discuss the problem of commutativity of iterated revisions, that is, whether the order in which we incorporate uncertain evidence matters.²

It is well known that iterated revisions by Jeffrey's rule are not commutative [Diaconis & Zabell, 1982]. As a simple example, assume that we are given a piece of uncertain evidence which suggests that the probability of event α is .7, followed by another piece of uncertain evidence which suggests that the probability of α is .8. After incorporating both pieces in that order, we will believe that the probability of α is .8. However, if the reversed order of revision is employed, we will believe that the probability of α is .7. In general, even if we are given pieces of uncertain evidence on different events, iterated revisions by Jeffrey's rule are not commutative.

This was viewed as a problematic aspect of Jeffrey's rule for a long time, until clarified recently by Wagner [2002]. First, Wagner observed and stressed that the evidence specification method adopted by Jeffrey is suitable for the "All things considered" interpretation of evidential statements. Moreover, he argued convincingly that when evidential statements carry this interpretation, they must not be commutative to start with. So the lack of commutativity is not a problem of the revision method, but a property of the method used to specify evidence. In fact, Wagner suggested a third method for specifying evidence based on Bayes factors [Good, 1950; 1983; Jeffrey, 1992], which leads to commutativity. Specifically, if Pr and Pr' are two probability distributions, and

²There is a key distinction between iterated revisions using certain evidence versus uncertain evidence. In the former case, pieces of evidence may be logically inconsistent, which adds another dimension of complexity to the problem [Darwiche & Pearl, 1997], leading to different properties and treatments.

γ_i and γ_j are two events, the *Bayes factor* (or odds factor), $\mathcal{F}_{Pr', Pr}(\gamma_i : \gamma_j)$, is defined as the ratio of new-to-old odds:

$$\mathcal{F}_{Pr', Pr}(\gamma_i : \gamma_j) \stackrel{\text{def}}{=} \frac{Pr'(\gamma_i)/Pr'(\gamma_j)}{Pr(\gamma_i)/Pr(\gamma_j)}.$$

Given this notion, one can specify uncertain evidence on a set of mutually exclusive and exhaustive events $\gamma_1, \dots, \gamma_n$ by specifying the Bayes factor for every pair of events γ_i and γ_j . One interesting property of this method of specification is that Bayes factors do not constrain the probability distribution Pr , i.e., any evidence specified in this way is compatible with every distribution Pr .³ Hence, they are suitable for a “Nothing else considered” interpretation of evidential statements.

Interestingly enough, Wagner [2002] showed that when evidence is specified using Bayes factors and revisions are accomplished by probability kinematics, belief revision becomes commutative.⁴ This was an illustration that Jeffrey’s rule is indeed composed of two independent elements: an evidence specification method, and a revision method.

In fact, we now show that the evidence specification method adopted by Pearl’s method corresponds to the method of specifying evidence by Bayes factors. This has a number of implications. First, it shows that revisions by the virtual evidence method are commutative. Second, it provides an alternate, more classical, semantics for the virtual evidence method. Finally, it shows again that Jeffrey’s rule and Pearl’s method both revise distributions using probability kinematics.

Specifically, suppose that we revise a probability distribution Pr using the method of virtual evidence, with likelihood ratios $\lambda_1, \dots, \lambda_n$ bearing on events $\gamma_1, \dots, \gamma_n$, and suppose that we get distribution Pr' as a result. We then have:

$$\begin{aligned} \mathcal{F}_{Pr', Pr}(\gamma_i : \gamma_j) &= \frac{Pr'(\gamma_i)/Pr'(\gamma_j)}{Pr(\gamma_i)/Pr(\gamma_j)} \\ &= \frac{\frac{Pr(\gamma_i)\lambda_i/(\sum_k Pr(\gamma_k)\lambda_k)}{Pr(\gamma_j)\lambda_j/(\sum_k Pr(\gamma_k)\lambda_k)}}{\frac{Pr(\gamma_i)}{Pr(\gamma_j)}} \\ &= \frac{\lambda_i}{\lambda_j}. \end{aligned}$$

That is, if we specify evidence by Bayes factor λ_i/λ_j for every pair of events γ_i and γ_j , and then revise distribution Pr using probability kinematics, we obtain the same distribution that arises from applying the virtual evidence method.

7 Bounding Belief Change Induced by Probability Kinematics

One important question relating to belief revision is that of measuring the extent to which a revision disturbs existing

³This is not true if we use ratios of probabilities instead of ratios of odds. For example, if $Pr'(\alpha) = 2Pr(\alpha)$, we must have $Pr(\alpha) \leq .5$ because $Pr'(\alpha) \leq 1$ [Wagner, 2002].

⁴Wagner shows not only that the representation of uncertain evidence using Bayes factors is sufficient for commutativity, but in a large number of cases, necessary.

beliefs. We have recently proposed a distance measure defined between two probability distributions which can be used to bound the amount of belief change induced by a revision [Chan & Darwiche, 2002]. We review this measure next and then use it to provide some guarantees on any revision which is based on probability kinematics.⁵

Definition 1 [Chan & Darwiche, 2002] *Let Pr and Pr' be two probability distributions over the same set of worlds ω . We define a measure $D(Pr, Pr')$ as follows:*

$$D(Pr, Pr') \stackrel{\text{def}}{=} \ln \max_{\omega} \frac{Pr'(\omega)}{Pr(\omega)} - \ln \min_{\omega} \frac{Pr'(\omega)}{Pr(\omega)},$$

where $0/0$ is defined as 1.

This distance measure can also be expressed using the Bayes factor:

$$D(Pr, Pr') = \ln \max_{\omega_i, \omega_j} \mathcal{F}_{Pr', Pr}(\omega_i : \omega_j).$$

This measure satisfies the three properties of distance: positiveness, symmetry, and the triangle inequality. It is useful to compute this distance measure between two probability distributions as it allows us to bound the difference in beliefs captured by them.

Theorem 1 [Chan & Darwiche, 2002] *Let Pr and Pr' be two probability distributions over the same set of worlds. Let α and β be two events. We then have:*

$$e^{-D(Pr, Pr')} \leq \frac{O'(\alpha | \beta)}{O(\alpha | \beta)} \leq e^{D(Pr, Pr')},$$

where $O(\alpha | \beta) = Pr(\alpha | \beta)/Pr(\bar{\alpha} | \beta)$ is the odds of event α given β with respect to Pr , and $O'(\alpha | \beta) = Pr'(\alpha | \beta)/Pr'(\bar{\alpha} | \beta)$ is the odds of event α given β with respect to Pr' .⁶ The bound is tight in the sense that for every pair of distributions Pr and Pr' , there are events α and β such that:

$$\frac{O'(\alpha | \beta)}{O(\alpha | \beta)} = e^{D(Pr, Pr')}; \quad \frac{O'(\bar{\alpha} | \beta)}{O(\bar{\alpha} | \beta)} = e^{-D(Pr, Pr')}.$$

According to Theorem 1, if we are able to compute the distance measure between the original and revised distributions, we can get a tight bound on the new belief in any conditional event given our original belief in that event. The following theorem computes this distance measure for belief revision methods based on probability kinematics.

Theorem 2 *If Pr' comes from Pr by probability kinematics on $\gamma_1, \dots, \gamma_n$, the distance measure between Pr and Pr' is given by:*

$$D(Pr, Pr') = \ln \max_i \frac{Pr'(\gamma_i)}{Pr(\gamma_i)} - \ln \min_i \frac{Pr'(\gamma_i)}{Pr(\gamma_i)}.$$

⁵The results in this section are reformulations of previous results [Chan & Darwiche, 2002], and are inspired by a new understanding of Jeffrey’s rule and Pearl’s method as two specific instances of revision based on probability kinematics, and the understanding of Pearl’s method in terms of Bayes factors.

⁶Of course, we must have $Pr(\beta) \neq 0$ and $Pr'(\beta) \neq 0$ for the odds to be defined.

Theorem 2 allows us to compute the distance measure for revisions based on Jeffrey’s rule and virtual evidence.

Corollary 1 *If Pr' comes from Pr by applying Jeffrey’s rule to events $\gamma_1, \dots, \gamma_n$, with posterior probabilities $q_1, \dots, q_n$, the distance measure between Pr and Pr' is given by:*

$$D(Pr, Pr') = \ln \max_i \frac{q_i}{Pr(\gamma_i)} - \ln \min_i \frac{q_i}{Pr(\gamma_i)}.$$

Corollary 2 *If Pr' comes from Pr by applying the method of virtual evidence, with likelihood ratios $\lambda_1, \dots, \lambda_n$ bearing on $\gamma_1, \dots, \gamma_n$, the distance measure between Pr and Pr' is given by:*

$$D(Pr, Pr') = \ln \max_i \lambda_i - \ln \min_i \lambda_i.$$

The significance of Corollaries 1 and 2 is that we can compute the distance measure easily in both cases. For Jeffrey’s rule, we can compute the distance measure by knowing only the prior and posterior probabilities of events $\gamma_1, \dots, \gamma_n$. For Pearl’s method, we can compute the distance measure by knowing only the likelihood ratios $\lambda_1, \dots, \lambda_n$. For both revision methods, the distance measure can be computed in constant time from the uncertain evidence, and we can guarantee a bound on the belief change due to the revision principle of probability kinematics, without explicitly knowing the revised probability distribution.

We close this section by showing that the principle of probability kinematics is optimal in a very precise sense: it commits to a probability distribution that minimizes our distance measure.

Theorem 3 *If Pr' comes from Pr by probability kinematics on $\gamma_1, \dots, \gamma_n$, it is optimal in the following sense. Among all possible distributions that agree with Pr' on the probabilities of events $\gamma_1, \dots, \gamma_n$, Pr' is the closest to Pr according to the distance measure given by Definition 1.*

8 Conclusion

In this paper, we analyzed two main methods for revising probability distributions given uncertain evidence: Jeffrey’s rule and Pearl’s method of virtual evidence. We showed that the two methods use the same belief revision principle, i.e. probability kinematics, with their difference being only in the manner in which they specify uncertain evidence, and showed how to translate between the two methods for specifying evidence. We also discussed the much debated problem of interpreting evidential statements. Moreover, we showed that the method of virtual evidence can be reformulated in terms of Bayes factors, which implies a number of results, including the commutativity of revisions based on this method. Finally, we showed that revisions based on probability kinematics are optimal in a very specific way, and pointed to a distance measure for bounding belief change triggered by any revision based on probability kinematics. Our bounds included Jeffrey’s rule and Pearl’s method as special cases.

Acknowledgments

This work has been partially supported by NSF grant IIS-9988543, MURI grant N00014-00-1-0617, and DiMI grant

00-10065. We also wish to thank Carl Wagner, Judea Pearl, and the late Richard Jeffrey, for their valuable comments and discussions.

References

- [Chan & Darwiche, 2002] Hei Chan and Adnan Darwiche. A distance measure for bounding probabilistic belief change. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 539–545, Edmonton, Canada, July 2002.
- [Darwiche & Pearl, 1997] Adnan Darwiche and Judea Pearl. On the logic of iterated belief revision. *Artificial Intelligence* 87(1–2):1–29, January 1997.
- [Diaconis & Zabell, 1982] Persi Diaconis and Sandy L. Zabell. Updating subjective probability. *Journal of the American Statistical Association* 77:822–830, 1982.
- [Good, 1950] Irving J. Good. *Probability and the Weighing of Evidence*. Charles Griffin, London, 1950.
- [Good, 1983] Irving J. Good. *Good Thinking: the Foundations of Probability and Its Applications*. University of Minnesota Press, Minneapolis, 1983.
- [Jeffrey, 1965] Richard C. Jeffrey. *The Logic of Decision*. McGraw-Hill, New York, 1965. (2nd edition) University of Chicago Press, Chicago, 1983. (Paperback correction) 1990.
- [Jeffrey, 1992] Richard C. Jeffrey. *Probability and the Art of Judgement*. Cambridge University Press, Cambridge, 1992.
- [Jensen, 2001] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer–Verlag, New York, 2001.
- [Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, 1988.
- [Pearl, 2001] Judea Pearl. On two pseudo-paradoxes in Bayesian analysis. *Annals of Mathematics and Artificial Intelligence* 32:171–177, 2001.
- [Wagner, 2002] Carl Wagner. Probability kinematics and commutativity. *Philosophy of Science* 69:266–278, 2002.

Quantifying information and contradiction in propositional logic through test actions

Sébastien Konieczny

IRIT - UPS

F-31062 Toulouse - France

konieczny@irit.fr

Jérôme Lang

IRIT - UPS

F-31062 Toulouse - France

lang@irit.fr

Pierre Marquis

CRIL - Université d'Artois

F-62307 Lens - France

marquis@cril.univ-artois.fr

Abstract

Degrees of information and of contradiction are investigated within a uniform propositional framework, based on test actions. We consider that the degree of information of a propositional formula is based on the cost of actions needed to identify the truth values of each atomic proposition, while the degree of contradiction of a formula is based on the cost of actions needed to make the formula classically consistent. Our definitions are to a large extent independent of the underlying propositional logic; this flexibility is of prime importance since there is no unique, fully accepted logic for reasoning under inconsistency.

1 Introduction

Information and contradiction are two fundamental aspects of knowledge processing. Quantifying them is an important issue when reasoning about beliefs (or preferences) stemming from one or different sources. Here are some contexts where quantifying information and contradiction is relevant:

- *diagnosis and testing*. In model-based diagnosis, some initial assumptions that each component works correctly are made; these assumptions may conflict with actual observations. Measuring the conflict of the resulting base may be a good hint about how hard it will be to identify the faulty components.
- *preference elicitation*. In the interactive process of eliciting the preference profile of an individual (user) about a set of possible alternatives, it is not unfrequent that contradictory preferences arise. In this situation, it is useful to quantify and localize the contradictions as well as the information about the user's preferences, so as to be in position to choose the next questions to ask.
- *belief merging*. In this framework, degrees of information and contradiction can be the basis on which one can decide whether to take or not into account the data conveyed by an agent. If the degree of contradiction of the data given by an agent is high, it may be relevant to reject the information, since there is a significant evidence that the source is not reliable; however, this must be balanced by the quantity of information furnished by the agent, especially when she also gives some important and uncontroversial pieces of information.
- *group decision making*. Contradictions arise frequently

when trying to reach a compromise among several agents who have possibly conflictual preferences about a common decision (like voting, resource sharing, public goods buying). In this context, not only it is relevant to compute a global degree of conflict (among the set of agents) but also degrees of conflicts associated with small groups of agents (*coalitions*) so as to localize as precisely as possible where the conflicts are.

Now, what do “degree of information” and “degree of contradiction” mean? There is no consensus about it. The main features shared by existing definitions (and there are not numerous, cf. Section 7) is that (1) such degrees are numerical values, and (2) they vary depending on the representation language. Thus, one may consider a as fully informative in the case where a is the single atomic proposition of the language but surely not fully informative when the vocabulary also contains b (provided that a and b are independent propositions).

In this paper, our point of view is that it is inadequate to quantify the information/contradiction conveyed by some data without considering at the same time a set of available actions and a goal to be reached. Accordingly, our degrees of information and contradiction are defined in an “active” way. Acting so as to reduce inconsistency or to gain information often relies on performing *knowledge-gathering* actions (also called *tests*). We consider that the degree of information of an information base is based on the number (or the cost) of actions needed to identify the truth value of each atomic proposition (the lower the cost the more informative the base); and that the degree of contradiction of an information base is based on the number (or the cost) of actions needed to render the base classically consistent. Thus, both degrees are dependent on the language but also on the given set of tests and the way plans costs are computed.

The rest of this paper is organized as follows. After some formal preliminaries in Section 2, we present our framework in Section 3. In order to show the generality of our framework, we instantiate it to three different propositional logics: classical logic (Section 4), the paraconsistent logic LP_m (Section 5) and a syntax-based approach to inconsistency handling (Section 6). Related work is given in Section 7, and some conclusions in Section 8.

2 Formal preliminaries and notations

We consider a propositional language $\mathcal{L}_{PS}$ based on a finite set of propositional symbols PS and a set of connectives that may vary depending on the logic used. Well-formed formulas

are denoted by φ, ψ , etc. The available information is represented by a formula Σ of $\mathcal{L}_{PS}$ (called *information base*).

Since we wish to keep a reasonable level of generality, we do not want to commit right now to a particular propositional logic. Indeed, flexibility is a major feature of our framework, and it is essential at least for two reasons. On the one hand, classical logic is inadequate to handle contradiction: *ex falso quodlibet sequitur* (a single, local contradiction is sufficient to pollute the whole information base). On the other hand, there is no unique, fully accepted, logic for reasoning under inconsistency. This is reflected by the various approaches that can be found in the literature under different names like belief revision, merging, paraconsistent logics, etc.

$\mathcal{L}_{PS}$ is required to possess the following ingredients:

1. A *consequence relation* $\models_L$ on $\mathcal{L}_{PS} \times \mathcal{L}_{PS}$.
2. An *acceptance function* $A_L \subseteq \mathcal{L}_{PS} \times \mathcal{L}_{PS}$: $A_L(\Sigma, \varphi)$ means that given the information base Σ , φ is accepted as a true information (we say that Σ *accepts* φ). By default, acceptance is defined by: $A_L(\Sigma, \varphi)$ iff $(\Sigma \models_L \varphi$ and $\Sigma \not\models_L \neg\varphi)$. We say that Σ is *informative about* φ iff exactly one of $A_L(\Sigma, \varphi)$ or $A_L(\Sigma, \neg\varphi)$ holds, and that Σ is *fully informative* iff for any $\varphi \in \mathcal{L}_{PS}$, Σ is informative about φ .
3. A *contradiction indicator* $C_L \subseteq \mathcal{L}_{PS} \times \mathcal{L}_{PS}$: if $C_L(\Sigma, \varphi)$ holds, then we say that Σ is *contradictory about* φ . By default, we define $C_L(\Sigma, \varphi)$ iff $(\Sigma \models_L \varphi$ and $\Sigma \models_L \neg\varphi)$. Σ is said to be *contradiction-free* iff for every $\varphi \in \mathcal{L}_{PS}$, we do not have $C_L(\Sigma, \varphi)$.
4. A *weak revision operator* $\star : \mathcal{L}_{PS} \times \mathcal{L}_{PS} \rightarrow \mathcal{L}_{PS}$: $\Sigma \star \varphi$ represents the new information base obtained once taking account of the observation φ into the information base Σ . For the sake of generality, we are not very demanding about $\star$. The only requirement is that $\Sigma \star \varphi \models_L \varphi$, which expresses that our tests are assumed reliable (each test outcome must be true in the actual world)¹. In the following we will simply refer to those operators as *revision operators* (omitting the *weak*). It would be interesting to explore in more details what happens when one puts more requirements on $\star$ (this is left for further research).

3 Degree of ignorance and degree of contradiction

In this section, we give general definitions of degrees of contradiction and ignorance, i.e., of lack of information, which are to a large extent independent of the logic chosen to represent information. These definitions will be specified further in Sections 4 to 6 where the logic will be fixed.

Definition 3.1 (test contexts) A test context $\mathcal{C}_{\mathcal{L}_{PS}}$ (w.r.t. $\mathcal{L}_{PS}$) is a pair $\langle T, c \rangle$ where T is a finite set of tests and c is a cost function from T to $\mathbf{N}^*$. The outcome to any test $t_\varphi \in T$ is one of $\varphi, \neg\varphi$, where $\varphi \in \mathcal{L}_{PS}$. We say that t_φ is the test about φ , and we assume w.l.o.g. that at most one test t_φ of T is about φ for each $\varphi \in \mathcal{L}_{PS}$. A context is *standard* iff $\forall t_\varphi \in T$, we have $c(t_\varphi) = 1$ (every test has a unit

¹While this condition corresponds to the “success” postulate in the AGM framework [Alchourrón *et al.*, 1985], we do not ask our operators to be AGM ones, especially because the AGM framework is not suited to characterize the revision of an inconsistent formula.

cost). A context is *universal* iff for every $\varphi \in \mathcal{L}_{PS}$, there is a test $t_\varphi \in T$. A context is *atomic* iff the testable formulas are exactly the atomic propositions ($t_x \in T$ iff $x \in PS$).

Definition 3.2 (test plans, trajectories) Given a test context $\mathcal{C}_{\mathcal{L}_{PS}}$, a test plan π is a finite binary tree; each of its non-terminal nodes is labelled with a test action t_φ ; the left and right arcs leaving every non-terminal node labelled with t_φ are respectively labelled with the outcomes φ and $\neg\varphi$. An (outcome) trajectory $\langle o_1, \dots, o_n \rangle$ with respect to π is the sequence of test outcomes on a branch of π . The cost of a trajectory $\langle o_1, \dots, o_n \rangle$ with respect to π is defined as $\sum_{i=1}^n c(t_{\varphi_i})$, where each t_{φ_i} is the test labelling the node of π reached by following the path $\langle o_1, \dots, o_{i-1} \rangle$ from the root of π .

Definition 3.3 (disambiguation, purification) Let π be a test plan and Σ the initial information base.

- The application of π on Σ is the tree $\text{apply}(\pi, \Sigma)$, isomorphic to π , whose nodes are labelled with information bases defined inductively as follows:
 - the root ϵ of $\text{apply}(\pi, \Sigma)$ is labelled with $\Sigma(\epsilon) = \Sigma$;
 - let n be a node of $\text{apply}(\pi, \Sigma)$, labelled with the information base $\Sigma(n)$, whose corresponding node in π is non-terminal and labelled with t_φ ; then n has two children in $\text{apply}(\pi, \Sigma)$, labelled respectively with $\Sigma(n) \star \varphi$ and $\Sigma(n) \star \neg\varphi$.
- π *disambiguates* φ given Σ iff for every terminal node n of $\text{apply}(\pi, \Sigma)$, $\Sigma(n)$ is informative about φ (i.e., either $A_L(\Sigma, \varphi)$ or $A_L(\Sigma, \neg\varphi)$). π (fully) *disambiguates* Σ iff it disambiguates all formulas of $\mathcal{L}_{PS}$, i.e., iff for any terminal node n of $\text{apply}(\pi, \Sigma)$, $\Sigma(n)$ is fully informative.
- π *purifies* φ given Σ iff for every terminal node n of $\text{apply}(\pi, \Sigma)$, $\Sigma(n)$ is not contradictory about φ (i.e., not $C_L(\Sigma(n), \varphi)$). π (fully) *purifies* Σ iff it eliminates all contradictions in Σ , i.e., iff for any terminal node n of $\text{apply}(\pi, \Sigma)$, $\Sigma(n)$ is contradiction-free.

Clearly enough, it can be the case that there is no plan to purify or disambiguate a formula; especially, there is no plan for purifying the constant $\perp$ (false) in any of the three logics considered in the following.

In our framework, degrees of ignorance and contradiction are defined in a uniform way: the degree of ignorance (resp. of contradiction) of Σ measures the minimal effort necessary to disambiguate (resp. to purify) Σ .

Definition 3.4 (degree of contradiction, of ignorance)

Let us define the cost $c(\pi)$ of a test plan π as the maximum of the costs of its trajectories. Then

- The degree of contradiction of Σ is defined by $d_C(\Sigma) = \min(\{c(\pi) \mid \pi \text{ purifies } \Sigma\})$. When no plan purifies Σ , we let $d_C(\Sigma) = +\infty$.
- The degree of ignorance of Σ is defined by $d_I(\Sigma) = \min(\{c(\pi) \mid \pi \text{ disambiguates } \Sigma\})$. When no plan disambiguates Σ , we let $d_I(\Sigma) = +\infty$.

Clearly, both degrees depend on the test context; by default, we consider the standard atomic context.

In the previous definition, we actually define *pessimistic* degrees of contradiction and ignorance (because the cost of

a plan is defined as the *maximum* cost among its trajectories); this principle, consisting in assuming the worst outcome, is known in decision theory as *Wald criterion*. Other criteria could be used instead, such as the optimistic criterion obtained by replacing *max* by *min*. More interesting, the criterion obtained by first using *max* and then *min* for tie-breaking, or the *leximax* criterion, allow for a better discrimination than the pure pessimistic criterion. The choice of a criterion is fairly independent from the other issues discussed in this paper, which gives our framework a good level of flexibility and generality. Due to space limitations, however, we consider only the pessimistic criterion in the rest of the paper.

The definitions below concern the cost of being informative about a formula φ (resp. of getting rid of contradictions about φ) given an initial information base Σ .

Definition 3.5 (disambiguation cost, purification cost)

- The disambiguation cost of φ given Σ , denoted by $C_D(\Sigma, \varphi)$, is the minimum cost of a test plan applied to Σ and disambiguating φ .
- The purification cost of φ given Σ , denoted by $C_P(\Sigma, \varphi)$, is the minimum cost of a test plan applied to Σ and purifying φ .

The disambiguation cost of φ given Σ can be seen as the price one is ready to pay for being informed about φ . The purification cost of φ given Σ is a bit less intuitive: in a database context, for instance, it represents the cost that the database holder is ready to pay to ensure that the database does not contain any contradictory information about a given formula.

Proposition 3.1 *The degrees defined above are related as follows:*

- $d_I(\Sigma) \geq d_C(\Sigma)$;
- $C_D(\Sigma, \varphi) \geq C_P(\Sigma, \varphi)$;
- $C_D(\Sigma, \varphi) = C_D(\Sigma, \neg\varphi)$;
- $d_I(\Sigma) \geq C_D(\Sigma, \varphi)$;
- $C_P(\Sigma, \varphi) = C_P(\Sigma, \neg\varphi)$;
- $d_C(\Sigma) \geq C_P(\Sigma, \varphi)$.

4 Case study 1: classical logic

In this section, we fix $\mathcal{L}_{PS}$ to classical propositional logic CL , which means that we fix the following:

- The language of CL is built up from the connectives $\wedge, \vee, \neg, \rightarrow$ and the constants $\top, \perp$;
- Interpretations are mappings from PS to $\{\top, \text{F}\}$ and the consequence relation is defined as usual by $\Sigma \models_{CL} \varphi$ iff $Mod_{CL}(\Sigma) \subseteq Mod_{CL}(\varphi)$, where $Mod_{CL}(\Sigma)$ denotes the set of classical models of Σ .

Clearly enough, $A_{CL}(\Sigma, \varphi)$ holds iff $\Sigma \models_{CL} \varphi$ and Σ is consistent; and $C_{CL}(\Sigma, \varphi)$ holds iff Σ is inconsistent. Finally, let $\Sigma \star \varphi = \Sigma \wedge \varphi$ (i.e., the revision operator is simple expansion).

Example 4.1 *Figure 1 reports a test plan of minimal cost that fully disambiguates $\Sigma = (a \vee b) \wedge (a \rightarrow c)$ (w.r.t. the standard atomic test context).*

Accordingly, the degree of ignorance of Σ is 2 given the standard atomic context; as Figure 1 also shows, such a degree can be strictly lower than the number of literals built up from PS that are not accepted by Σ (here, the information given by the test outcomes within the test plan are crucial).

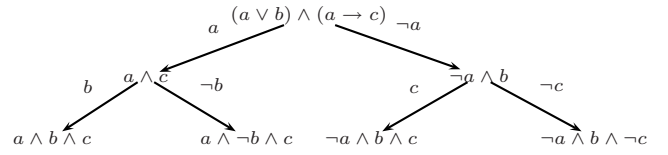


Figure 1: Degree of ignorance of $(a \vee b) \wedge (a \rightarrow c)$

Proposition 4.1 *Given any test context:*

- $C_D(\Sigma, \varphi) = 0$ iff Σ is consistent and $(\Sigma \models \varphi \text{ or } \Sigma \models \neg\varphi)$;
- if Σ is inconsistent, then $C_D(\Sigma, \varphi) = +\infty$;
- $C_D(\Sigma, \varphi \vee \psi) \leq C_D(\Sigma, \varphi) + C_D(\Sigma, \psi)$;
- $C_D(\Sigma, \varphi \wedge \psi) \leq C_D(\Sigma, \varphi) + C_D(\Sigma, \psi)$;
- $d_I(\Sigma) \leq \sum_{x \in PS} C_D(\Sigma, x)$.

Degrees of conflict and purification are not relevant when $\mathcal{L}_{PS}$ is classical logic, just because inconsistency is an extremely rough notion in classical logic. Indeed, we have:

$$d_C(\Sigma) = \begin{cases} 0 & \text{if } \Sigma \text{ is consistent} \\ +\infty & \text{if } \Sigma \text{ is inconsistent} \end{cases}$$

and $C_P(\Sigma, \varphi) = d_C(\Sigma)$ for all φ .

Example 4.2 *Let a, b be two (independent) atomic propositions we focus on. Starting from an information base represented by $\top$ (no contradiction, no information), stepwise expansions progressively lead to a base Σ having a single model over $\{a, b\}$ like $(a = \top, b = \top)$. In this situation, the degrees of ignorance and of contradiction (given the standard atomic context) are minimal (no contradiction, full information). But each additional expansion that does not lead to an equivalent information base leads to an inconsistent one; for instance, the resulting base $\Sigma \star \neg a$ has a maximal degree of contradiction. This is counterintuitive since it seems that $a \wedge b \wedge \neg a$ is both more informative and less contradictory than $a \wedge b \wedge \neg a \wedge \neg b$. This is not reflected by our degrees when classical logic is considered.*

- $d_I(\{\top\}) = 2, d_C(\{\top\}) = 0$
- $d_I(\{a\}) = 1, d_C(\{a\}) = 0$
- $d_I(\{a \wedge b\}) = 0, d_C(\{a \wedge b\}) = 0$
- $d_I(\{a \wedge b \wedge \neg a\}) = +\infty, d_C(\{a \wedge b \wedge \neg a\}) = +\infty$
- $d_I(\{a \wedge b \wedge \neg a \wedge \neg b\}) = +\infty, d_C(\{a \wedge b \wedge \neg a \wedge \neg b\}) = +\infty$

This example also shows that mere expansion is not a very satisfying revision operator. Indeed, since it does not enable to purify any inconsistent base (whatever the test context), expansion does not enable as well to disambiguate any inconsistent base. Furthermore, it may lead to degrees of contradiction (or purification costs) that are not intuitively correct. Thus, on the example, we have $d_I(\{a \wedge b \wedge \neg a\}) = +\infty$, while given the standard atomic context, two tests are sufficient to determine the actual world (over $\{a, b\}$). The reason of this discrepancy between what is expected and what is achieved is that expanding an inconsistent information base always leads to an inconsistent base, while it would be necessary to restore consistency² for achieving purification and disambiguation in classical logic. Note that using AGM revision instead of expansion would not help a lot since AGM operators do not behave well when applied to inconsistent bases.

²A way to do it consists in *forgetting information* [Lang and Marquis, 2002] (possibly everything) in $\{a \wedge b \wedge \neg a\}$.

5 Case study 2: the paraconsistent logic LP_m

Paraconsistent logics have been introduced to avoid *ex falso quodlibet sequitur* of classical logic, hence handling inconsistent bases in a much more satisfying way. While many paraconsistent logics have been defined so far and could be used in our framework, we focus here on the LP_m logic as defined in [Priest, 1991]. This choice is mainly motivated by the fact that this logic is simple enough and has an inference relation that coincides with classical entailment whenever the information base is classically consistent (this feature is not shared by many paraconsistent logics).

- The language of LP_m is built up from the connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$ and the constants $\top$, $\perp$.
- An interpretation ω for LP_m maps each propositional atom to one of the three “truth values” F, B, T , the third truth value B meaning intuitively “both true and false”. 3^{PS} is the set of all interpretations for LP_m . “Truth values” are ordered as follows: $F <_t B <_t T$.
 - $\omega(\top) = T$, $\omega(\perp) = F$
 - $\omega(\neg\alpha) = B$ iff $\omega(\alpha) = B$
 $\omega(\neg\alpha) = T$ iff $\omega(\alpha) = F$
 - $\omega(\alpha \wedge \beta) = \min_{\leq_t}(\omega(\alpha), \omega(\beta))$
 - $\omega(\alpha \vee \beta) = \max_{\leq_t}(\omega(\alpha), \omega(\beta))$
 - $\omega(\alpha \rightarrow \beta) = \begin{cases} \top & \text{if } \omega(\alpha) = F \\ \omega(\beta) & \text{otherwise} \end{cases}$
- The set of models of a formula φ is $Mod_{LP}(\varphi) = \{\omega \in 3^{PS} \mid \omega(\varphi) \in \{T, B\}\}$. Define $\omega! = \{x \in PS \mid \omega(x) = B\}$. Then $\min(Mod_{LP}(\varphi)) = \{\omega \in Mod_{LP}(\varphi) \mid \nexists \omega' \in Mod_{LP}(\varphi) \text{ s.t. } \omega' \subset \omega!\}$. The consequence relation is defined by $\Sigma \models_{LP_m} \varphi$ iff $\min(Mod_{LP}(\Sigma)) \subseteq Mod_{LP}(\varphi)$.
- The definitions of $A_{LP_m}(\Sigma, \varphi)$ and $C_{LP_m}(\Sigma, \varphi)$ are those by default; $C_{LP_m}(\Sigma, \varphi)$ holds only if Σ has no classical model.

Now, what about the revision operator? Actually, the issue of revision in paraconsistent logic has never been considered so far. Expansion is not satisfactory as a revision operator for LP_m because it enables neither the purification task nor the disambiguation one when Σ has no classical model ω (i.e., such that $\omega(x) \neq B$ for each $x \in PS$), whatever the test context. Among the many possible choices, we have considered the following revision operator, defined model-theoretically (for the sake of brevity, we characterize only its restriction to the case the revision formula φ is a literal l).

Let $force(\omega, l)$ be the interpretation of 3^{PS} defined by (for every literal $l = x$ or $\neg x$):

$$\begin{cases} force(\omega, x)(x) = T \\ \forall y \in PS, y \neq x, force(\omega, x)(y) = \omega(y) \\ force(\omega, \neg x)(x) = F \\ \forall y \in PS, y \neq x, force(\omega, \neg x)(y) = \omega(y) \end{cases}$$

Then the revision operator is defined by:

$$Mod_{LP}(\Sigma * l) = \begin{cases} \{\omega \mid \Sigma \mid \omega(l) = T\} & \text{if this set is non-empty,} \\ \text{empty,} & \text{otherwise} \\ \{force(\omega, l) \mid \omega \models \Sigma \text{ and } \omega(l) = B\}. \end{cases}$$

Example 5.1 Stepping back to Example 4.2, we can check that LP_m leads to more intuitive values for the degrees of ignorance and contradiction of the bases we considered. Given the standard atomic test context, we now have:

- $d_I(\{\top\}) = 2$, $d_C(\{\top\}) = 0$

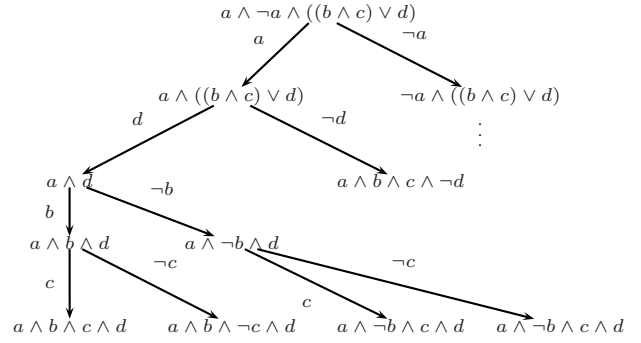


Figure 2: Degrees of ignorance and contradiction in LP_m

- $d_I(\{a\}) = 1$, $d_C(\{a\}) = 0$
- $d_I(\{a \wedge b\}) = 0$, $d_C(\{a \wedge b\}) = 0$
- $d_I(\{a \wedge b \wedge \neg a\}) = 1$, $d_C(\{a \wedge b \wedge \neg a\}) = 1$
- $d_I(\{a \wedge b \wedge \neg a \wedge \neg b\}) = 2$, $d_C(\{a \wedge b \wedge \neg a \wedge \neg b\}) = 2$

Thus, starting from $\top$, expansion (i.e. the logical strengthening of an information base) leads to more information up to a point where full information is reached, then progressively to more contradiction and less information, up to an information base with full contradiction and no information.

Example 5.2 Let us now consider the base $\Sigma = \{a \wedge \neg a \wedge ((b \wedge c) \vee d)\}$. Figure 2 reports a plan of minimal cost (given the standard atomic context) which disambiguates Σ (the subtree rooted below the $\neg a$ outcome is similar to the one of a and is not represented). Here, the degree of ignorance of Σ is 4 and the degree of contradiction of Σ is 1 (after testing a there is no contradiction any more).

Contrariwise to expansion, this revision operator is adequate to our objective:

Proposition 5.1 Every information base that has a model $\in 3^{PS}$ has a finite degree of ignorance and a finite degree of contradiction given any atomic or universal test context.

6 Case study 3: “syntax-based” information bases

Many approaches to reasoning under inconsistency make use of the selection of maximal consistent subsets of a belief base. This principle comes down to [Rescher and Manor, 1970] and is known under different names such as the possible-world approach [Ginsberg and Smith, 1988], assumption-based theories [Poole, 1988], supernormal default theories [Brewka, 1989], syntax-based approaches [Nebel, 1991] etc. We retain the last name and call the logic SBL^3 .

The language of SBL is defined as the set of all pairs $\Sigma = \langle \alpha, \Delta \rangle$, where $\Delta = \{\delta_1, \dots, \delta_n\}$; $\alpha, \delta_1, \dots, \delta_n$ are propositional formulas from the language of CL . α represent the set of hard facts and $\delta_1, \dots, \delta_n$ the defaults. Formulas of the form $\langle \alpha, \emptyset \rangle$ are called *simple formulas* and, slightly abusing notations, are abbreviated into α – thus the language of SBL can be seen as an extension of that of classical logic. A

³Saying that the approach is “syntax-based” actually means that $\{\varphi_1, \dots, \varphi_n\}$ should not be identified with $\varphi_1 \wedge \dots \wedge \varphi_n$, or, in other words, that the comma is viewed as a (non truth-functional) connective.

maximal scenario of Σ is a subset Δ' of Δ such that $\alpha \wedge \Delta'$ is consistent and there is no $\Delta'' \supset \Delta'$ such that $\alpha \wedge \Delta''$ is consistent. We denote by $MS(\Sigma)$ the set of all maximal scenarios of Σ .

The semantics of *SBL* is defined by a preference relation over classical interpretations. Let $\Sigma = \langle \alpha, \Delta \rangle$. If $\omega \in 2^{PS}$, we define $Sat(\omega, \Sigma) = \{i \mid \omega \models_{CL} \delta_i\}$. For $\omega, \omega' \in Mod_{CL}(\alpha)$, we say that $\omega \geq_\Sigma \omega'$ iff $Sat(\omega, \Sigma) \supseteq Sat(\omega', \Sigma)$. Now, $Mod_{SBL}(\Sigma) = \{\omega \mid \omega \models_{CL} \alpha \text{ and there is no } \omega' \models_{CL} \alpha \text{ such that } \omega' >_\Sigma \omega\}$. Consequence in *SBL* is defined by $\Sigma \models_{SBL} \varphi$ iff $Mod_{SBL}(\Sigma) \subseteq Mod_{SBL}(\Sigma') -$ especially, if φ is a simple formula, then $\Sigma \models_{SBL} \varphi$ iff φ is a skeptical consequence of Σ in the sense of default logic.

Let φ be a simple formula. Acceptance is defined by the default formulation. Equivalently, $A_{SBL}(\langle \alpha, \Delta \rangle, \varphi)$ holds iff $\langle \alpha, \Delta \rangle \models_{SBL} \varphi$ and α is consistent. Contradiction refers to maximal scenarios: $C_{SBL}(\Sigma, \varphi)$ iff $MS(\Sigma) = \emptyset$ or there exists $\Delta', \Delta'' \in MS(\Sigma)$ such that $\alpha \wedge \Delta' \models_{CL} \varphi$ and $\alpha \wedge \Delta'' \models_{CL} \neg\varphi$. Lastly, $\langle \alpha, \Delta \rangle \star \varphi$ is defined as $\langle \alpha \wedge \varphi, \Delta \rangle$, i.e., the hard facts are expanded with the revision formula.

Example 6.1 $\Sigma = \langle \emptyset, \{a, \neg a \wedge b, a \rightarrow c, \neg c, c \rightarrow a\} \rangle$. Here is a purification plan of minimal cost for Σ with respect to the standard atomic context. We start by testing a ; $\Sigma \star \neg a = \langle \neg a, \Delta \rangle$ is contradiction-free because it has a single maximal scenario $\{\neg a \wedge b, a \rightarrow c, \neg c, c \rightarrow a\}$; $\Sigma \star a = \langle a, \Delta \rangle$ is not contradiction-free because it has two maximal scenarios $\{a, a \rightarrow c, c \rightarrow a\}$ and $\{a, \neg c, c \rightarrow a\}$; then testing c leads to $\Sigma \star (a \wedge c)$ and $\Sigma \star (a \wedge \neg c)$, both being contradiction-free. Hence, given the standard atomic context, $d_C(\Sigma) = 2$ and $d_I(\Sigma) = 3$ (since disambiguating b will require one additional test).

Proposition 6.1

- Σ is contradiction-free iff $MS(\Sigma)$ is a singleton.
- $\Sigma = \langle \alpha, \Delta \rangle$ is fully informative iff $MS(\Sigma)$ is a singleton $\{\Delta'\}$ and $Mod_{CL}(\alpha \wedge \Delta')$ is a singleton.

Despite it mainly amounts to expansion of the hard facts, if the hard constraint α of Σ is true for sure in the actual world (i.e., it is consistent and consistency cannot be questioned by expansion with test outcomes), then Σ has a finite degree of contradiction and a finite degree of ignorance given any atomic or universal context.

7 Related work

To the best of our knowledge, only few proposals for a notion of degree of information can be found in the literature, and things are even worse to what concerns the notion of degree of contradiction. All existing approaches are stuck to specific propositional logics with the corresponding consequence relations, which address only some aspects of the paraconsistency issue, if any (as evoked previously, there is no undebatable paraconsistent inference relation).

Shannon's information theory [Shannon, 1948] provides the most famous approach on which notions of quantity of information can be defined, but it relies on the assumption that the available information is given under the form of a probability distribution; furthermore, it cannot directly address inconsistent data. Interestingly, our definition of degree

of information is general enough to recover classical entropy, applied to classical logic⁴.

Lozinskii [1994a] gives a set of properties that a measure of quantity of information should satisfy. Our degree of ignorance is fully compatible with Lozinskii's requirements in several cases. The degree of information defined by Lozinskii corresponds to the notion in Shannon's theory, assuming a uniform distribution over the set of propositional interpretations⁵. It is thus required that the input information base Σ has a classical model. [Lozinskii, 1994b] extends [Lozinskii, 1994a] by considering as well some inconsistent logical systems, through a more general notion of model. It is specifically focused on so-called quasi-models of the information set Σ , which are the models of the maximal (w.r.t. $\subseteq$) consistent subsets of Σ . This is sufficient to avoid the notion of degree of information to trivialize when applied to an inconsistent Σ (unless it is a singleton). However, no notion of degree of contradiction is specifically introduced in Lozinskii's approach. He claims that "an inconsistent information base always contains less semantic information than any of its maximal consistent subsets". In our point of view, this may lead to counterintuitive results. For instance, given $PS = \{a, b, c\}$, according to Lozinskii, $\{a\}$ contains more information than $\{a \wedge \neg a \wedge b \wedge c\}$. That is not always the case in our framework: it depends on the context under consideration. Thus, while the degree of contradiction of the former is lower than the degree of contradiction of the latter, the degree of information of the latter is greater than the one of the former (w.r.t. the standard atomic test context).

Wong and Besnard [2001] criticize the syntax-sensitivity of Lozinskii's approach. In particular, the presence of tautologies in Σ may unexpectedly change the quantity of information. As they note this can be easily repaired by considering the models of Σ over the set of variables on which Σ depends [Lang and Marquis, 1998], instead of the set of variables occurring in Σ . Wong and Besnard also adhere to Lozinskii's definition of degree of information; what changes is the underlying notion of quasi-model, since the paraconsistent logic they considered is quasi-classical logic [Besnard and Hunter, 1995; Hunter, 2000].

[Knight, 2003] reports some other postulates for a measure of quantity of information. Our measure d_I does not satisfy all of them, even in simple cases (for space reasons, we cannot detail it here). This contrasts with the two measures introduced by Knight, which generalize in an elegant way Shannon's entropy-based measure to the case the information base is an inconsistent set of formulas. However, both measures trivialize when the information set is an inconsistent singleton.

The only two approaches we are aware of, which consider (non-trivial) degrees of inconsistency defined for clas-

⁴Let pr_Σ be the probability distribution stating that all models of Σ over PS are equiprobable (and the other ones impossible), i.e., for any $\omega \models \Sigma$, $pr_\Sigma(\omega) = \frac{1}{|Mod_{CL}(\Sigma)|}$. The entropy of pr_Σ is defined as $H(pr_\Sigma) = \sum_{\omega \models \Sigma} -pr_\Sigma(\omega) \cdot \log_2 pr_\Sigma(\omega) = \log_2 |Mod_{CL}(\Sigma)|$. Therefore, the integer upper part of $H(pr_\Sigma)$ is nothing but the minimal number of tests that have to be performed to identify the actual world given the standard universal test context.

⁵Such a degree was already known by Kemeny [1953] and Hintikka [1970].

sical formulas (i.e., without additional information, like a preference preorder), are [Knight, 2002] and [Hunter, 2002]. Knight finds a set Σ of classical formulas maximally α -consistent whenever α is the greatest number for which a probability measure P exists, satisfying $P(\varphi) \geq \alpha$ for every $\varphi \in \Sigma$. Accordingly, his measure makes sense when the logic *SBL* is considered, but trivializes whenever Σ is a singleton. Hunter appeals to quasi-classical logic as a framework for dealing with inconsistent information. Considering the minimal quasi-classical models of the information base Σ (roughly, the “most classical” ones), Hunter defines the degree of coherence of Σ as the ratio between the amount of contradiction (conflict) and the amount of information (opinion) of the formula. Such a degree of coherence does not always give what is expected. Consider for instance the two information sets $\Sigma = \{e \wedge \neg e, a \vee (b \wedge c \wedge d)\}$ and $\Sigma' = \{e \wedge \neg e, (a \wedge b) \vee (c \wedge d)\}$. [Hunter, 2002] shows that $\text{Coherence}(\Sigma) > \text{Coherence}(\Sigma')$. We feel it counterintuitive since the two bases shares the same contradiction $e \wedge \neg e$, which is not related to the other formulas of both bases⁶.

Finally, Hunter [2003] defines a degree of *significance* of the contradictions, which makes sense when some information about the importance of potential conflicts is available.

8 Conclusion

The main contribution of the paper is a uniform action-based framework for quantifying both degrees of information and of contradiction. The framework is parameterized by a propositional logic (together with the corresponding notions of consequence, acceptance, contradiction and a revision operator), a test context and an aggregation criterion for computing plan costs. These parameters enable a great flexibility.

There are many interesting notions that can be easily defined in our framework but that we cannot mention here for space reasons. Let us note that through the notion of purification plan, our approach for quantifying contradiction also allows to *localize* conflicts. Note also that notions of *joint degrees* and *conditional degrees* of information / contradiction can be easily defined. Another simple extension would consist in taking advantage of additional knowledge about the sources of information and the origin of conflicts (e.g., in a diagnosis setting, it can be the case that the failure of a component physically causes the failure of other components).

Many other extensions of our approach can be envisioned. For instance, coping with preferences over the goal variables (determining whether a holds is more important than determining whether b holds). Another possible extension concerns the case where ontic actions are available and the objective is to let the actual world as unchanged as possible (i.e., we can execute invasive actions but we prefer not to do it).

Acknowledgements

The third author has been partly supported by the IUT de Lens, the Université d’Artois, the Région Nord / Pas-de-Calais under the TACT-TIC project, and by the European Community FEDER Program.

⁶If we instantiate our framework with LP_m logic (under standard atomic context), we get $d_C(\Sigma) = d_C(\Sigma') = 1$, $d_I(\Sigma) = 5$, and $d_I(\Sigma') = 4$, showing that there is the same amount of contradiction in the two bases and that there is less information in Σ than in Σ' .

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.
- [Besnard and Hunter, 1995] P. Besnard and A. Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In *ECSQARU’95*, pages 44–51, 1995.
- [Brewka, 1989] G. Brewka. Preferred subtheories: an extended logical framework for default reasoning. In *IJCAI’89*, pages 1043–1048, 1989.
- [Ginsberg and Smith, 1988] M. Ginsberg and D. E. Smith. Reasoning about action I: A possible world approach. *Artificial Intelligence*, 35:165–195, 1988.
- [Hintikka, 1970] J. Hintikka. On semantic information. *Information and Inference*, pages 3–27, 1970.
- [Hunter, 2000] A. Hunter. Reasoning with conflicting information using quasi-classical logic. *Journal of Logic and Computation*, 10:677–703, 2000.
- [Hunter, 2002] A. Hunter. Measuring inconsistency in knowledge via quasi-classical models. In *AAAI’02*, pages 68–73, 2002.
- [Hunter, 2003] A. Hunter. Evaluating significance of inconsistencies. In *IJCAI’03 (this issue)*, 2003.
- [Kemeny, 1953] J.G. Kemeny. A logical measure function. *Journal of Symbolic Logic*, 18:289–308, 1953.
- [Knight, 2002] K.M. Knight. Measuring inconsistency. *Journal of Philosophical Logic*, 31:77–98, 2002.
- [Knight, 2003] K.M. Knight. Two information measures for inconsistent sets. *Journal of Logic, Language and Information*, 12:227–248, 2003.
- [Lang and Marquis, 1998] J. Lang and P. Marquis. Complexity results for independence and definability in propositional logic. In *KR’98*, pages 356–367, 1998.
- [Lang and Marquis, 2002] J. Lang and P. Marquis. Resolving inconsistencies by variable forgetting. In *KR’02*, pages 239–250, 2002.
- [Lozinskii, 1994a] E. Lozinskii. Information and evidence in logic systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6:163–193, 1994.
- [Lozinskii, 1994b] E. Lozinskii. Resolving contradictions: a plausible semantics for inconsistent systems. *Journal of Automated Reasoning*, 12:1–31, 1994.
- [Nebel, 1991] B. Nebel. Belief revision and default reasoning: syntax-based approaches. In *KR’91*, pages 417–428, 1991.
- [Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.
- [Priest, 1991] G. Priest. Minimally inconsistent LP. *Studia Logica*, 50:321–331, 1991.
- [Rescher and Manor, 1970] N. Rescher and R. Manor. On inference from inconsistent premises. *Theory and Decision*, 1:179–219, 1970.
- [Shannon, 1948] C.E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.
- [Wong and Besnard, 2001] P. Wong and P. Besnard. Paraconsistent reasoning as an analytic tool. *Journal of the Interest Group in Propositional Logic*, 9:233–246, 2001.

Minimal Change and Maximal Coherence for Epistemic Logic Program Updates

Yan Zhang

School of Computing & IT
University of Western Sydney
Penrith South DC, NSW 1797, Australia
E-mail: yan@cit.uws.edu.au

Abstract

We consider the problem of updating nonmonotonic knowledge bases represented by epistemic logic programs where disjunctive information and notions of knowledge and beliefs can be explicitly expressed. We propose a formulation for epistemic logic program updates based on a principle called minimal change and maximal coherence. The central feature of our approach is that during an update procedure, contradictory information is removed on a basis of minimal change under the semantics of epistemic logic programs and then coherent information is maximally retained in the update result. By using our approach, we can characterize an update result in both semantic and syntactic forms. We show that our approach handles update sequences and satisfies the consistency requirement. We also investigate important semantic properties of our update approach such as reduction, persistence and preservation.

1 Introduction

Logic programming has been proved to be one of the most promising logic based formulations for problem solving, knowledge representation and reasoning, and reasoning about actions and plans. Recent research on logic program updates further shows that logic programming also provides a feasible framework for modeling agents' activities in dynamic environments [Alferes and *et al.*, 2000; Eiter and *et al.*, 2002; Sakama and Inoue, 1999; Zhang and Foo, 1998].

While all current approaches for logic program updates focus on the problem of updating extended logic programs or their variations, updating *epistemic logic programs*, however, has yet to be explored in the research. By combining *knowledge* and *belief* operators into logic rules, epistemic logic programming [Gelfond, 1994] is a powerful representation formalism in logic programming paradigm. It can deal with more difficult problems in reasoning with disjunctive information while traditional disjunctive extended logic programs fail to handle. Furthermore, epistemic logic programs seem more feasible for knowledge reasoning than many other autoepistemic logics [Gelfond, 1994] and has been used as a formal basis for modeling knowledge in action theories, e.g.

[Lobo *et al.*, 2001]. When we use an epistemic logic program to represent an agent's knowledge base, it is a nontrivial question how the agent's knowledge base (an epistemic logic program) can be updated when new information is received.

In this paper, we propose an approach for epistemic logic program updates. Contrary to other logic programs, notions of knowledge and beliefs in epistemic logic programs have strong semantic connections to the standard Kripke structures of modal logics. On the other hand, epistemic logic programs are also sensitive with various syntactic forms. Hence, we believe that a pure model-based or syntax-based approach will not be appropriate to handle epistemic logic program updates. Instead, we require our update formulation to meet three major criteria: (1) an update should be performed on a basis of minimal change semantics to remove contradictory information; (2) based on the minimal change semantics, the update result should have a clear syntactic representation and contain maximal consistent information from previous program(s); and (3) the underlying update procedure should be consistent, that is, updating a consistent program by another consistent program (or a sequence of consistent programs) should generate a consistent result. Our main idea to accomplish these criteria is so called *minimal change and maximal coherence* which presents both semantic and syntactic features in an update procedure.

The paper is organized as follows. Section 2 presents a brief overview on epistemic logic programs. Section 3 develops a formulation for epistemic logic program updates, while section 4 extends this formulation to handle update sequences. Section 5 investigates important semantic properties for our update approach. Finally, section 6 concludes the paper with discussions on related work and future research.

2 Epistemic Logic Programs: An Overview

In this section, we present a general overview on epistemic logic programs. Gelfond extended the syntax and semantics of disjunctive logic programs to allow the correct representation of incomplete information (knowledge) in the presence of multiple extensions. Consider the following disjunctive program about the policy of offering scholarships in some university [Gelfond, 1994]:

$\mathcal{P}$:

$r_1: eligible(x) \leftarrow highGPA(x),$
 $r_2: eligible(x) \leftarrow minority(x), fairGPA(x),$

$$\begin{aligned}
r_3: & \neg eligible(x) \leftarrow \neg fairGPA(x), \\
& \quad \neg highGPA(x), \\
r_4: & interview(x) \leftarrow not\ eligible(x), \\
& \quad not\ \neg eligible(x), \\
r_5: & fairGPA(mike) \text{ or } highGPA(mike) \leftarrow.
\end{aligned}$$

Rule r_4 can be viewed as a formalization of the statement: “the students whose eligibility is not decided by rules r_1 , r_2 and r_3 should be interviewed by the committee”. It is easy to see that $\mathcal{P}$ has two answer sets $\{highGPA(mike), eligible(mike)\}$ and $\{fairGPA(mike), interview(mike)\}$. Therefore the answer to query $interview(mike)$ is *unknown*, which seems too weak from our intuition. Epistemic logic programs will overcome this kind of difficulties in reasoning with incomplete information.

In epistemic logic programs, the language of (disjunctive) extended logic programs is expanded with two modal operators K and M . KF is read as “ F is known to be true” and MF is read as “ F may be believed to be true”. For our purpose, in this paper we will only consider propositional epistemic logic programs where rules containing variables are viewed as the set of all ground rules by replacing these variables with all constants occurring in the language. The semantics for epistemic logic programs is defined by pairs $(\mathcal{A}, W)$, where $\mathcal{A}$ is a collection of sets of ground literals called the sets of *possible beliefs* of certain agent, while W is a set in $\mathcal{A}$ called the agent’s *working set of beliefs*. The truth of a formula F in $(\mathcal{A}, W)$ is denoted by $(\mathcal{A}, W) \models F$ and the falsity by $(\mathcal{A}, W) \models F$, and are defined as follows:

$$\begin{aligned}
(\mathcal{A}, W) \models F & \text{ iff } F \in W \text{ where } F \text{ is a ground atom.} \\
(\mathcal{A}, W) \models KF & \text{ iff } (\mathcal{A}, W_i) \models F \text{ for all } W_i \in \mathcal{A}. \\
(\mathcal{A}, W) \models MF & \text{ iff } (\mathcal{A}, W_i) \models F \text{ for some } W_i \in \mathcal{A}. \\
(\mathcal{A}, W) \models F \wedge G & \text{ iff } (\mathcal{A}, W) \models F \text{ and } (\mathcal{A}, W) \models G. \\
(\mathcal{A}, W) \models F \text{ or } G & \text{ iff } (\mathcal{A}, W) \models \neg(\neg F \wedge \neg G). \\
(\mathcal{A}, W) \models \neg F & \text{ iff } (\mathcal{A}, W) \models F. \\
(\mathcal{A}, W) \models F \text{ iff } \neg F & \text{ in } W \text{ where } F \text{ is a ground atom.} \\
(\mathcal{A}, W) \models KF & \text{ iff } (\mathcal{A}, W) \not\models KF. \\
(\mathcal{A}, W) \models MF & \text{ iff } (\mathcal{A}, W) \not\models MF. \\
(\mathcal{A}, W) \models F \wedge G & \text{ iff } (\mathcal{A}, W) \models F \text{ or } (\mathcal{A}, W) \models G. \\
(\mathcal{A}, W) \models F \text{ or } G & \text{ iff } (\mathcal{A}, W) \models F \text{ and } (\mathcal{A}, W) \models G.
\end{aligned}$$

It is clear that if a formula G is of the form KF , $\neg KF$, MF or $\neg MF$, then its truth value in $(\mathcal{A}, W)$ will not depend on W and we call G a *subjective formula*. On the other hand, if G does not contain K or M , then its truth value in $(\mathcal{A}, W)$ will only depend on W and we call G an *objective formula*. In the case that G is subjective, we simply write $\mathcal{A} \models G$ instead of $(\mathcal{A}, W) \models G$, and $W \models G$ instead of $(\mathcal{A}, W) \models G$ in the case that G is objective.

An *epistemic logic program* is a finite set of rules of the form:

$$F \leftarrow G_1, \dots, G_m, notG_{m+1}, \dots, notG_n. \quad (1)$$

In (1) F is of the form $F_1 \text{ or } \dots \text{ or } F_k$ and $F_1, \dots, F_k$ are objective literals, $G_1, \dots, G_m$ are objective or subjective literals, and $G_{m+1}, \dots, G_n$ are objective literals. For an epistemic logic program $\mathcal{P}$, its semantics is given by its *world view* which is defined in the following steps:

Step 1. Let $\mathcal{P}$ be an epistemic logic program not containing modal operators K and M and negation as failure *not*. A set W of ground literals is called a *belief set* of $\mathcal{P}$ iff W

is a minimal set of satisfying conditions: (i) for each rule $F \leftarrow G_1, \dots, G_m$ from $\mathcal{P}$ such that $W \models G_1 \wedge \dots \wedge G_m$ we have $W \models F$; and (ii) if W contains a pair of complementary literals then $W = Lit$, i.e. W is an inconsistent belief set.

Step 2. Let $\mathcal{P}$ be an epistemic logic program not containing modal operators K and M and W be a set of ground literals in the language of $\mathcal{P}$. By $\mathcal{P}_W$ we denote the result of (i) removing from $\mathcal{P}$ all the rules containing formulas of the form $notG$ such that $W \models G$ and (ii) removing from the rules in $\mathcal{P}$ all other occurrences of formulas of the form $notG$.

Step 3. Finally, let $\mathcal{P}$ be an arbitrary epistemic logic program and $\mathcal{A}$ a collection of sets of ground literals in its language. By $\mathcal{P}_{\mathcal{A}}$ we denote the epistemic logic program obtained from $\mathcal{P}$ by (i) removing from $\mathcal{P}$ all rules containing formulas of the form G such that G is subjective and $\mathcal{A} \not\models G$, and (ii) removing from rules in $\mathcal{P}$ all other occurrences of subjective formulas.

Now we define that a collection $\mathcal{A}$ of sets of ground literals is a *world view* of $\mathcal{P}$ if $\mathcal{A}$ is the collection of all belief sets of $\mathcal{P}_{\mathcal{A}}$. Consider program $\mathcal{P}$ about the eligibility of scholarship discussed earlier, if we replace rule r_4 with the following rule:

$$\begin{aligned}
r'_4: & interview(x) \leftarrow \neg Keligible(x), \\
& \quad \neg K\neg eligible(x),
\end{aligned}$$

then the epistemic logic program that consists of rules r_1, r_2, r_3, r'_4 , and r_5 will have a unique world view $\{\{highGPA(mike), eligible(mike), interview(mike)\}, \{fairGPA(mike), interview(mike)\}\}$, which will result in a yes answer to the query $interview(mike)$.

3 Formalizing Epistemic Logic Program Updates

From this section, we start to develop a formulation for epistemic logic program updates. Consider the update of an epistemic logic program $\mathcal{P}_1$ by another epistemic logic program $\mathcal{P}_2$. Our approach consists of two stages: firstly, we update each world view of $\mathcal{P}_1$ by $\mathcal{P}_2$ - this will remove contradictory information between $\mathcal{P}_1$ and $\mathcal{P}_2$ and ensure a minimal change for the underlying update semantics; and secondly, from the first stage result, we will derive a resulting program which retains the maximal consistent information represented by $\mathcal{P}_1$.

3.1 Preliminaries

To begin with, we first introduce some useful notions. Let $\mathcal{A}$ be a collection of belief sets (i.e. sets of ground literals) and $\mathcal{P}$ an epistemic logic program. We call a pair $(\mathcal{A}, W)$ where $W \in \mathcal{A}$ an *epistemic model induced from $\mathcal{A}$* . If $\mathcal{A}$ is a world view of $\mathcal{P}$, then $\mathcal{M} = (\mathcal{A}, W)$ is also called an *epistemic model* of $\mathcal{P}$. We use $ind(\mathcal{A})$ to denote the set of all epistemic models induced from $\mathcal{A}$. Note $\|ind(\mathcal{A})\| = \|\mathcal{A}\|$.

Consider a rule r of the form (1). We use $H(r)$ and $B(r)$ to denote the head and body parts of rule r respectively. For instance, for rule $r: a \text{ or } \neg b \leftarrow c, Kd, not \neg e$, we have $H(r) = \{a, \neg b\}$, and $B(r) = \{c, Kd, not \neg e\}$. We say $H(r)$ is *satisfied* in an epistemic model $\mathcal{M}$, denoted as $\mathcal{M} \models H(r)$, iff for some $F \in H(r)$ $\mathcal{M} \models F$. We say $B(r)$ is *satisfied* in $\mathcal{M}$, denoted as $\mathcal{M} \models B(r)$, iff (1) for each objective or subjective literal $G \in B(r)$ $\mathcal{M} \models G$, and (2) for each $not G \in B(r)$ $\mathcal{M} \not\models G$. r is *satisfied* in $\mathcal{M}$, denoted

as $\mathcal{M} \models r$, if $\mathcal{M} \models B(r)$ implies $\mathcal{M} \models H(r)$. An epistemic logic program $\mathcal{P}$ is *satisfied* in $\mathcal{M}$ if each rule of $\mathcal{P}$ is satisfied in $\mathcal{M}$. $\mathcal{P}$ is *satisfied* in a collection of belief sets $\mathcal{A}$ if $\mathcal{P}$ is satisfied in all epistemic models induced from $\mathcal{A}$. A ground literal L is *derivable* from $\mathcal{P}$, denoted as $\mathcal{P} \models L$, if L is true in all world views of $\mathcal{P}$, i.e. L is true in all epistemic models induced from each world view of $\mathcal{P}$.

A collection of belief sets is *consistent* if each of its belief sets is consistent. An epistemic logic program is *consistent* if it has a world view and all of its world views are consistent. To simplify our presentation, in the rest of this paper, we will simply call an epistemic logic program *program*. By $\mathbf{V}(\mathcal{P})$ we denote the set of all collections of belief sets in which $\mathcal{P}$ is satisfied. We also denote the set of all world views of $\mathcal{P}$ as $\mathbf{A}(\mathcal{P})$. Clearly $\mathbf{A}(\mathcal{P}) \subseteq \mathbf{V}(\mathcal{P})$.

3.2 Minimal Change on World View Updates

Let W and W_1 be two belief sets. We define $\text{Diff}(W, W_1)$ to be the set $|(W \setminus W_1) \cup (W_1 \setminus W)|^1$. Our method of updating world views shares a similar spirit of traditional model based update [Winslett, 1988] by defining a closeness relation between collections of belief sets, and after the update, the resulting collection of belief sets should be as close as possible to the original one.

Definition 1 (Closeness) Let $\mathcal{A}, \mathcal{A}_1$ and $\mathcal{A}_2$ be three collections of belief sets. We say $\mathcal{A}_1$ is as close to $\mathcal{A}$ as $\mathcal{A}_2$, denoted as $\mathcal{A}_1 \leq_{\mathcal{A}} \mathcal{A}_2$, iff for any $W \in \mathcal{A}$ and $W_2 \in \mathcal{A}_2$, there exists a $W_1 \in \mathcal{A}_1$ such that $\text{Diff}(W, W_1) \subseteq \text{Diff}(W, W_2)$. We denote $\mathcal{A}_1 <_{\mathcal{A}} \mathcal{A}_2$ if $\mathcal{A}_1 \leq_{\mathcal{A}} \mathcal{A}_2$ and $\mathcal{A}_2 \not\leq_{\mathcal{A}} \mathcal{A}_1$.

Proposition 1 $\leq_{\mathcal{A}}$ is a partial ordering.

Definition 2 (Updating world views) Let $\mathcal{A}$ be a world view of some program and $\mathcal{P}$ a program. A collection of belief sets $\mathcal{A}'$ is a possible result of updating $\mathcal{A}$ by $\mathcal{P}$, if

1. $\mathcal{A}' \models \mathcal{P}$ (i.e. $\mathcal{A}' \in \mathbf{V}(\mathcal{P})$); and
2. there does not exist another collection of belief sets $\mathcal{A}''$ satisfying condition 1 and $\mathcal{A}'' <_{\mathcal{A}} \mathcal{A}'$.

We use $\text{Res}(\mathcal{A}, \mathcal{P})$ to denote the set of all possible collections of belief sets after updating $\mathcal{A}$ by $\mathcal{P}$.

Example 1 Consider two epistemic logic programs $\mathcal{P}_1$ and $\mathcal{P}_2$ where

$\mathcal{P}_1$:
 $a \leftarrow$,
 $\neg b \leftarrow \text{not } c$,
 $c \leftarrow \text{not } \neg b$, and
 $\mathcal{P}_2$:
 $b \text{ or } c \leftarrow Ka$,

Clearly, $\mathcal{P}_1$ has one world view $\mathcal{A}_1 = \{\{a, \neg b\}, \{a, c\}\}$. Updating $\mathcal{A}_1$ by $\mathcal{P}_2$, according to Definition 2, we obtain a unique result $\mathcal{A}' = \{\{a, b\}, \{a, c\}\}$. Note that neither $\mathcal{A}'' = \{\{a, c\}\}$ nor $\mathcal{A}^* = \{\{a, \neg b, c\}, \{a, c\}\}$ is a result of updating $\mathcal{A}$ by $\mathcal{P}_2$. Although $\mathcal{A}'' \models \mathcal{P}_2$ and $\mathcal{A}^* \models \mathcal{P}_2$, we have $\mathcal{A}' <_{\mathcal{A}_1} \mathcal{A}''$ and $\mathcal{A}' <_{\mathcal{A}_1} \mathcal{A}^*$ according to Definition 1. ■

¹For a set of ground literals W , $|W| = \{|l| \mid l \in W\}$ and here $|l|$ is l 's corresponding propositional atom, i.e. $|l| = a$ if l is a or $\neg a$.

Proposition 2 Given a collection of belief sets $\mathcal{A}$ and a program $\mathcal{P}$. Then $\text{Res}(\mathcal{A}, \mathcal{P}) = \text{Min}(\mathbf{V}(\mathcal{P}), \leq_{\mathcal{A}})$, where $\text{Min}(\mathbf{V}(\mathcal{P}), \leq_{\mathcal{A}})$ is the subset of $\mathbf{V}(\mathcal{P})$ containing all minimal elements with respect to ordering $\leq_{\mathcal{A}}$.

3.3 Maximal Coherence and Resulting Programs

As discussed earlier, during the second stage of an update procedure, we need to derive a resulting program which should contain the maximal consistent information represented by the initial program in syntactic forms. This is achieved by introducing the concept of *coherence*. Let $\mathcal{M} = (\mathcal{A}, W)$ be an epistemic model and r a rule of the form (1) and $\mathcal{M} \models r$. We define $\mathcal{S}(\mathcal{M}, H(r)) = \{l \mid l \in H(r) \cap W \text{ and } \mathcal{M} \models B(r)\}$. Intuitively, if $\mathcal{M} \models B(r)$, then $\mathcal{S}(\mathcal{M}, H(r))$ presents all literals occurring in $H(r)$ that are true in W . For example, let $\mathcal{M} = (\{\{a, b\}, \{a, c\}, \{a, b, c\}\}, \{a, b, c\})$ and $r: b \text{ or } c \leftarrow Ka$, then $\mathcal{S}(\mathcal{M}, H(r)) = \{b, c\}$. In the case $\mathcal{M} \not\models B(r)$, we have $\mathcal{S}(\mathcal{M}, H(r)) = \emptyset$.

Definition 3 (Subsumption) Given two collections of belief sets $\mathcal{A}_1$ and $\mathcal{A}_2$ and a program $\mathcal{P}$ where $\mathcal{A}_1 \in \mathbf{V}(\mathcal{P})$. $\mathcal{A}_2$ is subsumed by $\mathcal{A}_1$ with respect to $\mathcal{P}$, denoted as $\mathcal{A}_2 \subseteq_{\mathcal{P}} \mathcal{A}_1$, iff $\mathcal{A}_2 \in \mathbf{V}(\mathcal{P})$ and for each $\mathcal{M}_2 \in \text{ind}(\mathcal{A}_2)$ there exists a $\mathcal{M}_1 \in \text{ind}(\mathcal{A}_1)$ such that for each $r \in \mathcal{P}$, $\mathcal{S}(\mathcal{M}_2, H(r)) \subseteq \mathcal{S}(\mathcal{M}_1, H(r))$.

Given $\mathcal{A}_1 \models \mathcal{P}$, if $\mathcal{A}_2$ is subsumed by $\mathcal{A}_1$ with respect to $\mathcal{P}$, then each rule r in $\mathcal{P}$ is also satisfied in $\mathcal{A}_2$ without increasing the satisfied literals of $H(r)$ in epistemic models induced from $\mathcal{A}_2$. For instance, let $\mathcal{P} = \{r: a \text{ or } b \leftarrow Kc\}$, $\mathcal{A}_1 = \{\{a, b, c\}, \{b, c, d\}\}$ and $\mathcal{A}_2 = \{\{a, c\}, \{b, c, e\}\}$, then $\mathcal{A}_2 \subseteq_{\mathcal{P}} \mathcal{A}_1$. Note that belief set $\{a, c\}$ in $\mathcal{A}_2$ only contains one literal of $H(r)$ where belief set $\{a, b, c\}$ in $\mathcal{A}_1$ contains both literals of $H(r)$.

Definition 4 (Coherence) Let $\mathcal{P}$ and $\mathcal{P}'$ be two programs and $\mathcal{A} \in \mathbf{V}(\mathcal{P})$. $\mathcal{P}'$ is coherent with $\mathcal{P}$ with respect to $\mathcal{A}$ if for each $\mathcal{A}' \in \text{Res}(\mathcal{A}, \mathcal{P}')$, $\mathcal{A}'$ is consistent and $\mathcal{A}' \subseteq_{\mathcal{P}} \mathcal{A}$.

The intuitive meaning of coherence is explained as follows. Given a collection of belief sets $\mathcal{A}$ and programs $\mathcal{P}$ and $\mathcal{P}'$ where $\mathcal{A} \models \mathcal{P}$. Updating $\mathcal{A}$ by $\mathcal{P}'$ will change $\mathcal{A}$ to other collections of belief sets that satisfy $\mathcal{P}'$. Coherence ensures that such changes do not violate the satisfaction of $\mathcal{P}$, i.e. for each $\mathcal{A}' \in \text{Res}(\mathcal{A}, \mathcal{P}')$ $\mathcal{A}' \models \mathcal{P}$. Furthermore, it also ensures that the result $\mathcal{A}'$ is not larger than $\mathcal{A}$ in terms of $\mathcal{P}$'s satisfaction. Note that coherence only associates with consistent programs.

Proposition 3 Let $\mathcal{P}$ and $\mathcal{P}'$ be two programs and $\mathcal{A} \in \mathbf{V}(\mathcal{P})$. If $\mathcal{P}'$ is coherent with $\mathcal{P}$ with respect to $\mathcal{A}$, then $\mathcal{P} \cup \mathcal{P}'$ is a consistent program.

Proof: Since $\mathcal{P}'$ is coherent with $\mathcal{P}$ with respect to $\mathcal{A}$, it means that for each $\mathcal{A}' \in \text{Res}(\mathcal{A}, \mathcal{P}')$ $\mathcal{A}'$ is consistent, $\mathcal{A}' \models \mathcal{P}'$ and $\mathcal{A}' \models \mathcal{P}$. It also follows $\mathcal{A}' \in \mathbf{V}(\mathcal{P} \cup \mathcal{P}')$. From Lemma 1 in section 5, we know that $\mathcal{P} \cup \mathcal{P}'$ is a consistent program. ■

Now the resulting program can be specified by the following definition.

Definition 5 (Resulting programs) Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two programs and $\mathcal{A}_1 \in \mathbf{A}(\mathcal{P}_1)$. A program $\mathcal{P}'$ is a possible resulting program after updating $\mathcal{P}_1$ by $\mathcal{P}_2$, iff $\mathcal{P}' = \mathcal{P}_1^* \cup \mathcal{P}_2$, where $\mathcal{P}_1^*$ is a maximal subset of $\mathcal{P}_1$ such that for each $\mathcal{A}' \in \text{Res}(\mathcal{A}_1, \mathcal{P}_2)$ $\mathcal{P}_1^*$ is coherent with $\mathcal{P}_2$ with respect to $\mathcal{A}'$.

Example 2 Example 1 continued. Note that $\mathcal{A}_1 = \{\{a, \neg b\}, \{a, c\}\}$ is the unique world view of $\mathcal{P}_1$. After updating $\mathcal{A}_1$ by $\mathcal{P}_2$, we have $\text{Res}(\mathcal{A}_1, \mathcal{P}_2) = \{\{\{a, b\}, \{a, c\}\}\}$ as shown in Example 1. Then it can be verified that $\{a \leftarrow\}$ is the only maximal subset of $\mathcal{P}_1$ that is coherent with $\mathcal{P}_2$ with respect to $\{\{a, b\}, \{a, c\}\}$. Therefore, from Definition 5, we have

$$\begin{aligned} \mathcal{P}': \\ a \leftarrow, \\ b \text{ or } c \leftarrow Ka, \end{aligned}$$

from which, we conclude that $\mathcal{P}'$ has a unique world view $\{\{a, b\}, \{a, c\}\}$. ■

4 Handling Update Sequences

In this section, we extend our previous formulation to handle update sequences where more than two programs are involved. Let $\mathcal{P}_1, \dots, \mathcal{P}_k$ be consistent programs. $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$ is called an *update sequence*. Informally performing this update sequence means that program $\mathcal{P}_1$ is sequentially updated by $\mathcal{P}_2, \dots$, by $\mathcal{P}_k$. From our intuition, during this process we would like to assign $\mathcal{P}_j$ a higher priority of persistence than $\mathcal{P}_i$ where $j > i$ since $\mathcal{P}_j$ represents the agent's newly received knowledge comparing to $\mathcal{P}_i$. Therefore, after the performance of this update sequence, it is desirable to achieve a result which maximally contains information represented by $\mathcal{P}_k, \mathcal{P}_{k-1}, \dots, \mathcal{P}_1$ with progressively decreasing priorities. Similarly to our previous update formulation, we will formalize this principle from both semantic and syntactic considerations. We first illustrate our idea by the following example.

Example 3 Consider an update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3)$, where

$$\begin{aligned} \mathcal{P}_1: \\ r_1: b \leftarrow Ka, \\ r_2: \neg b \leftarrow Ka, \\ \mathcal{P}_2: \\ r_3: a \leftarrow, \\ \mathcal{P}_3: \\ r_4: \neg a \text{ or } \neg b \leftarrow. \end{aligned}$$

To perform update sequence $\mathbf{P}$, we first update $\mathcal{P}_1$ by $\mathcal{P}_2$, which, according to the approach developed earlier, will generate two resulting programs: $\mathcal{P}' = \{r_1, r_3\}$ and $\mathcal{P}'' = \{r_2, r_3\}$. Now we consider to update $\mathcal{P}'$ and $\mathcal{P}''$ by $\mathcal{P}_3$ respectively. However, this time, we cannot simply apply our previous update approach because both $\mathcal{P}'$ and $\mathcal{P}''$ contain rules from $\mathcal{P}_1$ and $\mathcal{P}_2$ where rules from $\mathcal{P}_2$ should have a higher priority to be maintained during the change. For instance, consider the update of $\mathcal{P}'$ by $\mathcal{P}_3$. $\mathcal{P}'$ has one world view $\mathcal{A}' = \{\{a, b\}\}$. Then updating $\mathcal{A}'$ by $\mathcal{P}_3$ will generate a result $\mathcal{A}'' = \{\{\neg a, b\}, \{a, \neg b\}\}$. Clearly, both $\{r_1\}$ and $\{r_3\}$ are the maximal subsets of $\mathcal{P}'$ that are coherent with $\mathcal{P}_3$ with respect to $\mathcal{A}''$. However, as we discussed above, we should

only have a final resulting program $\{r_3\} \cup \mathcal{P}_3$ because r_3 has a higher priority than r_1 to be persistent during the update. Updating $\mathcal{P}''$ by $\mathcal{P}_3$, on the other hand, will result in a resulting program $\mathcal{P}'' \cup \mathcal{P}_3$ as there is no any contradiction between $\mathcal{P}''$ and $\mathcal{P}_3$. ■

Let $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}'$ a subset of $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$. We denote $\mathcal{P}'[\mathbf{P}, i] = \{r \mid r \in \mathcal{P}_i \cap \mathcal{P}'\}$ ($1 \leq i \leq k$). Note that $\mathcal{P}'[\mathbf{P}, i] \subseteq \mathcal{P}_i$. Now the following two definitions formalize our idea of performing update sequences as discussed above.

Definition 6 (Preferred subsets) Let $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}'$ and $\mathcal{P}''$ two subsets of $\mathcal{P}_1 \cup \dots \cup \mathcal{P}_k$. We say that $\mathcal{P}'$ is more preferred than $\mathcal{P}''$ with respect to $\mathbf{P}$, denoted as $\mathcal{P}' \ll_{\mathbf{P}} \mathcal{P}''$, iff (i) $\mathcal{P}'' \subset \mathcal{P}'$; or (ii) there exist some i and j where $1 \leq i < j \leq k$ such that for all l ($j \leq l \leq k$), $\mathcal{P}''[\mathbf{P}, l] \subseteq \mathcal{P}'[\mathbf{P}, l]$, for some l' ($j \leq l' \leq k$) $\mathcal{P}''[\mathbf{P}, l'] \subset \mathcal{P}'[\mathbf{P}, l']$, and $\mathcal{P}'[\mathbf{P}, i] \subset \mathcal{P}''[\mathbf{P}, i]$.

Definition 7 (Update selection functions) Let $\mathbb{P}$ be the collection of all epistemic logic programs. For each $i \geq 1$, we define Π_i to be a i -ary update selection function if Π_i is a mapping:

$$\Pi_i : \underbrace{\mathbb{P} \times \dots \times \mathbb{P}}_i \longrightarrow 2^{\mathbb{P}}$$

where the following conditions hold:

1. if $i = 1$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1)$, $\Pi_1(\mathbf{P}) = \{\mathcal{P}_1\}$;
2. if $i = 2$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2)$, $\Pi_2(\mathbf{P}) = \{\mathcal{P}' \mid \text{where } \mathcal{P}' \text{ is a resulting program as described in Definition 5}\}$;
3. if $i = k$ and $k > 2$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$, $\mathcal{P} \in \Pi_k(\mathbf{P})$ iff $\mathcal{P} = \mathcal{P}^* \cup \mathcal{P}_k$, where $\mathcal{P}^*$ is obtained as follows:
 - (a) let $\mathbf{P}_1 = (\mathcal{P}_1, \dots, \mathcal{P}_{k-1})$ and $\mathcal{P}' \in \Pi_{k-1}(\mathbf{P}_1)$;
 - (b) let $\mathcal{A}$ be a world view of $\mathcal{P}'$, and $\mathcal{P}^*$ is a maximal subset of $\mathcal{P}'$ such that for all $\mathcal{A}' \in \text{Res}(\mathcal{A}, \mathcal{P}_k)$ $\mathcal{P}^*$ is coherent with $\mathcal{P}_k$ with respect to $\mathcal{A}'$;
 - (c) there does not exist another maximal subset $\mathcal{P}^\dagger$ of $\mathcal{P}'$ that satisfies condition (b) and $\mathcal{P}^\dagger \ll_{\mathbf{P}_1} \mathcal{P}^*$.

Example 4 Consider an irrigation system which has the following general rules to decide whether the plants should be watered:

If there is no evidence showing that it will not be raining next day, then we do not need to water the plants; and
If there is no evidence showing that the soil is dry, then we do not need to water the plants.

It is also assumed that the soil is dry or it will not be raining next day. This scenario can be represented by the following program $\mathcal{P}_1$:

²Note that here we refer to the same world view $\mathcal{A}$ of $\mathcal{P}'$ as in (b) when we say $\mathcal{P}^\dagger$ is coherent with $\mathcal{P}_k$ with respect to all $\mathcal{A}'$ in $\text{Res}(\mathcal{A}, \mathcal{P}_k)$.

$\mathcal{P}_1$:
 $r_1: \neg \text{watering} \leftarrow \text{not } \neg \text{to_be_raining},$
 $r_2: \neg \text{watering} \leftarrow \text{not } \text{dry},$
 $r_3: \text{dry or } \neg \text{to_be_raining} \leftarrow.$

$\mathcal{P}_1$ has one world view:

$\{\{\text{dry}, \neg \text{watering}\}, \{\neg \text{to_be_raining}, \neg \text{watering}\}\},$

from which it is concluded that we do not need to water the plants. However, from a conservative viewpoint for plants' growth, this result is rather optimistic because r_3 does not represent an exclusive disjunctive information. Therefore, we consider to update $\mathcal{P}_1$ by $\mathcal{P}_2$:

$\mathcal{P}_2$:
 $r_4: \text{watering} \leftarrow \neg K \neg \text{dry}, M \neg \text{to_be_raining},$
 $r_5: \neg \text{watering} \leftarrow \neg \text{dry},$
 $r_6: \neg \text{watering} \leftarrow \text{to_be_raining},$

which says that if it is not known that the soil is dry and it may be believed that it will not be raining next day, then we water the plants; and we do not need to water the plants if the soil is not dry or it will be raining next day. After a period of time, suppose new information is further received that is represented by $\mathcal{P}_3$ as follows:

$\mathcal{P}_3$:
 $r_7: \neg \text{dry or } \text{to_be_raining} \leftarrow.$

Now we consider the update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3)$. Let $\mathbf{P}_1 = (\mathcal{P}_1, \mathcal{P}_2)$. Then according to Definition 7, We have $\Pi_2(\mathbf{P}_1) = \{\mathcal{P}'\}$, where $\mathcal{P}' = \{r_3, r_4, r_5, r_6\}$ (note that rules r_1 and r_2 are removed from the initial program $\mathcal{P}_1$). $\mathcal{P}'$ has one world view $\{\{\text{dry}, \text{watering}\}, \{\neg \text{to_be_raining}, \text{watering}\}\}$, from which it is concluded that we need to water the plants. Finally, we have a unique resulting program $\Pi_3(\mathbf{P}) = \{\mathcal{P}''\}$ where $\mathcal{P}'' = \{r_4, r_5, r_6, r_7\}$, from which it is concluded that we no longer need to water the plants, i.e. $\mathcal{P}'' \models \neg \text{watering}$. ■

5 Semantic Characterizations

In this section, we study important semantic properties of our approach for epistemic logic program updates. Let $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}$ a program, by $(\mathbf{P}, \mathcal{P})$ we denote the update sequence $(\mathcal{P}_1, \dots, \mathcal{P}_k, \mathcal{P})$. By $\text{Body}(\mathcal{P})$ and $\text{Head}(\mathcal{P})$ we denote the sets of all objective literals occurring in the bodies and heads of rules in $\mathcal{P}$ respectively³. To simplify our presentation, we also use notion $\pi_k(\mathbf{P})$ to denote an arbitrary resulting program in $\Pi_k(\mathbf{P})$ whenever there is no confusion in the context. For instance, by $\pi_k(\mathbf{P}) \models L$, we mean that for any $\mathcal{P} \in \Pi_k(\mathbf{P})$, $\mathcal{P} \models L$; and by $\pi_k(\mathbf{P}) = \pi_{k+1}(\mathbf{P}, \mathcal{P})$, we mean that any resulting program in $\Pi_k(\mathbf{P})$ is also a resulting program in $\Pi_{k+1}(\mathbf{P}, \mathcal{P})$, and vice versa, etc..

Lemma 1 *A program $\mathcal{P}$ is consistent iff $\mathbf{V}(\mathcal{P}) \neq \emptyset$ and there is a consistent collection of belief sets in $\mathbf{V}(\mathcal{P})$.*

Theorem 1 *Let $\mathbf{P}$ be an update sequence with a length of k and $\mathcal{P}$ a program. Then the following properties hold:*

³Note that $\text{Body}(\mathcal{P})$ is different from $B(r)$ while $\text{Head}(\mathcal{P}) = \cup_{r \in \mathcal{P}} H(r)$. See section 3.1 for definitions on $B(r)$ and $H(r)$.

1. $\pi_k(\mathbf{P})$ is a consistent program (**consistency property**);
2. $\mathbf{V}(\pi_{k+1}(\mathbf{P}, \mathcal{P})) \subseteq \mathbf{V}(\mathcal{P})$;
3. if $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$, then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \pi_k(\mathbf{P}) \cup \mathcal{P}$.

The following theorem provides a sufficient condition under which the computation of a $(k + 1)$ -length update sequence can be reduced to the computation on a k -length update sequence.

Theorem 2 (Reduction property) *Let $\mathbf{P}$ be an update sequence with a length of k , and $\mathcal{P}$ and $\mathcal{P}'$ two programs. If $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P}) \subseteq \mathbf{V}(\mathcal{P}')$, then $\pi_{k+2}(\mathbf{P}, \mathcal{P}, \mathcal{P}') = \pi_{k+1}(\mathbf{P}, \mathcal{P} \cup \mathcal{P}')$.*

Now we investigate two specific properties called *persistence* and *preservation* for epistemic logic program updates. Informally, the persistence property ensures that if a literal is derivable from a program, then after updating this program, this literal is still derivable from the resulting program. The preservation property, on the other hand, says that if a literal is derivable from a program, then updating other program by this program will still preserve this literal's derivability from the resulting program. While the persistence property is usually not valid for classical belief revision and update due to their nonmonotonicity, the preservation property, nevertheless, indeed holds for classical belief revision and update. It is not difficult to observe that generally none of these two properties holds for extended logic program updates or epistemic logic program updates. However, it is always worthwhile to explore their restricted forms because under certain conditions, these properties may significantly simplify the computation of a query to the update result. We first present the following lemma.

Lemma 2 *Let $\mathcal{P}$ be a program and L a ground literal. Suppose $\mathcal{P}'$ is a subset of $\mathcal{P}$ such that $\mathcal{P}' \models L$ and $\text{Body}(\mathcal{P}') \cap \text{Head}(\mathcal{P} \setminus \mathcal{P}') = \emptyset$. Then $\mathcal{P} \models L$.*

Proof: (Proof Sketch) The proof for this lemma is rather technical and involves the proof of a result so called *splitting theorem* for epistemic logic programs which inherits a similar feature of the splitting theorem for extended logic programs [Lifschitz and Turner, 1994]. Basically, the splitting theorem for epistemic logic programs states that if $\text{Body}(\mathcal{P}') \cap \text{Head}(\mathcal{P} \setminus \mathcal{P}') = \emptyset$ (here $\mathcal{P}$ is a consistent program), then for any world view $\mathcal{A}$ of $\mathcal{P}$ and every belief set W in $\mathcal{A}$, there exists a belief set W' that is in some world view of $\mathcal{P}'$ such that $W' \subseteq W$. Due to a space limit, here we have to omit the formal proof of the splitting theorem of epistemic logic programs [Zhang, 2003]⁴. Based on the splitting theorem, this lemma is proved as follows. Consider program $\mathcal{P}'$. Since $\mathcal{P}' \models L$, for each world view $\mathcal{A}'$ of $\mathcal{P}'$, $\mathcal{A}' \models L$. That is, for each $W' \in \mathcal{A}'$, $L \in W'$. Suppose $\mathcal{P} \not\models L$. Then there exists a world view $\mathcal{A}$ of $\mathcal{P}$ such that $\mathcal{A} \not\models L$. Then there must be some belief set $W \in \mathcal{A}$ such that $L \notin W$. However, since $\text{Body}(\mathcal{P}') \cap \text{Head}(\mathcal{P} \setminus \mathcal{P}') = \emptyset$,

⁴One referee pointed that as a general extension of Lifschitz and Turner's result, Watson also proposed a splitting set theorem for epistemic logic programs [Watson, 2000]. It appears that our splitting theorem has a different feature from Watson's though the later may be also used for this proof.

from the splitting theorem, there exists a world view $\mathcal{A}^*$ of $\mathcal{P}'$ such that for some $W^* \in \mathcal{A}^*$ we have $W^* \subseteq W$. Since $\mathcal{A}^* \models L$, it follows $L \in W^*$ and then $L \in W$. Obviously this contradicts the fact $L \notin W$. So we have $\mathcal{P} \models L$. ■

From Lemma 2, we can prove the following two properties.

Theorem 3 (Persistence property) *Let $\mathbf{P}$ be an update sequence with a length of k , $\mathcal{P}$ a program, and L a ground literal. Suppose $\pi_k(\mathbf{P}) \models L$. Then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$ if $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$ and $\text{Body}(\pi_k(\mathbf{P})) \cap \text{Head}(\mathcal{P}) = \emptyset$.*

Proof: Since $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$, from Theorem 1, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \pi_k(\mathbf{P}) \cup \mathcal{P}$. On the other hand, from condition $\text{Body}(\pi_k(\mathbf{P})) \cap \text{Head}(\mathcal{P}) = \emptyset$ and Lemma 2, it follows $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$. ■

Theorem 4 (Preservation property) *Let $\mathbf{P} = (\mathcal{P}_1, \dots, \mathcal{P}_k)$ be an update sequence with a length of k , $\mathcal{P}$ a program, and L a ground literal. Suppose $\mathcal{P} \models L$. Then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$ if $\text{Head}(\bigcup_{i=1}^k \mathcal{P}_i) \cap \text{Body}(\mathcal{P}) = \emptyset$.*

Proof: Clearly $\mathcal{P} \subseteq \pi_{k+1}(\mathbf{P}, \mathcal{P})$. On the other hand, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \mathcal{P}^* \cup \mathcal{P}$, where $\mathcal{P}^* \subseteq \bigcup_{i=1}^k \mathcal{P}_i$ and satisfies conditions (b) and (c) in Definition 7. So we have $\text{Head}(\mathcal{P}^*) \cap \text{Body}(\mathcal{P}) = \emptyset$. Then directly from Lemma 2, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$. ■

6 Concluding Remarks

In this paper, we proposed a formulation for epistemic logic program updates. Our update approach was developed based on the principle of minimal change and maximal coherence. By using our approach, not only a minimal change semantics is embedded into the underlying update procedure, but also a maximal syntactic coherence is achieved after the update. We also investigated important semantic properties of our update approach. This work can be viewed as a further development on knowledge update [Baral and Zhang, 2001]. Although all current approaches of logic program updates have their own features, it is not clear yet whether they are suitable to handle epistemic logic program updates. For instance, in [Zhang, 2003] we demonstrated that a straightforward extension of Alferes *et al.*'s approach [Alferes and *et al.*, 2000] or Eiter *et al.*'s approach [Eiter and *et al.*, 2002] to epistemic logic program updates may generate incorrect solutions, while the proposed generic framework in [Eiter and *et al.*, 2001] seems not applicable to our case either. On the other hand, a syntax-based approach, e.g. [Sakama and Inoue, 1999], cannot characterize the semantics of epistemic logic program updates, and some approaches, e.g. [Eiter and *et al.*, 2002], do not obey the consistency requirement that is believed to be one of the essential requirements for any revision and update systems [Katsuno and Mendelzon, 1991].

Several issues remain open for our future research. First, in our update approach, we did not consider the issue of preference over different resulting programs. In practice, it is possible that one resulting program is more preferred than the other in terms of the domain semantics. This problem involves conflict resolution which is a difficult issue in logic program updates [Zhang and Foo, 1998]. Second, as world view se-

mantics can be viewed as a generalization of answer set semantics, our update approach is also applicable for extended logic program updates. Therefore, it is important to characterize similarities and differences between our approach and other logic program update approaches from a semantic viewpoint. Finally, since epistemic logic programs have been used as a main component in knowledge based action theories, e.g. [Lobo *et al.*, 2001], we expect that these theories may be significantly enhanced for representing interactions between actions and knowledge by applying our update approach.

References

- [Alferes and *et al.*, 2000] J. Alferes and *et al.* Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-2):43–70, 2000.
- [Baral and Zhang, 2001] C. Baral and Y. Zhang. On the semantics of knowledge update. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 97–102. Morgan Kaufmann Publishers, Inc., 2001.
- [Eiter and *et al.*, 2001] T. Eiter and *et al.* A framework for declarative update specification in logic programs. In *Proceedings of IJCAI-01*, pages 649–654, 2001.
- [Eiter and *et al.*, 2002] T. Eiter and *et al.* On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, 2:711–767, 2002.
- [Gelfond, 1994] M. Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12:98–116, 1994.
- [Katsuno and Mendelzon, 1991] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, pages 387–394, 1991.
- [Lifschitz and Turner, 1994] V. Lifschitz and H. Turner. Splitting a logic program. In *Proceedings of Eleventh International Conference on Logic Programming*, pages 23–37. MIT Press, 1994.
- [Lobo *et al.*, 2001] J. Lobo, G. Mendez, and S.R. Taylor. Knowledge and the action description language $\neg$. *Theory and Practice of Logic Programming*, 1:129–184, 2001.
- [Sakama and Inoue, 1999] C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Proceedings of LPNMR'99*, pages 147–161. LNAI, Vol 1730, Springer, 1999.
- [Watson, 2000] R. Watson. A splitting set theorem for epistemic specifications. In *Proceedings of NMR2000*, 2000.
- [Winslett, 1988] M. Winslett. Reasoning about action using a possible models approach. In *Proceedings of AAAI-88*, pages 89–93, 1988.
- [Zhang and Foo, 1998] Y. Zhang and N.Y. Foo. Updating logic programs. In *Proceedings of ECAI-98*, pages 403–407. John Wiley & Sons, Inc., 1998.
- [Zhang, 2003] Y. Zhang. Minimal change and maximal coherence for epistemic logic program updates. In *Manuscript*, 2003.

CASE-BASED REASONING

Increasing Dialogue Efficiency in Case-Based Reasoning Without Loss of Solution Quality

David McSherry

School of Computing and Information Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

Abstract

Increasing dialogue efficiency in case-based reasoning (CBR) must be balanced against the risk of commitment to a sub-optimal solution. Focusing on incremental query elicitation in recommender systems, we examine the limitations of naive strategies such as terminating the dialogue when the similarity of any case reaches a predefined threshold. We also identify necessary and sufficient conditions for recommendation dialogues to be terminated without loss of solution quality. Finally, we evaluate a number of attribute-selection strategies in terms of dialogue efficiency given the requirement that there must be no loss of solution quality.

1 Introduction

In *conversational* case-based reasoning (CCBR), a query is incrementally elicited in an interactive dialogue with the user [Aha *et al.*, 2001]. Attribute-selection strategies that aim to minimize the length of such dialogues have recently attracted much research interest [Doyle and Cunningham, 2000; Kohlmaier *et al.*, 2001; McSherry, 2001; Schmitt *et al.*, 2002]. Potential benefits include avoiding frustration for the user, reducing network traffic in e-commerce domains, and simplifying explanations of how conclusions were reached [Breslow and Aha, 1997; Doyle and Cunningham, 2000]. While our focus in this paper is on incremental query elicitation in recommender systems, it is worth noting that the ability to solve problems by asking a small number of questions has been a major factor in the success of help-desk applications of CBR [Aha *et al.*, 2001; Watson, 1997].

An advantage of information gain [Quinlan, 1986] as a basis for attribute selection is that it tends to produce small decision trees, thus helping to reduce the length of problem-solving dialogues [Doyle and Cunningham, 2000; McSherry, 2001]. However, concerns about its suitability in e-commerce domains include the fact that no use is made of the system's similarity knowledge [Kohlmaier *et al.*, 2001;

Schmitt *et al.*, 2002]. Any importance weights associated with the case attributes are also ignored. As a result, it is possible for a case to be recommended simply because no other case has the preferred value for an attribute of low importance, even if its values for other attributes are unacceptable to the user.

Kohlmaier *et al.* [2001] propose a *similarity-based* approach to attribute selection in which the best attribute is the one that maximizes the expected variance of the similarities of candidate cases. In domains in which cases are indexed by different features, another alternative to information gain is to rank questions in decreasing order of their frequency in the most similar cases [Aha *et al.*, 2001].

To address the trade-off between dialogue efficiency and solution quality, a CBR system must also be capable of recognizing when the dialogue can be terminated while minimizing the risk of commitment to a sub-optimal solution. Existing approaches include terminating the dialogue when the similarity of any case reaches a predefined threshold, or the achievable information gain is less than a predefined level, or the set of candidate cases has been reduced to a manageable size [Aha *et al.*, 2001; Doyle and Cunningham, 2000; Kohlmaier *et al.*, 2001].

However, a limitation of these approaches is that there is no guarantee that a better solution would not be found if the dialogue were allowed to continue. Whether it is possible to identify more reliable criteria for termination of problem-solving dialogues is an issue that has received little attention, if any, in CBR research.

In Section 2, we examine the trade-off between dialogue efficiency and solution quality in recommender systems. In Section 3, we present empirical techniques for identifying cases that can never emerge as the “best” case and can thus be eliminated. In Section 4, we identify necessary and sufficient conditions for the dialogue to be terminated without loss of solution quality. In Section 5, we evaluate a number of attribute-selection strategies in terms of dialogue efficiency given the requirement that there must be no loss of solution quality. Our conclusions are presented in Section 6.

2 CCBR in Product Recommendation

A generic algorithm for CCBR in product recommendation is shown in Figure 1. In this context, the elicited query, or partial query, represents the preferences of the user with respect to the attributes of the available products. At each stage of the recommendation dialogue, the system selects the next most useful attribute, asks the user for the preferred value of this attribute, and retrieves the case (or product) that is most similar to the query thus far elicited. The dialogue continues until the *termination criteria* are satisfied, or until no further attributes remain. At this point, the case that is most similar to the current query is presented to the user as the recommended case.

```

algorithm CCBR
begin
   $Q \leftarrow \emptyset$ 
  repeat
    select the next most useful attribute  $a$ 
    ask the user for the preferred value  $v$  of  $a$ 
     $Q \leftarrow Q \cup \{a = v\}$ 
    retrieve the case  $C$  that is most similar to  $Q$ 
  until termination criteria are satisfied
  recommend  $C$ 
end

```

Figure 1. CCBR in product recommendation.

2.1 Similarity Measures

The similarity of a given case C to a query Q over a set of case attributes A is typically defined as:

$$Sim(C, Q) = \sum_{a \in A} w_a sim_a(C, Q)$$

where for each $a \in A$, w_a is a numeric weight representing the importance of a , and $sim_a(C, Q)$ is a measure of the similarity of $\pi_a(C)$, the value of a in C , to $\pi_a(Q)$, the preferred value of a . As usual in practice, we assume that for all $a \in A$, $0 \leq sim_a(x, y) \leq 1$ and $sim_a(x, y) = 1$ if and only if $x = y$. Often in practice, $Sim(C, Q)$ is divided by $\sum_{a \in A} w_a$ to

give an overall similarity score between 0 and 1, though this has no effect on the ranking of the retrieved cases. We will say that Sim is a *regular* similarity measure if for each $a \in A$, the distance measure $d_a(x, y) = 1 - sim_a(x, y)$ satisfies the triangle inequality: $d_a(x, z) \leq d_a(x, y) + d_a(y, z)$ for all x, y, z .

At each stage of the CCBR process, retrieval is based on an *incomplete* query Q in which preferred values are specified for a subset A_Q of the case attributes A . The similarity of a given case C to an incomplete query Q is:

$$Sim(C, Q) = \sum_{a \in A_Q} w_a sim_a(C, Q)$$

In practice, the full-length query Q^* that represents the preferences of the user with respect to all the case attributes may never be known, and is only one of the many possible *completions* of Q in the sense of the following definition.

Definition 1 Given an incomplete query Q , we say that a full-length query Q^+ is a completion of Q if $\pi_a(Q^+) = \pi_a(Q)$ for all $a \in A_Q$.

Proposition 1 If Q is an incomplete query, then for any case C and completion Q^+ of Q ,

$$Sim(C, Q^+) = Sim(C, Q) + \sum_{a \in A - A_Q} w_a sim_a(C, Q^+)$$

2.2 Measures of Retrieval Performance

Often in recommender systems, cases other than those that are maximally similar to a target query are presented as alternatives that the user may wish to consider [e.g. McGinty and Smyth, 2002]. However, in measuring retrieval performance, we assume that the *retrieval set* for a given query Q is the set of cases C for which $Sim(C, Q)$ is maximal; that is, no case is more similar to Q .

Definition 2 Given a query Q , we define the retrieval set for Q to be:

$$rs(Q) = \{C: Sim(C, Q) \geq Sim(C^\circ, Q) \text{ for all } C^\circ\}$$

Often in practice, $rs(Q)$ contains a single case; if not, we assume that the user is shown all cases that are maximally similar to her final query. A simple measure of dialogue efficiency is the number of questions, on average, that the user is asked before a recommendation is made. We measure precision and recall for an incomplete query Q relative to the full-length query Q^* that represents the preferences of the user with respect to all the case attributes.

Definition 3 Given an incomplete query Q , we define:

$$precision(Q) = \frac{|rs(Q) \cap rs(Q^*)|}{|rs(Q)|} \times 100$$

$$recall(Q) = \frac{|rs(Q) \cap rs(Q^*)|}{|rs(Q^*)|} \times 100$$

2.3 Similarity Thresholds

We now use an example recommender system in the PC domain to illustrate the trade-off between dialogue efficiency and solution quality in CCBR with termination based on similarity thresholds. The case library contains the descriptions of 120 personal computers [McGinty and Smyth, 2002]. The attributes in the case library and weights assigned to them in our experiments are type (8), price (7), manufacturer (6), processor (5), speed (4), monitor size (3), memory (2), and hard disk capacity (1). In this initial experiment, attributes are selected in decreasing order of their importance weights and the dialogue is terminated

when the similarity of any case reaches a predefined threshold.

We use a *leave-one-out* approach in which each case is temporarily removed from the case library and used to represent the preferences of the user in a simulated recommendation dialogue. We measure dialogue length as the percentage of the 8 possible questions the user is asked before the dialogue is terminated. Average dialogue length, precision and recall over all simulated dialogues are shown in Figure 2 for similarity thresholds in the range from 0.4 to 1. In this case library, there is never more than a single case in the retrieval set for the full-length query that provides the baseline for our evaluation of retrieval performance. It follows that for each threshold, recall is the percentage of dialogues in which this “best” case is recommended.

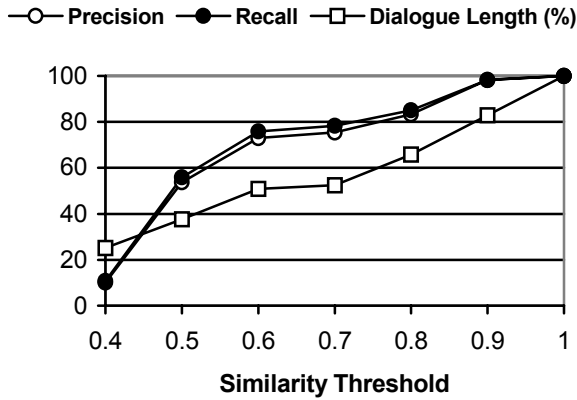


Figure 2. Trade-off between dialogue efficiency and solution quality with termination based on a predefined threshold.

The similarity threshold of 0.7 can be seen to have reduced average dialogue length by almost 50%. This equates to about 4 out of 8 questions, on average, being asked before a recommendation is made. However, the trade-off is a reduction of more than 20% in both precision and recall. This means that the best case is recommended in less than 80% of dialogues. As precision is less than recall for the 0.7 threshold, there are also occasions when the system recommends the best case along with one or more other cases that are equally similar to the final query. It is also worth noting that even a threshold of 0.9, though providing a reduction in dialogue length of 17%, does not ensure that the best case is always recommended.

2.4 When Can the Dialogue be Safely Terminated?

The potentially damaging effects of similarity thresholds on solution quality highlight the need for more reliable criteria for terminating CCBR dialogues. An incomplete query Q gives perfect precision and recall if and only if $rs(Q) = rs(Q^*)$, but the problem is that Q^* is unknown. One can imagine an approach to CCBR that relies on exhaustive search to determine when the dialogue can be safely terminated; that is, in the certain knowledge that there can

be no loss of precision or recall. The dialogue would be terminated only if all possible completions Q^+ of the current query Q yielded the same retrieval set as Q . However, this approach is unfeasible in practice as the number of possible completions of a given query is often very large.

In Section 4, we identify criteria for safely terminating the dialogue that require minimal computational effort in comparison with exhaustive search. The approach is based on the concept of *case dominance* that we now introduce.

Definition 4 We say that a given case C_2 is dominated by another case C_1 with respect to an incomplete query Q if $Sim(C_2, Q) < Sim(C_1, Q)$ and $Sim(C_2, Q^+) < Sim(C_1, Q^+)$ for all possible completions Q^+ of Q .

The importance of case dominance can easily be seen. Any case that is dominated with respect to the current query can be eliminated as it can never emerge as the best case *regardless* of the preferences of the user with respect to the remaining attributes. In the following section, we present empirical techniques for identifying dominated cases that can easily be applied in practice.

3 Identifying Dominated Cases

We now present 3 alternative criteria for identifying cases that are dominated with respect to an incomplete query. We will refer to the dominance criteria identified in Theorems 1, 2 and 3 as DC1, DC2 and DC3 respectively. DC1 and DC2 are sufficient but not necessary conditions for a given case to be dominated by another case, while DC3 is both a necessary and a sufficient condition. DC1 and DC2 have the advantage of not relying on the triangle inequality, but only DC3 is guaranteed to detect all dominance relationships.

Theorem 1 A given case C_2 is dominated by another case C_1 with respect to an incomplete query Q if:

$$Sim(C_2, Q) + \sum_{a \in A - A_Q} w_a < Sim(C_1, Q)$$

Proof Immediate from Theorem 2. $\square$

A limitation of DC1 is that it fails to recognize that the similarity of the more similar case C_1 is a *moving target* for the less similar case C_2 , and that the latter may be dominated even if it can equal or exceed the current similarity of C_1 . For example, if C_1 and C_2 have the same values for one of the remaining attributes, then any increase in similarity gained by C_2 with respect to this attribute is also gained by C_1 . Our second dominance criterion, DC2, ignores attributes for which C_1 and C_2 have the same values, thus making it less susceptible to this problem.

Theorem 2 A given case C_2 is dominated by another case C_1 with respect to an incomplete query Q if:

$$Sim(C_2, Q) + \sum_{a \in A_1} w_a < Sim(C_1, Q)$$

where $A_1 = \{a \in A - A_Q : \pi_a(C_1) \neq \pi_a(C_2)\}$.

Proof If $A_2 = \{a \in A - A_Q : \pi_a(C_1) = \pi_a(C_2)\}$, then for any completion Q^+ of Q :

$$\begin{aligned} \text{Sim}(C_2, Q^+) &= \\ \text{Sim}(C_2, Q) + \sum_{a \in A_1} w_a \text{sim}_a(C_2, Q^+) + \sum_{a \in A_2} w_a \text{sim}_a(C_2, Q^+) \\ &\leq \text{Sim}(C_2, Q) + \sum_{a \in A_1} w_a + \sum_{a \in A_2} w_a \text{sim}_a(C_1, Q^+) \\ &< \text{Sim}(C_1, Q) + \sum_{a \in A_2} w_a \text{sim}_a(C_1, Q^+) \leq \text{Sim}(C_1, Q^+). \quad \square \end{aligned}$$

However, the *moving target* problem is only partially addressed by disregarding attributes for which C_1 and C_2 have the same values. Any increase in similarity gained by C_2 with respect to an attribute for which they have different values may also be gained in equal or greater measure by C_1 . Our third dominance criterion addresses this issue by taking account of the similarity between C_1 and C_2 with respect to each of the remaining attributes.

Lemma 1 If Sim is a regular similarity measure, then for any cases C_1, C_2 , query Q , and $a \in A$:

$$\text{sim}_a(C_2, Q) \leq \text{sim}_a(C_1, Q) + 1 - \text{sim}_a(C_1, C_2)$$

Proof By the triangle inequality, $1 - \text{sim}_a(C_1, Q) \leq 1 - \text{sim}_a(C_1, C_2) + 1 - \text{sim}_a(C_2, Q)$. The required inequality easily follows. $\square$

Theorem 3 If Sim is a regular similarity measure, then a given case C_2 is dominated by another case C_1 with respect to an incomplete query Q if and only if:

$$\text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a (1 - \text{sim}_a(C_1, C_2)) < \text{Sim}(C_1, Q)$$

Proof If the latter condition holds then it follows from Lemma 1 that for any completion Q^+ of Q :

$$\begin{aligned} \text{Sim}(C_2, Q^+) &= \text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a \text{sim}_a(C_2, Q^+) \\ &\leq \text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a (\text{sim}_a(C_1, Q^+) + 1 - \text{sim}_a(C_1, C_2)) \\ &< \text{Sim}(C_1, Q) + \sum_{a \in A - A_Q} w_a \text{sim}_a(C_1, Q^+) = \text{Sim}(C_1, Q^+). \end{aligned}$$

So C_2 is dominated by C_1 as required. It remains to show that if:

$$\text{Sim}(C_1, Q) \leq \text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a (1 - \text{sim}_a(C_1, C_2))$$

then C_2 is not dominated by C_1 . Let Q^+ be the completion of Q such that $\pi_a(Q^+) = \pi_a(C_2)$ for all $a \in A - A_Q$. It can be seen that $\text{sim}_a(C_2, Q^+) = 1$ and $\text{sim}_a(C_1, Q^+) = \text{sim}_a(C_1, C_2)$ for all $a \in A - A_Q$, and so:

$$\begin{aligned} \text{Sim}(C_1, Q^+) &= \text{Sim}(C_1, Q) + \sum_{a \in A - A_Q} w_a \text{sim}_a(C_1, Q^+) \\ &\leq \text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a (\text{sim}_a(C_1, Q^+) + 1 - \text{sim}_a(C_1, C_2)) \\ &= \text{Sim}(C_2, Q) + \sum_{a \in A - A_Q} w_a = \text{Sim}(C_2, Q^+). \end{aligned}$$

It follows as required that C_2 is not dominated by C_1 . $\square$

Figure 3 shows the numbers of dominated cases, on average, according to DC1, DC2 and DC3 after each question in simulated dialogues based on the PC case library [McGinty and Smyth, 2002]. The experimental setup is the same here as in Section 2.3, except that the dialogue is allowed to continue until no further attributes remain.

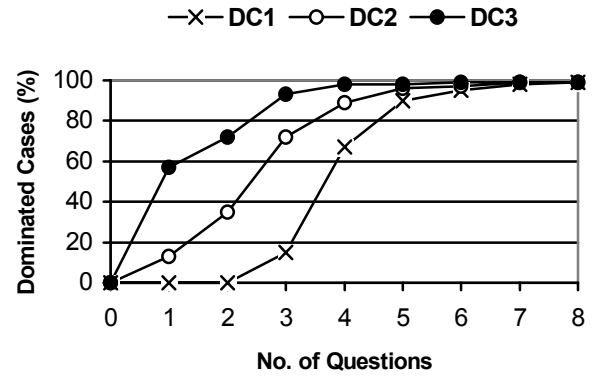


Figure 3. Numbers of dominated cases in the PC case library according to DC1, DC2, and DC3.

The results clearly show the inferiority of DC1 as a basis for detecting dominance relationships. It fails to detect any dominance relationships until three questions have been asked, while it can be seen from the results for DC3 that more than 70% of cases, on average, are in fact dominated after the second question. The results also show DC2 to be much more effective in detecting dominance relationships than DC1, though unable to compete with DC3.

4 Safely Terminating the Dialogue

We now identify conditions in which a recommender system dialogue can be terminated without loss of precision or recall. We also show that these conditions *must* be satisfied in order for the dialogue to be safely terminated. That is, termination on the basis of any other criterion runs the risk of some loss of solution quality.

Lemma 2 If Q is an incomplete query such that two cases $C_1, C_2 \in rs(Q)$ have the same values for all $a \in A - A_Q$, then $\text{Sim}(C_1, Q^+) = \text{Sim}(C_2, Q^+)$ for any completion Q^+ of Q . Moreover, any case C_3 that is dominated by C_1 is also dominated by C_2 .

Proof It easily follows from Proposition 1 that $\text{Sim}(C_1, Q^+) = \text{Sim}(C_2, Q^+)$ for any completion Q^+ of Q . If C_3 is

dominated by C_1 , then $\text{Sim}(C_3, Q) < \text{Sim}(C_1, Q) = \text{Sim}(C_2, Q)$ and for any completion Q^+ of Q , $\text{Sim}(C_3, Q^+) < \text{Sim}(C_1, Q^+) = \text{Sim}(C_2, Q^+)$. So C_3 is dominated by C_2 as required.

Theorem 4 *The dialogue can be safely terminated if and only if the current query Q is such that: (a) all cases in $rs(Q)$ have the same values for all $a \in A - A_Q$ and (b) for all $C_1 \in rs(Q)$ and $C_2 \notin rs(Q)$, C_1 dominates C_2 .*

Proof If (a) and (b) are true, then for any completion Q^+ of Q , all cases in $rs(Q)$ are equally similar to Q^+ by Lemma 2. On the other hand, if $C_3 \notin rs(Q)$, then (b) ensures that C_3 is less similar to Q^+ than any case in $rs(Q)$. It follows that $rs(Q^+) = rs(Q)$ for all completions Q^+ of Q , and so the dialogue can be safely terminated.

If (a) is not true, then there exist $C_1, C_2 \in rs(Q)$ and $a^\circ \in A - A_Q$ such that $\pi_{a^\circ}(C_1) \neq \pi_{a^\circ}(C_2)$. It follows that if Q^+ is the completion of Q such that $\pi_{a^\circ}(Q^+) = \pi_{a^\circ}(C_1)$ for all $a \in A - A_Q$, then $\text{sim}_{a^\circ}(C_2, Q^+) < \text{sim}_{a^\circ}(C_1, Q^+) = 1$, and so $\text{Sim}(C_2, Q^+) < \text{Sim}(C_1, Q^+)$. Thus $C_2 \notin rs(Q^+)$. It follows that (a) is a necessary condition for the dialogue to be safely terminated.

If (b) is not true, then there exist $C_1 \in rs(Q)$ and $C_2 \notin rs(Q)$ such that C_2 is not dominated by C_1 . As $\text{Sim}(C_1, Q) > \text{Sim}(C_2, Q)$, there must be a possible completion Q^+ of Q such that $\text{Sim}(C_2, Q^+) \geq \text{Sim}(C_1, Q^+)$. It follows that $rs(Q^+) \neq rs(Q)$, so the dialogue cannot be safely terminated. Thus (b) is also a necessary condition for the dialogue to be safely terminated. $\square$

The cost of testing condition (a) of Theorem 4 increases only linearly with the size of the retrieval set. At first sight, condition (b) may seem expensive to test, particularly in the early stages of query elicitation when $|rs(Q)|$ may be large. However, it can be seen from Lemma 2 that if (a) is true and C_1 is any case selected from $rs(Q)$, then (b) is true if and only if C_1 dominates C_2 for all $C_2 \notin rs(Q)$. Since (b) need only be tested if (a) is true, this reduces the cost of testing whether or not the dialogue can be safely terminated to one that increases only linearly with the size of the case library.

It is worth noting that failure of the underlying similarity measure to respect the triangle inequality does not affect the ability of a CCB algorithm that uses the termination criteria presented in Theorem 4 to provide perfect precision and recall. However, the use of DC2 (which does not rely on the triangle inequality) as the dominance criterion is likely to affect retrieval performance in terms of dialogue efficiency. In the rest of this paper, we assume that the underlying similarity measure is regular, thus permitting the use of DC3 in the identification of dominance relationships.

5 Attribute-Selection Strategies

We now examine the effects on dialogue efficiency of four approaches to attribute selection in CCB algorithms that use the termination criteria we have shown to be essential to ensure perfect precision and recall. Two of our algorithms are *goal driven* in that attribute selection is based on the

hypothesis that the preferred values of the remaining attributes are the values in a target case C_t initially selected at random and revised if necessary as the user's preferences are elicited. No change is needed as long as $C_t \in rs(Q)$, which means that no case is more similar to the current query Q than the target case. If $C_t \notin rs(Q)$, then the target case is replaced by another case selected at random from $rs(Q)$.

The algorithms compared in our evaluation are:

- CCBR1: Select attributes in random order
- CCBR2: Select attributes in order of decreasing importance
- CCBR3: Select the attribute that maximizes the similarity variance of cases not currently dominated by the target case
- CCBR4: Select the attribute that maximizes the number of cases dominated by the target case

Attribute selection in CCB3 is adapted from the approach proposed by Kohlmaier *et al.* [2001]. However, an *expected* similarity variance over all values of each attribute is not computed in our approach. Instead, the impact on similarity variance of each attribute is evaluated only for its value in the target case; this greatly reduces the computational effort involved in attribute selection.

Our experimental method is designed to compare average dialogue length for each attribute-selection strategy with the optimal dialogue length that can be achieved by any CCB algorithm that always gives perfect precision and recall. Any such algorithm, like the algorithms in our evaluation, must use the termination criteria identified in Theorem 4. As in our previous experiments, each case is temporarily removed from the case library and used to represent the preferences of the user. To determine the optimal dialogue length for a left-out case, we simulate all possible dialogues based on that case; that is, with the available attributes selected in every possible order. The optimal dialogue length is the minimum number of questions asked over all such dialogues. For each left-out case, we also record the dialogue length for each of our attribute-selection strategies.

For each strategy, Figure 4 shows the maximum, minimum, and average number of questions asked over all simulated dialogues. Similar statistics are shown for the optimal dialogues determined as described above. CCB4 gave the best performance, reducing the number of questions asked by up to 63% and by 35% on average relative to a full-length query. Its average dialogue length of 5.2 is only 4% higher than the lowest possible average that can be achieved by any CCB algorithm that guarantees perfect precision and recall.

Attribute selection based on similarity variance (CCB3) also performed well on this case library, with an average dialogue length of 5.4 compared with 7.4 for the random strategy (CCB1). With an average dialogue length of 5.8, selecting attributes in order of decreasing importance (CCB2) was also more effective in reducing average dialogue length than the random strategy.

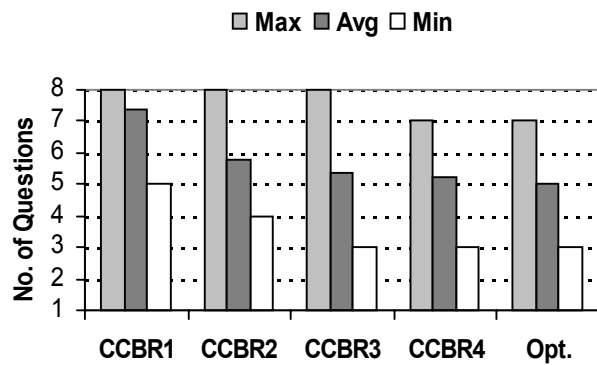


Figure 4. Comparison of attribute-selection strategies on the PC case library.

The case library used in our final experiment (www.ai-cbr.org) contains over 1,000 holidays and their descriptions in terms of 8 attributes such as price, region, duration, and season. The experimental setup is the same as in our previous experiment except that we do not attempt to determine the optimal length of each dialogue. The results are shown in Figure 5. Once again, CCBR4 gave the best performance, reducing dialog length by up to 63% and by 25% on average. On this occasion CCBR3, though reducing average dialogue length more effectively than CCBR1, was outperformed by CCBR2.

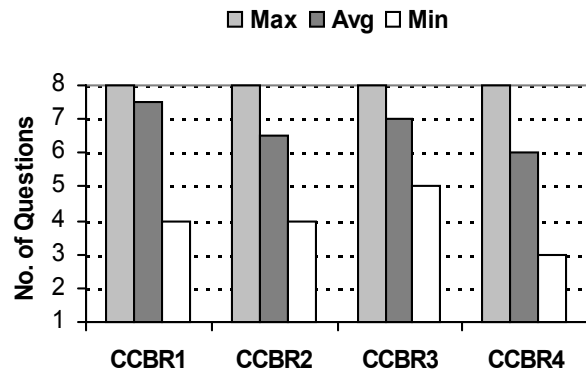


Figure 5. Comparison of attribute-selection strategies on the Travel case library.

6 Conclusions

Recognizing when problem-solving dialogues can be terminated while minimizing the impact on solution quality is an important issue that has received little attention in CBR research. Focusing on incremental query elicitation in recommender systems, we have identified necessary and sufficient conditions for the dialogue to be terminated without loss of solution quality. We have also evaluated several attribute-selection strategies in terms of dialogue efficiency given the requirement that there must be no loss of solution quality. The best results were obtained with a *goal-driven* strategy in which the selected attribute is the one that maximizes the number of cases dominated by the

target case. In spite of its low computational cost (linear in the size of the case library) this strategy gave close to optimal performance on the PC case library. It was also more effective in reducing average dialogue length than the other strategies evaluated on the Travel case library, a standard benchmark containing over 1,000 cases.

A feature of CBR recommender systems on which the techniques presented in this paper depend is that each outcome class (a unique product or service) is represented by a single case [McSherry, 2001]. Investigation of criteria for safe termination of problem-solving dialogues in other areas of CBR is an important objective for further research.

References

- [Aha *et al.*, 2001] David Aha, Leonard Breslow, and Héctor Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9-32, 2001.
- [Breslow and Aha, 1997] Leonard Breslow and David Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12:1-40, 1997.
- [Doyle and Cunningham, 2000] Michelle Doyle and Pádraig Cunningham. A dynamic approach to reducing dialog in on-line decision guides. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*, pages 49-60, Trento, 2000. Springer-Verlag.
- [Kohlmaier *et al.*, 2001] Andreas Kohlmaier, Sasha Schmitt, and Ralph Bergmann. A similarity-based approach to attribute selection in user-adaptive sales dialogues. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 306-320, Vancouver, Canada, 2001. Springer-Verlag.
- [McGinty and Smyth, 2002] Lorraine McGinty and Barry Smyth. Comparison-based recommendation. In *Proceedings of the Sixth European Conference on Case-Based Reasoning*, pages 575-589, Aberdeen, Scotland, 2002. Springer-Verlag.
- [McSherry, 2001] David McSherry. Minimizing dialog length in interactive case-based reasoning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 993-998, Seattle, Washington, 2001.
- [Quinlan, 1986] Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- [Schmitt *et al.*, 2002] Sascha Schmitt, Philipp Dopichaj, and Patricia Domínguez-Marín. Entropy-based vs. similarity influenced: attribute selection methods for dialogs tested on different electronic commerce domains. In *Proceedings of the Sixth European Conference on Case-Based Reasoning*, pages 380-394, Aberdeen, Scotland, 2002. Springer-Verlag.
- [Watson, 1997] Ian Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, San Francisco, California, 1997.

The Power of Suggestion*

Barry Smyth and Lorraine McGinty

Smart Media Institute, Department of Computer Science,
University College Dublin, Belfield, Dublin 4, Ireland.

Email: {barry.smyth , lorraine.mcginty}@ucd.ie

Abstract

User feedback is vital in many recommender systems to help guide the search for good recommendations. Preference-based feedback (e.g. “*Show me more like item A*”) is an inherently ambiguous form of feedback with a limited ability to guide the recommendation process, and for this reason it is usually avoided. Nevertheless we believe that certain domains demand the use of preference-based feedback. As such, we describe and evaluate a flexible recommendation strategy that has the potential to improve the performance of case-based recommenders that rely on preference-based feedback.

1 Introduction

At last year’s European Case-Based Reasoning conference (ECCBR 2002) in Scotland, delegates were treated to a whiskey tasting. A range of different whiskeys were sampled, each accompanied by a feature-based description (e.g. *distillery, age, alcohol content, cask, peatiness, sweetness, palette, etc.*). However, the value of these descriptions was limited as most of us had little or no understanding of the features, nor could we reliably reconcile individual features with the taste of a particular whiskey. Nevertheless virtually everyone could chose a favourite whiskey by the end of the session (after a lot of sampling it has to be said!). This example corresponds to a challenging problem for developers of recommender systems, for reasons that will become clear.

Normally the success of case-based recommenders depends on two important domain properties: (1) the items need to be described using well-defined features; and (2) users must have some understanding of these features and how they relate to their requirements. For example, restaurants are readily described using features such as *cuisine, location* and *price*, and most people have a reasonable understanding of their needs in relation to these features. When the above characteristics are present content-based recommendation strategies can be combined with different forms of user feedback, such as value elicitation (e.g. “I want an *expensive, Asian* restaurant”) [Bridge, 2001], or critiquing (e.g. “Show

me more like *Rolly’s Bistro* but *less formal*”), to good effect [Burke, 2000]. However, if these properties are not present then other strategies are needed. For example, if item descriptions are not available, collaborative filtering can be used to recommend items that have been rated positively by similar users [Burke, 2000; Shardanand and Maes, 1995]. The whiskey recommendation scenario above is different. The availability of item descriptions suggests case-based recommendation, but the lack of user expertise limits the use of feedback strategies such as value elicitation or critiquing.

Indeed the whiskey domain is not unique in this respect. In many domains users are able to express a preference without necessarily understanding individual item properties. This is especially true in domains where a specialised vocabulary already exists to describe items (e.g. fashion, jewellery, art, music, etc.). Consider a recommender system designed to help a bride-to-be to choose her ideal dress. Individual dresses are described using specific features (e.g. *coul-back, scoop-cut, georgian-style*, etc.) but most of these terms are meaningless to the average bride, and yet the typical bride-to-be is capable of recognising *what* she likes *when* she sees it. This problem is more than the user being unaware of the correct vocabulary to use during recommendation. The point is that even if they were presented with the appropriate vocabulary terms, they would not be in a position to use or appreciate these terms.

One form of feedback that could be used is preference-based feedback; the user simply expresses a preference for an item (e.g. “Show me more like dress 2”). This has been largely avoided by researchers - it is assumed to be too limited to efficiently guide the recommendation process. Nevertheless in our research we are interested in looking at how to make this a practical form of feedback in these challenging domains. Incidentally, unlike other feedback types, preference-based feedback also benefits from minimal interface needs, and so is particularly relevant for recommenders designed for use with the current generation of mobile devices, such as WAP phones and PDAs.

In this work we describe a way to improve the performance of recommenders that use preference-based feedback. Our novel *adaptive selection* method mirrors how real-life sales assistants adapt their recommendation strategies in response to user feedback by balancing the importance of similarity and diversity during each recommendation cycle. Dramatic improvements in recommendation performance have

*The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged

been observed (e.g. reductions in dialog length of up to 80%).

2 Comparison-Based Recommendation

Comparison-based recommendation is a content-based (or case-based) recommendation strategy [Burke, 2000; McGinty and Smyth, 2002], as opposed to a collaborative recommendation strategy [Burke, 2000; Konstan *et al.*, 1997; Shardanand and Maes, 1995]. The notion of comparison-based recommendation was introduced by [McGinty and Smyth, 2002] to emphasise the role of feedback in content-based recommenders, and to allow for an analysis of preference-based feedback in particular. The focus was on how query modification methods might be used with preference-based feedback to produce efficient recommendation dialogs without richer forms of feedback like value elicitation or critiquing. We briefly review this earlier work, describing the basic comparison-based recommendation framework and some successful query modification strategies.

2.1 The Basic Algorithm

Comparison-based recommendation (Figure 1) is an iterative recommendation algorithm that presents the user with a selection of k items as recommendations during each of a sequence of n recommendation cycles. During each cycle the user expresses a *preference* for one of the suggestions and this *feedback* is used to *revise* the query for the next cycle; in the vocabulary of Shimazu, it is a type of *navigation by proposing* [Shimazu, 2001]. The *recommendation dialog* terminates when the user is presented with an acceptable suggestion.

2.2 Simple Selection: More Like This

The simplest approach to comparison-based recommendation is to recommend the k most similar items to the current query, and use the newly preferred item as the query for the recommendation next cycle. This *more like this* (MLT) method is a facility often provided by Web search engines; although it should be noted that search engines rarely rely upon this method, primarily focusing instead on query elicitation and term-based search. The MLT method is flawed by a tendency to follow so-called *false-leads* as it overfits to the current preference, and this results in protracted recommendation dialogs. Using each preferred item as the next query is problematic if some of its features are not relevant to the user. For example, a user may indicate a preference for a whiskey that is *oak-aged*, *sweet*, and *peaty*. But if the preference was largely a result of the peatiness and independent of, or even at odds with, the sweetness or the aging, then blindly focusing the next selection on *sweet* and *oak-aged* whiskeys, as well as *peaty* ones, may result in poor recommendations.

Figure 1 shows the scope of this problem by graphing the similarity of the preference to the target in a sample recommendation dialog. Instead of an increasing similarity profile, we find sustained decreases in similarity as MLT follows false-leads. Between cycle 3 and 11, target similarity falls from 0.46 to as low as 0.26 as the user is forced to accept poor recommendations, and again between cycle 11 and 40 there is a noticeable similarity trough. Indeed it is only after cycle 38 that MLT finally begins to focus on the target region and similarity rises.

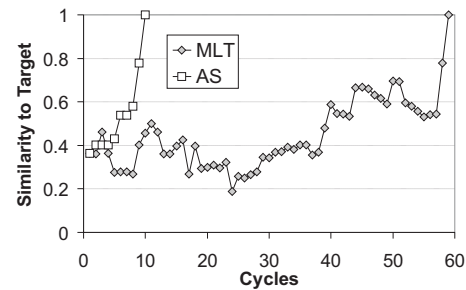


Figure 1: Profiling the similarity of the preference case to the target case in a typical recommendation session

2.3 Query Modification Strategies

Query modification techniques can relieve the false-lead problem, and have been shown to improve recommendation efficiency, reducing MLT dialogs by 30% on average [McGinty and Smyth, 2002]. The central idea is to identify features in the preference case that *are* likely to be the reason for the user's choice, and those features that are likely to be irrelevant. Only relevant features are transferred to the new query. Returning to the whiskey example above, if the rejected cases are also very sweet then sweetness may be irrelevant to the user. Hence, one query modification strategy, *partial MLT* (pMLT) only includes preference features in the query that are not found in any of the rejected cases.

[McGinty and Smyth, 2002] also propose more sophisticated query modification strategies that weight preference features in the query according to how likely they are to be reliable. One strategy computes a weight for a preference feature based on the proportion of rejected items that also contain that feature. If 3 cases are recommended to the user, and if the preference and just one of the rejected items are *oak-aged* and then this feature is transferred to the new query with a weight of 0.5 (50% of the rejected items are *oak-aged*).

3 Adaptive Case Selection

So far we have assumed the use of similarity-based selection during each cycle. However, in the real-world two selection strategies can be observed in the dialogs that take place between customers and sales assistants. If a customer is unsure of their exact needs the sales assistant will tend to present a diverse range of alternatives, based on a preliminary set of requirements, to *focus* in on a promising region of a product space. A good sales assistant can recognise when the customer sees something that they genuinely like and uses this as a cue to switch their strategy to one that tries to *refine* subsequent recommendations in the region of the preference by selecting similar items.

In this paper, instead of using query modification methods to improve recommendation efficiency, we propose an alternative called *adaptive selection*. It adapts the way that new items are selected for recommendation and is motivated by the above observation that different selection strategies are appropriate at different times in the recommendation process.

3.1 Similarity vs Diversity

Our observation about real-life recommendation scenarios is partly echoed by recent research that has questioned the apparent over-emphasis on similarity in content-based recommenders, with diversity forwarded as an important additional selection constraint. A number of strategies for improving recommendation diversity have been suggested. [Shimazu, 2001] proposes the recommendation of three items or cases, i_1 , i_2 and i_3 , per recommendation cycle: i_1 is the most similar case to the query; i_2 is maximally dissimilar from i_1 ; and i_3 is maximally dissimilar from i_1 and i_2 . By design these items are maximally diverse, but similarity to the query may be compromised as there are no guarantees that i_2 and i_3 will be very similar to the query.

Other researchers have proposed alternative diversity enhancing mechanisms with a more manageable balance between item diversity and query similarity [Bridge, 2001; McSherry, 2002; Smyth and McClave, 2001]. Although all of these techniques introduce diversity into selection in a uniform way, they do not fully reflect the strategy switching seen in real-world scenarios. Moreover, while increasing diversity may improve the efficiency of a recommender by allowing it to cover more of the item-space per recommendation cycle, it may also lead to new types of inefficiencies. Diversity enhancing methods pass over items that might otherwise be selected and the target item may be one of these.

A more flexible mechanism for introducing diversity must be adopted based on whether or not the recommender has focussed on the correct region of the item space. If there is evidence that it is correctly focused then a pure similarity-based selection method can be used to refine the recommendations and safe-guard against passing over the target item. If the recommender is not correctly focused, then diversity is introduced in order to maximise the coverage of the item space in the next cycle, and so help to refocus the recommender by offering the user contrasting recommendations.

3.2 Preference Carrying

Unfortunately it is not immediately obvious how to judge whether the recommender is correctly focused; the target item is unknown and user requirements are typically vague. Adaptive selection solves this by evaluating whether the recommendations made in the i^{th} cycle are an improvement on those in the $i - 1^{th}$ cycle before choosing a selection strategy for the $i + 1^{th}$ cycle.

This is achieved by *carrying the preference* item from the previous cycle to the current cycle. If the user selects the carried preference in the current cycle it must mean that they are unhappy with the $k - 1$ new alternatives in that cycle. These new items must have failed to improve upon the recommendations made during the previous cycle, and so the recommender must not be correctly focussed. If the user ignores the carried preference and selects one of the newly recommended items then the recommender must be correctly focused. Thus, by carrying the preference item, and monitoring whether or not the user reselects it, we can implement a switching mechanism between two alternative selection strategies: a *refine* strategy that emphasises similarity; and a *refocus* strategy that

balances similarity and diversity for improved recommendation coverage.

In theory carrying the preference may lead to new inefficiencies because the carried preference takes up a valuable slot in each cycle, thus limiting recommendation coverage. However, this is easily compensated for because carrying the preference helps protect against false-leads. If none of the $k - 1$ new cases are relevant then by reselecting the carried preference the user is at least maintaining the previous best recommendation rather than being forced to accept a lower quality false-lead. In practice, this on its own can offer a substantial improvement to recommendation efficiency.

3.3 Refine & Refocus

As mentioned above, the refine strategy makes use of a standard similarity-based selection method, picking $k - 1$ new items that are maximally similar to the current query. The refocus strategy uses the bounded-greedy diversity technique proposed by [Smyth and McClave, 2001]; see Figure 2.

Very briefly, the bounded-greedy technique involves two basic phases. First, the bk most similar items to the query are selected (where b is typically an integer between 2 and 5). During the second phase, the set (R) of selected items is built incrementally. During each step of this build the remainder of the bk items are ordered according to their *quality* and the highest quality item added to R . The quality of a item i is proportional to the similarity between i and the current query q , and to the diversity of i relative to those items so far selected, $R = \{r_1, \dots, r_m\}$; see Equations 1 & 2.

$$Quality(q, i, R) = \alpha * Sim(q, i) + (1 - \alpha) * Div(i, R) \quad (1)$$

$$\begin{aligned} Div(i, R) &= 1 \text{ if } R = \{\}; \\ &= \frac{\sum_{j=1..m} (1 - Sim(i, r_j))}{m} \text{ otherwise} \end{aligned} \quad (2)$$

The first case to be selected is always the most similar to the query and in subsequent iterations, the chosen case has the highest quality relative to the query and diversity and cases selected so far. Note, we set $b = 3$ and $\alpha = 0.5$ to balance similarity and diversity during refocusing.

4 Evaluation

In this section we evaluate the performance of adaptive selection focusing on the average number of cycles and unique items that must be presented to a user in a typical recommendation dialog. The shorter the dialog, the more successful the recommender is likely to be, and we demonstrate that adaptive selection delivers dramatic reductions in dialog length.

4.1 Setup

A case-base of 585 unique Scottish whiskey cases (Figure 3) is used to compare two comparison-based recommenders using preference-based feedback: MLT uses the standard MLT strategy; MLT+AS implements adaptive selection. We use a leave-one-out methodology for testing using 200 randomly selected cases as test cases. Each test case (*base*) is removed from the case-base and used in two ways.

<pre> 1. define Comparison-Based-Recommend(q, CB, k) 2. $i_{p-1}, i_p \leftarrow \text{null}$ 3. do 4. $R \leftarrow \text{ItemRecommend}(q, CB, k, i_p, i_{p-1})$ 5. $i_p \leftarrow \text{UserReview}(R, CB)$ 6. $Q \leftarrow \text{QueryRevise}(q, i_p, R)$ 7. $i_{p-1} \leftarrow i_p$ 8. until UserAccepts(i_p) 9. define QueryRevise(q, i_p, R) 10. $R' \leftarrow R - \{i_p\}$ 11. $q \leftarrow i_p$ 12. return q 13. define UserReview(R, CB) 14. $i_p \leftarrow \text{user's preferred case from } R$ 15. $CB \leftarrow CB - R$ 16. return i_p </pre>	<pre> 17. define ItemRecommend(q, CB, k, i_p, i_{p-1}) 18. if ($i_p \neq \text{null}$) && ($i_p == i_{p-1}$) 19. $R \leftarrow \text{ReFocus}(q, CB, k)$ 20. else 21. $R \leftarrow \text{ReFine}(q, CB, k)$ 22. return R 23. define ReFine(q, CB, k) 24. $CB' \leftarrow \text{sort } CB \text{ in decreasing order of their sim to } q$ 25. $R \leftarrow \text{top } k \text{ items in } CB'$ 26. return R 27. define ReFocus(q, CB, k, i_p, i_{p-1}) 28. return BoundedGreedySelection(q, CB, k, b) 29. define BoundedGreedySelection(q, CB, k, b) 30. $CB' := \text{bk cases in } CB \text{ that are most similar to } q$ 31. $R := \{\}$ 32. For j := 1 to k 33. Sort CB' by $\text{Quality}(q, i, R)$ for each case i in CB' 34. $R := R + \text{First}(CB')$ 35. $CB' := CB' - \text{First}(CB')$ 36. EndFor 37. return R </pre>
---	--

Figure 2: The Comparison-Based Recommendation algorithm with adaptive selection.

CASE ID	DISTILLERY	AGE	PROOF	SWEETNESS	PEATINESS	AVAILABILITY	COLOR	NOSE	FLAVOR/PALATTE	FINISH
Case 500	The Glenlivet	13	40	7	4	5	gold	sweet	medium-peat	full-body

Figure 3: The features of a sample whiskey case include nominal and continuous features.

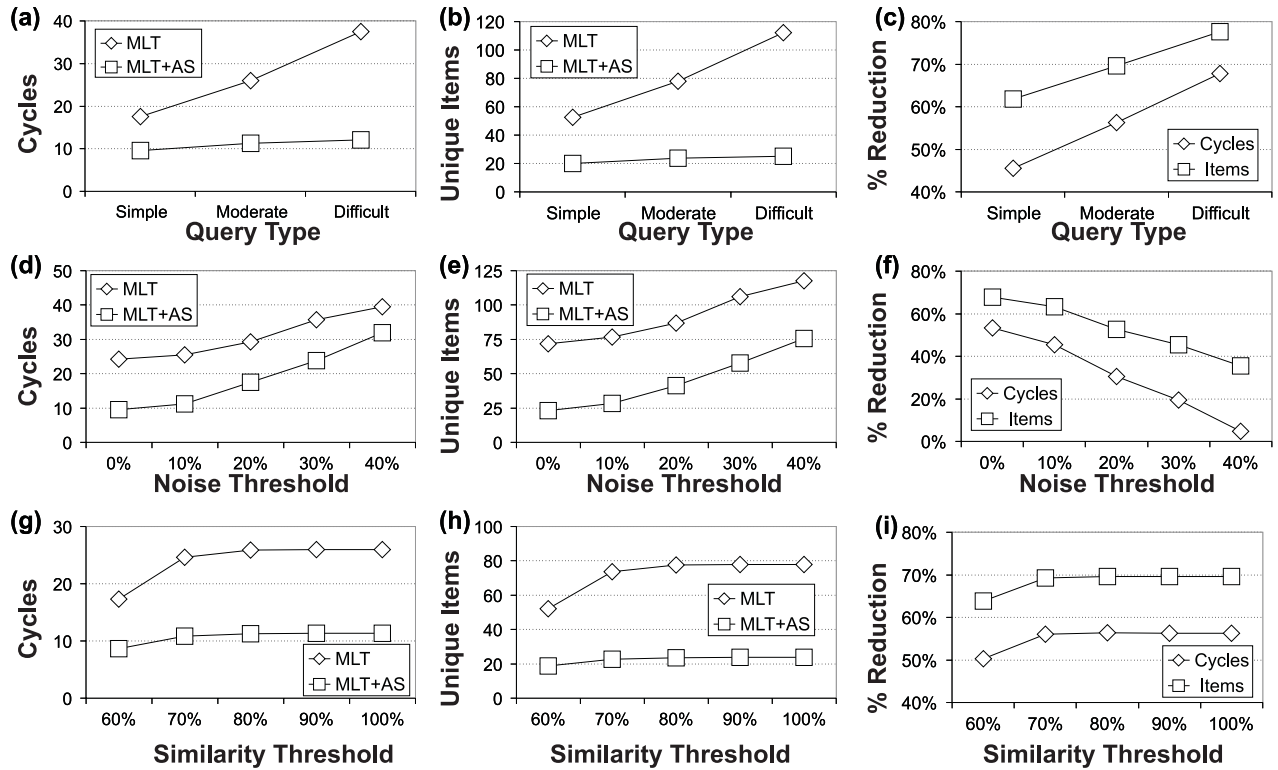


Figure 4: Key experimental results

First it serves as the basis for a set of queries made by taking random subsets of features. Second, it is used to identify a maximally similar (*target*) case. Thus, the base is the ideal query for a user, the generated query is the initial test query given to the recommender, and the target is the best case for the user given their ideal. During each recommendation cycle 3 cases are recommended to the user, and the most similar to the target is chosen as a preference, thus simulating a user that is capable of correctly selecting the best preference item in each cycle; we relax this assumption in Section 4.3. A query is satisfied when the target is returned in a cycle, thereby simulating the situation where the user is seeking a specific case; once again, we relax this condition in Section 4.4. In total 1250 queries are generated and divided into three groups based on their difficulty; where difficulty is based on the number of recommendation cycles required by MLT.

4.2 Recommendation Efficiency

Basic recommendation efficiency can be measured in terms of the average number of cycles (or unique cases or items) that a user must work through before being presented with their ideal target. The leave-one-out method outlined above is used by the two recommenders for each of the 1250 queries and the mean number of cycles and unique items presented to the user are measured. The results are shown in Figure 4(a-b) as graphs of mean cycles and items for each algorithm and query group. The benefits of MLT+AS should be clear for both measures. For example, we see that MLT requires 17.5 cycles (and 52.5 items) for simple queries, but MLT-AS needs only 9.5 cycles (and 20 items), a relative reduction of 45% in terms of cycles, and 62% in terms of items.

We see similar results for the other query sets. In fact, the relative reductions enjoyed by MLT-AS increase with query difficulty. The recommendation dialogs associated with more difficult queries include more false-leads than the dialogs for simpler queries, and so offer adaptive selection an even greater opportunity for dialog reduction. This trend is clearly seen in Figure 4(c), which graphs the percentage reduction in cycles and items due to MLT-AS, relative to MLT. For cycles, the relative reduction grows from 45% (simple) to 68% (difficult) and for unique items it grows from 62% to 78%. These results clearly show a dramatic benefit for adaptive selection. Indeed this benefit is so great that MLT-AS is capable of satisfying the most difficult queries far more efficiently than MLT takes to satisfy even the simplest queries.

4.3 Preference Noise

In this experiment we re-evaluate our recommenders by relaxing the assumption that the user always prefers the most similar item to the target to test whether benefits are found with sub-optimal preferences. The above experiment is repeated except that noise is introduced into the preference selection by perturbing the similarities between each recommended item and the target by some random amount within a set noise limit. A 10% noise means that each similarity value can change by up to $\pm 10\%$ of its actual value. This will potentially change the internal ordering of recommended items during each cycle resulting in the selection of a preference that may not be the most similar to the target. This approach

mimics the situation where users are likely to make preference mistakes more frequently if there is little difference between the target similarities of the recommended items.

Figure 4(d&e) graph the mean number of cycles and items presented to the user versus the noise limit for moderate queries. As expected, introducing preference noise has a negative impact on the ability of each recommender to locate the target item. MLT dialogs increase from 24 cycles at 0% noise to 39 cycles at 40% noise, and MLT-AS dialogs increase from 11 cycles to 37 cycles. The number of items required is also seen to increase in a similar manner. However, the benefits of MLT-AS remain across all levels of noise (see Figure 4(f)) although the magnitude of these benefits is seen to fall as noise increases. For example, the MLT-AS benefit, in terms of unique items, falls from 68% at the 0% noise level to 36% at the 40% level. Nevertheless, adaptive selection once again offers significant improvements in recommendation efficiency even when users make imperfect preference selections. While the results presented here focus only on moderate queries, for reasons of brevity, it is worth noting that qualitatively similar results are found for simple and difficult queries. Once again the MLT-AS benefits increase with query difficulty; for example, in terms of unique items, at the 40% noise level, a 27% MLT-AS benefit is observed for simple queries, and a 43% MLT-AS benefits for difficult queries.

4.4 Target Noise

Our second key assumption is that users are interested in a single target item during recommendation, and that the dialog only terminates when this item is returned. It is perhaps more realistic to assume that the user will be satisfied by any one of a group of items that are similar to the optimal target. We re-evaluate our recommenders under this more relaxed termination condition by repeating the basic efficiency experiment except that we terminate each dialog once an item has been recommended that is within some pre-defined similarity of the target. A similarity threshold of 70% means that the dialog terminates when an item that is at least 70% similar to the target has been recommended.

Figure 4(g-i) presents the results in a number of ways. First Figures 4(g&h) graph the mean number of cycles and items presented to the user versus the similarity threshold for moderate queries. As expected, relaxing the termination condition results in shorter recommendation dialogs for MLT and MLT-AS. For example, MLT dialogs reduce from just under 26 cycles at the 100% similarity threshold (where the optimal item must be recommended) to just over 17 cycles at 60% similarity. MLT-AS dialogs reduce from 11 cycles to just under 9 cycles across the same similarity range. The number of items required is also seen to reduce in a similar manner. However, positive MLT-AS benefits are maintained across all similarity thresholds (see Figure 4(i)). In fact, as the threshold increases, and the termination condition becomes more rigid, we find increasing benefits for MLT-AS in terms of cycles and items. For example, the MLT-AS benefit, in terms of unique items, grows from 54% at the 60% similarity threshold to just under 70% at the 100% threshold. Once again, adaptive selection delivers significant improvements in recommendation efficiency across a variety of termination conditions.

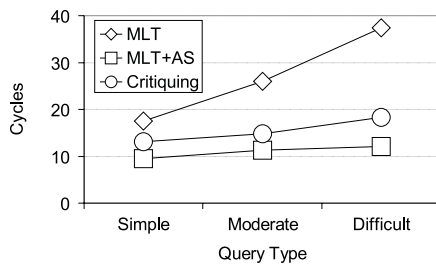


Figure 5: Preference-Based Feedback vs Critiquing.

4.5 Preference-Based vs Critiquing

Finally, it is worth briefly considering the impressive dialog reductions observed for adaptive selection with preference-based feedback in the context of a much richer form of feedback, namely critiquing [Burke, 2000; McGinty and Smyth, 2003]. Figure 5 presents an efficiency comparison (in terms of average recommendation cycles) between our two preference-based recommenders and an equivalent recommender that implements critiquing; briefly, during each cycle feedback corresponds to a preference case *plus* a critique that constrains a selected feature value.

On the face of it we expect critiquing to produce more efficient dialogs than preference-based feedback. And so it does for all query types when compared to standard preference-based feedback. However, the surprise is that combining preference-based feedback and adaptive selection produces recommendation dialogs that are consistently and significantly shorter than even those produced by critiquing. For example, relative to MLT-AS, critiquing requires 37% more cycles for simple queries and 52% more cycles for the difficult queries.

5 Conclusions

To date preference-based feedback has been largely ignored by recommender systems. It is an inherently ambiguous form of feedback with a limited ability to efficiently guide the recommendation process. Nevertheless, we believe that this form of feedback is useful, and perhaps even vital, in certain recommendation domains where other forms of feedback cannot be used, perhaps because of limited user expertise or even basic device restrictions.

We have described how preference-based feedback can be made more efficient using the adaptive selection technique, which modifies its recommendation strategy depending on whether or not the recommender is correctly focused on the right region of the recommendation space. Using this method we have demonstrated dramatic performance improvements over standard preference-based feedback, across a variety of experimental conditions, with reductions in dialog length of up to nearly 80%. Indeed, adaptive selection is so effective that it is capable of using simple and cheap preference-based feedback to produce recommendation dialogs that are even more efficient than those available with richer forms of feedback such as critiquing.

In this paper we have presented evaluation results from one recommendation domain, Whiskey. This domain is especially

interesting because of its dependency on preference-based feedback (as discussed in Section 1). We have also tested our technique on more traditional domain datasets (i.e. Travel, PC) and have found similar results [McGinty and Smyth, 2003]. We believe that these results have the potential to change the perception of simple preference-based feedback, facilitating its practical use in a variety of recommendation scenarios. Indeed adaptive selection is in no way limited to preference-based feedback. In our future research we intend to investigate if similar performance benefits can be achieved by integrating this technique in recommenders that use other forms of feedback (e.g. *critiquing* and *rating-based*).

References

- [Bridge, 2001] D. Bridge. Product Recommendation Systems: A New Direction. In D. Aha and I. Watson, editors, *Workshop on CBR in Electronic Commerce at The International Conference on Case-Based Reasoning (ICCBR-01)*, 2001. Vancouver, Canada.
- [Burke, 2000] R. Burke. Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Systems*, 69(32), 2000.
- [Konstan *et al.*, 1997] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gorgan, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
- [McGinty and Smyth, 2002] L. McGinty and B. Smyth. Comparison-Based Recommendation. In Susan Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*, pages 575–589. Springer, 2002. Aberdeen, Scotland.
- [McGinty and Smyth, 2003] L. McGinty and B. Smyth. The Role of Diversity in Conversational Recommender Systems. In D. Bridge and K. Ashley, editors, *Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03)*. Springer, 2003. Troindheim, Norway.
- [McSherry, 2002] D. McSherry. Diversity-Conscious Retrieval. In Susan Craw, editor, *Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02)*, pages 219–233. Springer, 2002. Aberdeen, Scotland.
- [Shardanand and Maes, 1995] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating “Word of Mouth”. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, pages 210–217. ACM Press, 1995. New York, USA.
- [Shimazu, 2001] H. Shimazu. ExpertClerk : Navigating Shoppers’ Buying Process with the Combination of Asking and Proposing. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 1443–1448. Morgan Kaufmann, 2001. Seattle, Washington, USA.
- [Smyth and McClave, 2001] B. Smyth and P. McClave. Similarity v’s Diversity. In D. Aha and I. Watson, editors, *Proceedings of the International Conference on Case-Based Reasoning*, pages 347–361. Springer, 2001.

A Weighted Polynomial Information Gain Kernel for Resolving Prepositional Phrase Attachment Ambiguities with Support Vector Machines

Bram Vanschoenwinkel*

Bernard Manderick

Vrije Universiteit Brussel, Department of Computer Science, Computational Modeling Lab
Pleinlaan 2, 1050 Brussel, Belgium
{bvschoen@ – bernard@arti}.vub.ac.be

Abstract

We introduce a new kernel for Support Vector Machine learning in a natural language setting. As a case study to incorporate domain knowledge into a kernel, we consider the problem of resolving Prepositional Phrase attachment ambiguities. The new kernel is derived from a distance function that proved to be succesful in memory-based learning. We start with the Simple Overlap Metric from which we derive a Simple Overlap Kernel and extend it with Information Gain Weighting. Finally, we combine it with a polynomial kernel to increase the dimensionality of the feature space. The closure properties of kernels guarantee that the result is again a kernel. This kernel achieves high classification accuracy and is efficient in both time and space usage. We compare our results with those obtained by memory-based and other learning methods. They make clear that the proposed kernel achieves a higher classification accuracy.

1 Introduction

An important issue in natural language analysis is the resolution of structural ambiguity. A sentence is said to be structurally ambiguous when it can be assigned to more than one syntactic structure [Zavrel *et al.*, 1997]. In *Prepositional Phrase (PP)* attachment one wants to disambiguate between cases where it is uncertain whether the PP attaches to the verb or to the noun.

Example 1 Consider the following two sentences:

1. *I bought the shirt with pockets.*
2. *I washed the shirt with soap.*

In sentence 1, **with** modifies the noun **shirt** because **with pockets (PP)** describes the **shirt**. In sentence 2 however, **with** modifies the verb **washed** because **with soap (PP)** describes how the shirt is **washed** [Ratnaparkhi, 1998].

This type of attachment ambiguity is easy for people to resolve because they can use their world knowledge [Stetina

* Author funded by a doctoral grant of the institute for advancement of scientific technological research in Flanders (IWT).

and Nagao, 1997]. A computer program usually cannot rely on that kind of knowledge.

This problem has already been tackled using memory-based learning like for example k -nearest neighbours. Here, the training examples are first stored in memory and classification of a new example is done based on the closest example stored in memory. Therefore, one needs a function that expresses the distance or similarity between examples. There already exist several dedicated distance functions to solve all kind of natural language problems using memory-based learning [Veenstra *et al.*, 2000; Zavrel *et al.*, 1997; Daelemans *et al.*, 2002].

We will use a Support Vector Machine (SVM) to tackle the problem of PP attachment disambiguation. Central to SVM learning is the kernel function $K : X \times X \rightarrow \mathbb{R}$ where X contains the examples and the kernel K calculates an inner product in a second space, the feature space F . This product expresses how similar examples are.

Our goal is to combine the power of SVMs with the distance functions that are well-suited for the problem for which they were designed. Deriving a distance from a kernel is straightforward, see Section 2.1. However, deriving a kernel from a distance is not trivial since kernels must satisfy some extra conditions, i.e. being a kernel is a much stronger condition than being a distance. In this paper we will describe a method that shows how such dedicated distance functions can be used as a basis for designing kernels that sequentially can be used in SVM learning.

We use the PP attachment problem as a case study to illustrate our approach. As a starting point we take the *Overlap Metric* that has been succesfully used in memory-based learning for the same problem [Zavrel *et al.*, 1997].

Section 2 will give a short overview of the theory of SVMs together with some theorems and definitions that are needed in Section 4. Based on [Zavrel *et al.*, 1997], section 3 gives an overview of metrics developed for memory-based learning applied to the PP attachment problem. In Section 4 the new kernels will be introduced. Finally Sections 5 and 6 give some experimental results and a conclusion of this work.

2 Support Vector Machines

For simplicity, in our explanation we will consider the case of binary classification only, i.e. we consider an input space X with input vectors $\mathbf{x}$ and a target space $D = \{1, -1\}$. The

goal of the SVM is to assign every $\mathbf{x}$ to one of two classes $D = \{1, -1\}$. The decision boundary that separates the input vectors belonging to different classes is usually an arbitrary $n - 1$ -dimensional manifold if the input space X is n -dimensional.

One of the basic ideas behind SVMs is to have a mapping Φ from the original input space X into a high-dimensional feature space F that is a Hilbert space, i.e. a complete vector space provided with an inner product. Separation of the transformed feature vectors $\Phi(\mathbf{x})$ in F is done linearly, i.e. by a hyperplane. Cover's theorem states that any consistent training set can be made linear separable provided the dimension of F is high enough.

However, transforming the training set $TS = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\} \subset X$ into such a higher-dimensional space incurs both computational and learning-theoretic problems. The high dimensionality of F makes it very expensive both in terms of memory and time to represent the feature vectors $\Phi(\mathbf{x}_i)$ corresponding to the training vectors $\mathbf{x}_i$. Moreover, separating the data in this way exposes the learning system to the risk of *overfitting* the data if the separating hyperplane is not chosen properly.

SVMs sidestep both difficulties [Vapnik, 1998]. First, overfitting is avoided by choosing the unique *maximum margin hyperplane* among all possible hyperplanes that can separate the data in F . This hyperplane maximizes the distance to the closest data points.

Second, the maximum margin hyperplane in F can be represented entirely in terms of training vectors $\mathbf{x}_i \in TS$ and a kernel K .

Definition: A *kernel* is a function $K : X \times X \rightarrow \mathbb{R}$ so that for all $\mathbf{x}$ and $\mathbf{y}$ in X , $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$ where Φ is a (non-linear) mapping from the input space X into the Hilbert space F provided with the inner product $\langle \cdot, \cdot \rangle$ [Cristianini and Shawe-Taylor, 2000].

To be more precise, once we have chosen a kernel K we can represent the maximal margin hyperplane (or decision boundary) by a *linear equation* in $\mathbf{x}$

$$\sum_{\mathbf{x}_i \in TS} \alpha_i d_i K(\mathbf{x}_i, \mathbf{x}) + b = 0 \quad (1)$$

where the training vectors $\mathbf{x}_i$ and their class labels d_i are given. The decision boundary is determined completely once we know the α_i and b . They are obtained by optimizing a convex quadratic objective function with linear constraints. Moreover, most of the α_i prove to be zero. By definition the vectors $\mathbf{x}_i$ corresponding with non-zero α_i are called the *support vectors* SV and this set consists of those data points that lie closest to the hyperplane and thus are the most difficult to classify.

In order to classify a new point $\mathbf{x}_{new}$, one determines the sign of

$$\sum_{\mathbf{x}_i \in SV} \alpha_i d_i K(\mathbf{x}_i, \mathbf{x}_{new}) + b$$

If this sign is positive $\mathbf{x}_{new}$ belongs to class 1, if negative to class -1, if zero $\mathbf{x}_{new}$ lies on the decision boundary. Note that we have now restricted the summation to the set SV of support vectors because the other α_i are zero anyway.

To conclude, SVM can sidestep the above two difficulties because neither the feature space F nor the map Φ from the input space X into F are explicitly defined, they are replaced by the kernel K that operates on vectors of the input space X .

2.1 Some Properties of Kernels

The selection of an appropriate kernel K is the most important design decision in SVMs since it implicitly defines the feature space F and the map Φ . A SVM will work correctly even if we don't know the exact form of the features that are used in F . Moreover, the kernel expresses prior knowledge about the patterns being modelled, encoded as a similarity measure between two vectors [Brown *et al.*, 1999].

But not all maps over $X \times X$ are kernels. Since a kernel K is related to an inner product, cfr. the definition above, it has to satisfy some conditions that arise naturally from the definition of an inner product and are given by Mercer's theorem: the map must be continuous and positive definite [Vapnik, 1998].

In this paper we will use following methods to construct kernels [Cristianini and Shawe-Taylor, 2000]:

- M1 Making kernels from kernels: Based on the fact that kernels satisfy a number of closure properties. In this case, the Mercer conditions follow naturally from the closure properties of kernels.
- M2 Making kernels from features: Start from the features of the input vectors and obtain a kernel by working out their inner product. A feature is a component of the input vector. In this case, the Mercer conditions follow naturally from the definition of an inner product.

The set of kernels is closed under certain operations, this allows us to make new kernels from existing ones (see M1). We mention only the closure property used in the paper:

Closure properties of kernels: Let $\Phi : X \rightarrow \mathbb{R}^n$ be a map and K_1 be a kernel over $\mathbb{R}^n \times \mathbb{R}^n$ then the following function is a kernel $K(\mathbf{x}, \mathbf{y}) = K_1(\Phi(\mathbf{x}), \Phi(\mathbf{y}))$.

Another interesting property is that we can derive a distance d from a kernel K as follows:

Distance derived from a kernel: Let K be a kernel over $X \times X$, then d defined as $d(\mathbf{x}, \mathbf{y}) = \sqrt{K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{y}) + K(\mathbf{y}, \mathbf{y})}$ is a distance on X and consequentially (X, d) is a metric space.

3 Metrics for Memory-based Learning

In this section, we will focus on the distance functions [Zavrel *et al.*, 1997; Cost and Salzberg, 1993] used for memory-based learning with symbolic values. First, we will have a look at the *Simple Overlap Metric* (SOM) and next we will discuss *Information Gain Weighting* (IGW). Memory-based learning is a class of machine learning techniques where training instances are stored in memory first and classification of new instances later on is based on the distance (or similarity) between the new instance and the closest training instances that have already been stored in memory. A well-known example of memory-based learning is *k-nearest neighbours classification*. We will not go into further detail, the literature offers

many books and articles that provide a comprehensive introduction to memory-based learning, e.g. [Mitchell, 1997]. For our purpose, the important thing to remember is that we will be working with symbolic values like strings over an alphabet of characters. Instances are n -dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ in some universe Ω^n . The components of the vectors in Ω^n are strings, i.e. $a_i, b_i \in \Sigma^*$, with Σ^* the set of all strings over some alphabet Σ . Notice that for $n = 1$, Ω^n becomes Ω and the elements of Ω are strings in Σ^* . In order for the inner products defined below to be meaningful, the set of strings Ω has to be finite. As a result the set of vectors Ω^n is also finite as required.

3.1 Simple Overlap Metric

The most basic metric for vectors with symbolic values is the *Simple Overlap Metric* (SOM) [Zavrel et al., 1997]:

$$d_{SOM} : \Omega^n \times \Omega^n \rightarrow \mathbb{R}^+ : d_{SOM}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n \delta(a_i, b_i) \quad (2)$$

with

$$\delta(a_i, b_i) = 1 - \delta_{a_i}(b_i) \quad (3)$$

and

$$\delta : \Omega \times \Omega \rightarrow \{1, 0\} : \delta_{a_i}(b_i) = 1 \text{ if } a_i = b_i, \text{ else } 0 \quad (4)$$

Here $d_{SOM}(\mathbf{a}, \mathbf{b})$ is the distance between the vectors $\mathbf{a}$ and $\mathbf{b}$, represented by n features and δ is the distance per feature. The k -nearest neighbour algorithm equipped with this metric is called IB1 [Aha et al., 1991]. The IB1 algorithm simply counts the number of (mis)matching feature values in both vectors. This is a reasonable choice if we have no information about the importance of the different features. But if we do have information about feature importance then we can add linguistic bias to weight the different features.

3.2 Information Gain Weighting

Information Gain Weighting (IGW) measures for every feature i separately how much information it contributes to our knowledge of the correct class label. The Information Gain (IG) of a feature i is measured by calculating the entropy between the cases with and without knowledge of the value of that feature:

$$w_i = H(C) - \frac{\sum_{v \in V_i} P(v) \times H(C|v)}{si(i)} \quad (5)$$

$$si(i) = - \sum_{v \in V_i} P(v) \log_2 P(v)$$

Where C is the set of class labels, V_i is the set of values for feature i , $H(C) = - \sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels and $si(i)$ is the split info. The resulting weights w_i can be used to extend Equation 2 with weights (see Equation 6). The k -nearest neighbour algorithm equipped with this metric is called IB1-IG [Zavrel et al., 1997] and the corresponding distance called d_{IG} :

$$d_{IG} : \Omega^n \times \Omega^n \rightarrow \mathbb{R}^+ : d_{IG}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n w_i \delta(a_i, b_i) \quad (6)$$

with δ like in Eqs. 3 and 4.

4 A Kernel for Natural Language settings

In this section, we will describe the kernel that we have constructed from the IGW metric d_{IG} from the previous section. A kernel will be a function $K : \Omega^n \times \Omega^n \rightarrow \mathbb{R}$ and the mapping to the feature space F will be of the form $\Phi : \Omega^n \rightarrow \mathbb{R}^{\Omega^n}$. We will begin with the simple, one-dimensional, unweighted case (Section 4.1). Next, we will extend it to n dimensions (Section 4.2) and add IGW (Section 4.3) and finally in Section 4.4 the Polynomial Information Gain (PIG) kernel will be introduced.

4.1 A Simple Overlap Kernel (SOK) in one dimension

If we only want to consider vectors in a one-dimensional space then we have to take a closer look at Equations 3 and 4. To derive a kernel from this distance function we will use M2 from Section 2.1. Consider the inner product space $\mathbb{R}^\Omega$ of functions $f : \Omega \rightarrow \mathbb{R}$, the inner product of two functions f and g is defined as:

$$\langle f, g \rangle = \sum_{x \in \Omega} f(x) \cdot g(x) \quad (7)$$

Next, let $\phi_{SOK} : \Omega \rightarrow \mathbb{R}^\Omega$ with

$$\phi_{SOK}(a) = \frac{1}{\sqrt{2}} \delta_a \quad (8)$$

be a feature mapping to the feature space F . The function δ_a is defined as $\delta_a(x) = 1$ when $x = a$ and 0 otherwise (see Equation 4). Starting from the features and the above defined feature mapping we will now work out the inner product to come to the corresponding kernel $k_{SOK} : \Omega \times \Omega \rightarrow \mathbb{R}$:

$$\begin{aligned} k_{SOK}(a, b) &= \langle \phi_{SOK}(a), \phi_{SOK}(b) \rangle \\ &= \langle \frac{1}{\sqrt{2}} \delta_a, \frac{1}{\sqrt{2}} \delta_b \rangle \\ &= \frac{1}{2} \sum_{x \in \Omega} \delta_a(x) \cdot \delta_b(x) \\ &= \frac{1}{2} \delta_a(b) \end{aligned}$$

We don't have to prove that the kernel k_{SOK} satisfies the Mercer conditions because this follows naturally from the definition of the inner product. We started from the features, defined a mapping ϕ_{SOK} on the features and we worked out the inner product between them, see M2 from Section 2.1. However, to show that the kernel really corresponds to the distance function given in Equations 3 and 4 we have to verify the distance formula for kernels given in Section 2.1:

$$\begin{aligned} \delta(a, b) &= \sqrt{k_{SOK}(a, a) - 2k_{SOK}(a, b) + k_{SOK}(b, b)} \\ &= \sqrt{\frac{1}{2}(\delta_a(a) - 2\delta_a(b) + \delta_b(b))} \\ &= \sqrt{\frac{1}{2}(2 - 2\delta_a(b))} \\ &= 1 - \delta_a(b) \end{aligned}$$

The last step is justified by the fact that $1 - \delta_a(b)$ is either 0 or 1 [Ramon, 2002].

4.2 Extending the SOK to n Dimensions

In this section we consider vectors $\mathbf{a}, \mathbf{b} \in \Omega^n$, the distance between these vectors is measured by the distance d_{SOM} from Equation 2. Now consider the inner product space $\mathbb{R}^{\Omega^n}$ (the space of functions $f : \Omega^n \rightarrow \mathbb{R}$) comparable to the one in Equation 7, with the following inner product:

$$\langle f, g \rangle = \sum_{\mathbf{x} \in \Omega^n} f(\mathbf{x}) \cdot g(\mathbf{x}) \quad (9)$$

We begin by defining a mapping $\Phi_{SOK} : \Omega^n \rightarrow \mathbb{R}^{\Omega^n}$:

$$\Phi_{SOK}(\mathbf{a}) = \frac{1}{\sqrt{2}} \delta_{\mathbf{a}}$$

We will again use M2 from Section 2.1 to derive a kernel for d_{SOM} . We work out the inner product between the feature vectors $\Phi_{SOK}(\mathbf{a})$ and $\Phi_{SOK}(\mathbf{b})$ to come to the corresponding kernel $K_{SOK} : \Omega^n \times \Omega^n \rightarrow \mathbb{R}$:

$$K_{SOK}(\mathbf{a}, \mathbf{b}) = \langle \Phi_{SOK}(\mathbf{a}), \Phi_{SOK}(\mathbf{b}) \rangle$$

$$= \frac{1}{2} \sum_{\mathbf{x} \in \Omega^n} \delta_{\mathbf{a}}(\mathbf{x}) \cdot \delta_{\mathbf{b}}(\mathbf{x})$$

$$= \frac{1}{2} \delta_{\mathbf{a}}(\mathbf{b})$$

$$= \frac{1}{2} \sum_{i=1}^n \delta_{a_i}(b_i)$$

Here the function $\delta_{\mathbf{a}}$ is defined as $\delta_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^n \delta_{a_i}(x_i)$ with $\delta_{a_i}(x_i)$ like in Equations 4 and 8. Notice that this corresponds to the inner product between the vectors $\tilde{\mathbf{a}} = (\phi_{SOK}(a_1), \dots, \phi_{SOK}(a_n))$ and $\tilde{\mathbf{b}} = (\phi_{SOK}(b_1), \dots, \phi_{SOK}(b_n))$. So, alternatively we can write $K_{SOK} : \Omega^n \times \Omega^n \rightarrow \mathbb{R}$:

$$K_{SOK}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n k_{SOK}(a_i, b_i) \quad (10)$$

We don't have to prove that the kernel K_{SOK} is a valid kernel as this follows naturally from the definition of the inner product. However, we again have to show that it is a kernel that corresponds to the distance function d_{SOM} from Equation 2. In the following we will show that the kernel K_{SOK} in fact corresponds to the square root of d_{SOM} , we will call this distance function d_r :

$$d_r(\mathbf{a}, \mathbf{b}) = \sqrt{K_{SOK}(\mathbf{a}, \mathbf{a}) - 2K_{SOK}(\mathbf{a}, \mathbf{b}) + K_{SOK}(\mathbf{b}, \mathbf{b})}$$

$$= \sqrt{(1 - \delta_{a_1}(b_1)) + \dots + (1 - \delta_{a_n}(b_n))}$$

$$= \sqrt{\sum_{i=1}^n \delta(a_i, b_i)}$$

$$= \sqrt{d_{SOM}(\mathbf{a}, \mathbf{b})}$$

However, this does not impose any problems for the kernel we are aiming to develop.

4.3 Adding Information Gain to the SOK

Next, we will introduce IGW into the kernel. We will construct a kernel based on the distance function d_{IG} from Equation 6. We have to define new mappings ϕ and Φ to cope with the weights:

$$\Phi_{IG} : \Omega^n \rightarrow \mathbb{R}^{\Omega^n} : \Phi_{IG}(\mathbf{a}) = \frac{\mathbf{w}}{\sqrt{2}} \delta_{\mathbf{a}} \quad (11)$$

$$\phi_{IG} : \Omega \rightarrow \mathbb{R}^{\Omega} : \phi_{IG}(a_i) = \frac{w_i}{\sqrt{2}} \delta_{a_i} \quad (12)$$

We will start again by working out the inner product for the 1-dimensional case $k_{IG} : \Omega \times \Omega \rightarrow \mathbb{R}$:

$$k_{IG}(a_i, b_i) = \langle \phi_{IG}(a_i), \phi_{IG}(b_i) \rangle$$

$$= \langle \frac{w_i}{\sqrt{2}} \delta_{a_i}, \frac{w_i}{\sqrt{2}} \delta_{b_i} \rangle$$

$$= \frac{w_i^2}{2} \sum_{x \in \Omega} \delta_{a_i}(x) \delta_{b_i}(x)$$

$$= \frac{w_i^2}{2} \delta_{a_i}(b_i)$$

The function k_{IG} is guaranteed to be a valid kernel, this follows naturally from the definition of the inner product (see M2 from Section 2.1). In the same way as for K_{SOK} we can show that K_{IG} corresponds to the inner product between the vectors $\tilde{\mathbf{a}} = (\phi_{IG}(a_1), \dots, \phi_{IG}(a_n))$ and $\tilde{\mathbf{b}} = (\phi_{IG}(b_1), \dots, \phi_{IG}(b_n))$, yielding the following result for the kernel $K_{IG} : \Omega^n \times \Omega^n \rightarrow \mathbb{R}$:

$$K_{IG}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n k_{IG}(a_i, b_i)$$

Also, in the same way as we did before we can show for k_{IG} that it corresponds to a distance function $\delta_{IG} : \Omega \times \Omega \rightarrow \mathbb{R}$ with $\delta_{IG}(a_i, b_i) = w_i \delta(a_i, b_i)$ with δ as defined in Equation 3. Furthermore, in the same way as we did for K_{SOK} in Section 4.2 we can show that the kernel K_{IG} corresponds to the square root of the distance function d_{IG} from Equation 6.

4.4 A Polynomial Information Gain Kernel

In this section, we will increase the dimensionality of the feature space F by making use of a polynomial kernel K_{poly} . We will start this section by giving an example [Cristianini and Shawe-Taylor, 2000]:

Example 2 (Polynomial kernel): Consider a two dimensional input space X , with vectors $\mathbf{x}, \mathbf{y} \in X$:

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 = \left(\sum_{i=1}^2 x_i y_i \right)^2$$

$$= \sum_{(i,j)=(1,1)}^{(2,2)} (x_i x_j) (y_i y_j)$$

$$= K_{poly}(\mathbf{x}, \mathbf{y})$$

which is equivalent to an inner product between feature vectors of the form

$$\Phi_{poly}(\mathbf{x}) = (x_1^2, x_2^2, 2x_1 x_2)$$

We call the kernel K_{poly} a polynomial kernel because it maps the vectors from the input space X to the feature space F of all monomials of degree 2.

Next, we will derive the *Polynomial Information Gain (PIG)* kernel K_{PIG} , making use of M1 from Section 2.1. We will do this based on the closure property mentioned in Section 2.1. Consider the feature mapping Φ_{IG} from Equation 11, the kernel K_{PIG} will be defined by a combination of this feature mapping and the polynomial kernel K_{poly} from the above example.

$$K_{PIG}(\mathbf{a}, \mathbf{b}) = K_{poly}(\Phi_{IG}(\mathbf{a}), \Phi_{IG}(\mathbf{b})) \quad (13)$$

with

$$\Phi_{IG} : \Omega^n \rightarrow \mathbb{R}^{\Omega^n}$$

as defined in Equation 11 and

$$K_{poly} : \mathbb{R}^{\Omega^n} \times \mathbb{R}^{\Omega^n} \rightarrow \mathbb{R}$$

From the closure properties of kernels, it follows naturally that K_{PIG} indeed is a valid kernel which calculates the inner product of two vectors transformed by a feature mapping $\Phi_{PIG} : \Omega^n \rightarrow \mathbb{R}^{\Omega^m}$.

The mapping Φ_{PIG} maps vectors $\mathbf{a} \in \Omega^n$ to the space $\mathbb{R}^{\Omega^m}$ with $m > n$. The features of the vectors in $\mathbb{R}^{\Omega^m}$ are combinations (inner products) of the following form: $\langle \phi_{IG}(a_i), \phi_{IG}(a_i) \rangle$ and $2 \langle \phi_{IG}(a_i), \phi_{IG}(a_j) \rangle$. In this way we aim to capture more relevant information that might be present in such combinations. Notice that the inner product here above is the one defined in Equation 7.

5 Experimental Results

In the following section, we will give some experimental results we obtained with the kernels K_{IG} and K_{PIG} . First we will describe the PP attachment disambiguation problem in more detail, followed by the experimental set up (Section 5.2). Finally, we will present the results and compare them to other methods that have been applied on the same data set (Section 5.3).

5.1 PP attachment disambiguation problem

If it is uncertain in a sentence whether the preposition attaches to the verb or to the noun then we have a prepositional phrase (PP) attachment problem. For example, in sentence 1 of Example 1, **with** modifies the noun **shirt** because **with pockets** (PP) describes the **shirt**. In contrast, in sentence 2, **with** modifies the verb **washed** because **with soap** (PP) describes how the shirt is **washed** [Ratnaparkhi, 1998].

Example 3 For the purpose of PP attachment disambiguation, sentences 1 and 2 from Example 1 will be reduced to vectors $\mathbf{a}$ and $\mathbf{b} \in \Omega^4$ as follows:

1. $\mathbf{a} = (\text{bought}, \text{shirt}, \text{with}, \text{pockets})$
2. $\mathbf{b} = (\text{washed}, \text{shirt}, \text{with}, \text{soap})$

In fact, we only keep those words that are of any importance to the PP attachment problem, i.e. $(V(\text{erb}), N(\text{oun}), P(\text{reposition}), N(\text{oun}))$.

In the case where sentences are reduced to quadruples as illustrated in Example 3, the human performance is approximately 88.2% [Ratnaparkhi *et al.*, 1994]. This performance rate gives us an acceptable upper limit for the maximum performance of a computer because it seems unreasonable to expect an algorithm to perform much better than a human. As we will show in our experimental results the kernel K_{IG} achieves a classification accuracy up to 82.9%, see Section 5.3. However, in [Zavrel *et al.*, 1997] the IB1-IG attains a maximum classification accuracy of 84.1%, this is a good indication of the classification accuracy that should be possible to obtain with a kernel based on the distance defined in Equation 6.

5.2 Experimental Setup

The experiments have been done with LIBSVM, a C/C++ and Java library for SVMs [Chih-Chung and Chi-Jen, 2002]. The machine we have used is a Pentium III with 256MB RAM memory, running Windows XP. We choose to implement the kernels K_{IG} and K_{PIG} in Java.

The type of SVM learning we have used is C-SVM [Chih-Chung and Chi-Jen, 2002]. The parameter C controls the model complexity versus the model accuracy and has to be chosen based on the complexity of the problem.

The experiments in this section are conducted on a simplified version of the full PP attachment problem (see Example 3). The data consists of four-tuples of words, extracted from the Wall Street Journal Treebank [Marcus *et al.*, 1993] by a group at IBM [Ratnaparkhi *et al.*, 1994]¹.

The data set contains 20801 training patterns, 3097 test patterns and an independent validation set of 4093 patterns for parameter optimization. All of the models described below were trained on all training examples and the results are given for the 3097 test patterns. For the benchmark comparison with other models from the literature, we use only results for which all parameters have been optimized on the validation set. For more details concerning the data set we refer to [Zavrel *et al.*, 1997].

5.3 Results

The table below gives the results obtained with the kernels K_{IG} and K_{PIG} and compares them to other methods taken from the literature:

Method	Percent correct
K_{IG}	82.9%
K_{PIG}	84.8%
IB1	83.7%
IB1-IG	84.1%
C4.5	79.7%
Maximum Entropy	77.7%
Transformations	81.9%
Back-off model	84.1%
Late Closure	59.0%
Most Likely for each P	72.0%

The results for K_{IG} and K_{PIG} were obtained after optimization of the parameter C on the validation set. For K_{IG} this optimal value was $C = 0.8$ and for K_{PIG} this optimal value was $C = 0.55$. The scores of IB1, IB1-IG, C4.5, Late Closure and Most Likely for each P are taken from [Zavrel *et al.*, 1997], the scores from Maximum Entropy are taken from [Ratnaparkhi *et al.*, 1994] and the scores of Transformations are taken from [Collins and Brooks, 1995].

The fact that the kernel K_{IG} performs worse than IB1-IG, although it is equipped with the very same distance metric, may seem somehow surprising. We believe this is because SVMs perform linear separation in the feature space F . The decision boundary of IB1-IG on the other hand is non-linear. Due to the linearity of the decision boundary of the SVM, some points get misclassified. The number of misclassifications is controlled by the parameter C . Choosing a larger

¹This data set is freely available by ftp from <ftp://ftp.cis.upenn.edu/pub/adwait/PPattachData/>.

value for C forces the SVM to avoid classification errors. However, choosing a value for C that is too high will result in overfitting. Increasing the dimensionality of F makes linear separation easier (Cover's theorem). This can be seen by comparing the classification accuracies between the kernels K_{IG} and K_{PIG} . The results also show that our kernel K_{PIG} outperforms all other methods in the comparison.

6 Conclusion

In this paper, we have proposed a method for designing kernels for SVM learning in natural language settings. As a starting point we took distance functions that have been used in memory-based learning. The resulting kernel K_{PIG} achieves high classification accuracy compared to other methods that have been applied on the same data set. In our approach we started from the vectors of the input space, defined a mapping on those vectors and worked out their inner product in the feature space. All necessary kernel conditions follow naturally from the definition of the inner product. We used the distance formula for kernels to show that the kernels are actually based on the distance functions from Section 3. The experimental results of the kernel K_{PIG} show that increasing the dimensionality in the feature space yields better results. We increased the dimensionality of the feature space by making combinations of the features Φ_{IG} through a polynomial kernel. In this way, we do not only take into account the similarity between corresponding features of vectors, but we also take into account the similarity between non-corresponding features of the vectors. In fact this comes down to taking into account, to a greater extend, the context in which a word occurs.

We used the *resolution of PP attachment ambiguities* as a case-study to validate our findings, but it is our belief that the kernel K_{PIG} can be used for a wide range of natural language problems. In the future we will apply our kernels in more complex natural language settings and compare the results to other methods that have been applied on the same problems. At the moment we are already running experiments on *language independent named-entity recognition* [Sang, 2002]. The first results look promising, but it is too early to draw any significant conclusions. We will also try to further extend the kernels proposed in this paper to achieve even higher accuracies. Moreover, there are still many distance functions reported in the literature that have not yet been investigated to see whether they are applicable in SVM learning, we intend to investigate such distance functions and if possible derive a kernel from them.

References

- [Aha *et al.*, 1991] D. Aha, D. Kibler, and M. Albert. Instance based learning algorithms. *Machine Learning*, 6:37 – 66, 1991.
- [Brown *et al.*, 1999] Michael P.S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, and David Haussler. Support vector machine classification of microarray gene expression data, 1999.
- [Chih-Chung and Chi-Jen, 2002] Chang Chih-Chung and Lin Chi-Jen. Libsvm: a library for support vector machines, 2002.
- [Collins and Brooks, 1995] M.J. Collins and J. Brooks. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora, Cambridge*, 1995.
- [Cost and Salzberg, 1993] Scott Cost and Steven Salzberg. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57 – 78, 1993.
- [Cristianini and Shawe-Taylor, 2000] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.
- [Daelemans *et al.*, 2002] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. Timbl: Tilburg memory-based learner, version 4.3. Technical report, Tilburg University and University of Antwerp, 2002.
- [Marcus *et al.*, 1993] M. Marcus, Santorini B, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313 – 330, 1993.
- [Mitchell, 1997] Tom Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.
- [Ramon, 2002] Jan Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, K.U. Leuven, Belgium, 2002.
- [Ratnaparkhi *et al.*, 1994] Adwait Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Workshop on Human Language Technology, Plainsboro, NJ*, 1994.
- [Ratnaparkhi, 1998] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1998.
- [Sang, 2002] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002, Taipei, Taiwan*, pages 155 – 158, 2002.
- [Stetina and Nagao, 1997] Jiri Stetina and Makoto Nagao. Corpus based pp attachment ambiguity resolution with a semantic dictionary., 1997. Kyoto University, Yoshida Honmachi, Kyoto 606, Japan.
- [Vapnik, 1998] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1998.
- [Veenstra *et al.*, 2000] Jorn Veenstra, Antal van den Bosch, Sabine Buchholz, Walter Daelemans, and Jakub Zavrel. Memory-based word sense disambiguation. *Computers and the Humanities*, 34:1-2:171 – 177, 2000.
- [Zavrel *et al.*, 1997] Jakub Zavrel, Walter Daelemans, and Jorn Veenstra. Resolving pp attachment ambiguities with memory-based learning. In *Proceedings CoNLL, Madrid*, pages 136 – 144. Computational Linguistics, Tilburg University, 1997.

CAUSALITY

A Logic For Causal Reasoning

Alexander Bochman

Computer Science Department,
Holon Academic Institute of Technology, Israel
e-mail: bochmana@hait.ac.il

Abstract

We introduce a logical formalism of irreflexive causal production relations that possesses both a standard monotonic semantics, and a natural non-monotonic semantics. The formalism is shown to provide a complete characterization for the causal reasoning behind causal theories from [McCain and Turner, 1997]. It is shown also that any causal relation is reducible to its Horn sub-relation with respect to the nonmonotonic semantics. We describe also a general correspondence between causal relations and abductive systems, which shows, in effect, that causal relations allow to express abductive reasoning. The results of the study seem to suggest causal production relations as a viable general framework for nonmonotonic reasoning.

1 Introduction

Causal theories have been introduced in [McCain and Turner, 1997] as a nonmonotonic formalism that provides a natural solution for both the frame and ramification problem in reasoning about actions (see [Giunchiglia *et al.*, 2001] for a detailed exposition). A causal theory is a set of causal rules $A \Rightarrow B$ that express a kind of a causal relation among propositions. The nonmonotonic semantics of such theories is determined by causally explained interpretations, namely the interpretations that are both closed with respect to the causal rules and such that every fact holding in them is caused.

The above formalism has been defined semantically, and the main aim of our study consists in laying down its logical foundations. As we will show, such foundations can be built in the framework of an inference system for causal rules that we will call *causal production relations*. The logical origins of the latter are in input/output logics [Makinson and der Torre, 2000], but we will supply them with a natural nonmonotonic semantics allowing to represent significant parts of nonmonotonic reasoning. Thus, the main result of the present study is that causal production relations completely characterize the inference rules for causal conditionals that preserve the nonmonotonic semantics of causal theories. It will be shown also that any causal theory is reducible with respect to this semantics to a *determinate* causal theory that contain only Horn causal rules with literals in heads.

The importance of determinate theories lies in the fact that (modulo some mild finiteness restrictions) the explained interpretations of such a theory are precisely the interpretations of its classical *completion* (see [McCain and Turner, 1997]). Consequently, the nonmonotonic consequences of such theories are obtainable by the use of classical inference tools (such as the Causal Calculator, described in [Giunchiglia *et al.*, 2001]).

Finally, we will describe a relationship between causal relations and abductive reasoning. As we will see, the latter will be representable via a special kind of causal relations called *abductive causal relations*. This will serve as yet another justification for our general claim that causal production relations could be used as a general-purpose formalism for nonmonotonic reasoning, a viable alternative to other non-monotonic formalisms (such as default logic).

2 Causal production relations

We will assume that our language is an ordinary propositional language with the classical connectives and constants $\{\wedge, \vee, \neg, \rightarrow, \mathbf{t}, \mathbf{f}\}$. In addition, $\models$ and Th will denote, respectively, the classical entailment and the associated logical closure operator.

A *causal rule* is a rule of the form $A \Rightarrow B$, where A and B are classical propositions. Informally, such a rule says that, whenever A holds, B is caused. The following definition describes an inference system for causal rules that will be shown to be adequate (and complete) for causal theories. Actually, many of the postulates below (e.g., And and Or) have already been suggested in the literature (see, [Lifschitz, 1997; Schwind, 1999]).

Definition 2.1. A *causal production relation* (or simply a *causal relation*) is a relation $\Rightarrow$ on the set of propositions satisfying the following conditions:

(Strengthening) If $A \models B$ and $B \Rightarrow C$, then $A \Rightarrow C$;

(Weakening) If $A \Rightarrow B$ and $B \models C$, then $A \Rightarrow C$;

(And) If $A \Rightarrow B$ and $A \Rightarrow C$, then $A \Rightarrow B \wedge C$;

(Or) If $A \Rightarrow C$ and $B \Rightarrow C$, then $A \vee B \Rightarrow C$;

(Cut) If $A \Rightarrow B$ and $A \wedge B \Rightarrow C$, then $A \Rightarrow C$;

(Truth) $\mathbf{t} \Rightarrow \mathbf{t}$;

(Falsity) $\mathbf{f} \Rightarrow \mathbf{f}$.

Though causal relations satisfy most of the rules for classical entailment, their distinctive feature is that they are ir-reflexive, that is, they do not satisfy the postulate $A \Rightarrow A$. Actually, such relations correspond to a strongest kind of *input-output logics* from [Makinson and der Torre, 2000], basic input-output logics with reusable output, with the only distinction that causal relations satisfy also Falsity. The latter restricts the universe of discourse to consistent theories¹.

As a consequence of Cut, any causal relation will already be transitive, that is, it will satisfy

(Transitivity) If $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$.

However, transitivity is a weaker property than Cut, since it does not imply the latter (see [Makinson and der Torre, 2000]). In addition, the material implications corresponding to the causal rules can be used as auxiliary assumptions in making derivations. This is a consequence of the following property that holds for causal relations:

(AS) If $A \Rightarrow B$ and $C \wedge (A \Rightarrow B) \Rightarrow D$, then $C \Rightarrow D$.

Remark. The notion of a causal production sanctioned by the above postulates is *atemporal*. For example, the rule $p \wedge q \Rightarrow \neg q$ cannot be understood as saying that p and q jointly cause $\neg q$ (afterwards) in a temporal sense; instead, by Cut and Falsity it implies $p \wedge q \Rightarrow \mathbf{f}$, which means, in effect, that $p \wedge q$ cannot hold. Speaking generally, ‘causally consistent’ propositions cannot cause an effect incompatible with them. A representation of temporal and action domains in this formalism can be achieved, however, by adding explicit temporal arguments to propositions, just as in the classical logic (see [Giunchiglia *et al.*, 2001]).

A *constraint* is a causal rule of the form $A \Rightarrow \mathbf{f}$. Such constraints correspond to state constraints in action theories. Now, any causal rule implies the corresponding constraint:

(Reduction) If $A \Rightarrow B$, then $A \wedge \neg B \Rightarrow \mathbf{f}$.

Note, however, that the reverse entailment does not hold. Actually, given the rest of the postulates, Reduction can replace the Cut postulate:

Lemma 2.1. *Cut is equivalent to Reduction.*

Another important fact about causal relations is the following ‘decomposition’ of causal rules:

Lemma 2.2. *Any causal rule $A \Rightarrow B$ is equivalent to a pair of rules $A \wedge \neg B \Rightarrow \mathbf{f}$ and $A \wedge B \Rightarrow B$.*

Causal rules of the form $A \wedge B \Rightarrow B$ are ‘classically trivial’, but they play an important explanatory role in non-reflexive causal reasoning. Namely, they say that, in any causally explained interpretation in which A holds, we can freely accept B , since it is self-explanatory in this context. Accordingly, such rules could be called *conditional abducibles*. We will discuss the role of abducibles later as part of the general correspondence between causal and abductive reasoning².

¹In fact, it has been shown in [Bochman, 2002] that other, weaker input-output logics can also be given a semantics of the kind described in what follows.

²Note also that, under the general correspondence between causal inference and default logic (see [Turner, 1999]), such rules correspond to normal defaults in default logic.

Now the above lemma says that any causal rule can be decomposed into a (non-causal) constraint and a conditional abducible. This decomposition neatly delineates two kinds of information conveyed by causal rules. One is a logical information that constraints the set of admissible models, while the other is an explanatory information describing what propositions are caused (explainable) in such models³.

For a finite set u of propositions, $\bigwedge u$ will denote their conjunction. We will extend causal rules to rules having arbitrary sets of propositions as premises using the following ‘compactness’ recipe: for any set u ,

$$u \Rightarrow A \equiv \bigwedge a \Rightarrow A, \text{ for some finite } a \subseteq u.$$

For a given causal relation $\Rightarrow$, we will denote by $\mathcal{C}(u)$ the set of propositions caused by u , that is

$$\mathcal{C}(u) = \{A \mid u \Rightarrow A\}.$$

The operator $\mathcal{C}$ will play much the same role as the usual derivability operator for consequence relations. It satisfies the following familiar properties:

$$\text{If } u \subseteq v, \text{ then } \mathcal{C}(u) \subseteq \mathcal{C}(v)$$

$$\mathcal{C}(\mathcal{C}(u)) \subseteq \mathcal{C}(u)$$

Still, $\mathcal{C}$ is not inclusive, that is, $u \subseteq \mathcal{C}(u)$ does not always hold. Also, it is not idempotent, that is, $\mathcal{C}(\mathcal{C}(u))$ can be distinct from $\mathcal{C}(u)$.

3 Monotonic semantics of causal inference

We will describe now a monotonic semantics for causal relations. Actually, it will be just a slight modification of the semantics given in [Turner, 1999].

A fully formal semantic interpretation of causal relations could be given in terms of possible worlds frames of the form (i, W) , where i is a propositional interpretation, while W is a set of propositional interpretations that contains i (see [Turner, 1999]). In order to make our descriptions more transparent, however, we will identify such frames with pairs of theories of the form (α, u) , where α is a world (maximal deductively closed set), while u is a deductively closed set included in α . Such pairs will be called *bitheories*. Clearly, any possible worlds frame (i, W) determines a certain bitheory, and vice versa, so all our definitions will admit a purely semantic reformulation.

By a *causal semantics* we will mean a set of bitheories. The corresponding notion of validity for causal rules is given in the next definition.

Definition 3.1. • A causal rule $A \Rightarrow B$ holds in a bitheory (α, u) if either $A \notin \alpha$, or $B \in u$.

- $A \Rightarrow B$ is valid with respect to a causal semantics $\mathcal{B}$ if it holds in all bitheories from $\mathcal{B}$.

We will denote by $\Rightarrow_{\mathcal{B}}$ the set of all causal rules that are valid in a causal semantics $\mathcal{B}$. It can be easily verified that this set is closed with respect to the postulates for causal relations, and hence we have

³Cf. a similar decomposition of causal rules in [Thielscher, 1997].

Lemma 3.1. $\Rightarrow_B$ is a causal production relation.

To prove completeness, for any causal relation $\Rightarrow$ we consider its *canonical causal semantics* defined as the set $\mathcal{B}_\Rightarrow$ of all bitheories of the form $(\alpha, \mathcal{C}(\alpha))$ such that $\mathcal{C}(\alpha) \subseteq \alpha$. Then the following theorem shows that this semantics determines precisely the source causal relation, including the causal rules with arbitrary sets of premises. In other words, the source causal relation is *strongly complete* for this semantics.

Theorem 3.2. For any set of propositions v ,

$v \Rightarrow A$ iff $A \in u$, for any $(\alpha, u) \in \mathcal{B}_\Rightarrow$ such that $v \subseteq \alpha$.

Thus, any causal semantics determines a causal relation and vice versa, any causal relation is generated by some causal semantics. Hence we conclude with the following general completeness result:

Corollary 3.3. A relation on propositions is a causal relation iff it is generated by a causal semantics.

3.1 Possible worlds semantics

Causal relations can also be given a semantic interpretation in terms of standard possible worlds models.

As usual, a *possible worlds model* is a triple $\mathbb{W} = (W, R, V)$, where W is a set of worlds, R is a binary accessibility relation on W , while V is a function assigning each world a propositional interpretation. Intuitively, $\alpha R \beta$ means that α and β are, respectively, an initial state (input) and a possible output state of a causal production based on a given set of causal rules. A possible worlds model is *quasi-reflexive* if its accessibility relation satisfies the condition:

If $\alpha R \beta$, then $\alpha R \alpha$.

The following definition provides the notion of validity for causal rules in such models:

Definition 3.2. $A \Rightarrow B$ will be said to be *valid* in a possible worlds model (W, R) if, for any worlds α, β such that $\alpha R \beta$, if A holds in α , then B holds in β .

Then we have

Theorem 3.4. A relation is causal if and only if it is determined by a quasi-reflexive possible worlds model.

The above semantics immediately suggests a modal translation of causal rules (see [Turner, 1999]). Namely, the validity of $A \Rightarrow B$ in a possible worlds model is equivalent to validity of the formula $A \rightarrow \Box B$, where $\Box$ is a standard modal operator. Consequently, causal rules are representable by such modal formulas.

4 The nonmonotonic semantics

In addition to the monotonic semantics, a causal relation determines also a natural nonmonotonic semantics.

Recall that the canonical semantics of a causal relation $\Rightarrow$ is determined by all worlds α for which $\mathcal{C}(\alpha) \subseteq \alpha$. These are the worlds that are closed with respect to the causal rules of $\Rightarrow$. Still, due to non-reflexivity of $\Rightarrow$, only some of them are worlds in which any proposition is also caused by some causal rule. These are the worlds α that are fixed points of $\mathcal{C}$, that is, the worlds for which $\alpha = \mathcal{C}(\alpha)$. In such worlds any fact is causally explained. It is these worlds that determine the nonmonotonic semantics, defined below.

Definition 4.1. A *causally explained world* for a causal relation $\Rightarrow$ is a world α such that $\alpha = \mathcal{C}(\alpha)$. The *nonmonotonic semantics* of $\Rightarrow$ is the set of all causally explained worlds.

Propositions that hold in all causally explained worlds are considered as the nonmonotonic consequences determined by the causal relation. This semantics is indeed nonmonotonic, since the set of such consequences changes nonmonotonically with changes in the underlying set of causal rules.

Now we will establish a correspondence between the above semantics and the nonmonotonic semantics of causal theories from [McCain and Turner, 1997].

A *causal theory* is an arbitrary set of causal rules. Given a causal theory Δ and a set u of propositions, we define the following set:

$$\Delta(u) = \{A \mid B \Rightarrow A \in \Delta, \text{ for some } B \in u\}$$

According to [McCain and Turner, 1997], a world α is *causally explained* with respect to Δ if it is the only world containing $\Delta(\alpha)$. Or, equivalently, when $\alpha = \text{Th}(\Delta(\alpha))$.

Now, causal relations can also be considered as (rather big) causal theories. Moreover, it can be easily verified that for the latter the above definition is equivalent to the definition of causally explained worlds, given earlier. Note also that for any causal theory Δ there exists a least causal relation that includes Δ . We will denote it by $\Rightarrow_\Delta$. Actually, $\Rightarrow_\Delta$ is the set of all causal rules that can be derived from Δ using the postulates for causal relations.

Let us introduce now the following definition:

Definition 4.2. Two causal theories will be called (non-monotonically) *equivalent* if they determine the same set of causally explained worlds.

As a first part of our general result, the next theorem shows that the postulates of causal inference preserve the above non-monotonic semantics.

Theorem 4.1. $\Rightarrow_\Delta$ is equivalent to Δ .

The above theorem shows, in effect, that the logic of causal relations is adequate for the nonmonotonic reasoning in causal theories.

Let us say that a causal rule is *Horn* one if it has the form $A \Rightarrow l$, where l is a literal or the falsity constant $\mathbf{f}$ ⁴. Our next result will show that any causal theory is equivalent to a Horn theory. But first one more distinction.

Definition 4.3. • A causal theory is *determinate* if it contains only Horn rules. A causal relation will be called *determinate* if it is generated by a determinate causal theory.

- A causal theory Δ will be said to be *locally finite* if, for any propositional atom p , there is only a finite number of causal rules in Δ that contain p in their heads.
- A causal theory will be called *definite* if it is determinate and locally finite.

Clearly, any finite causal theory will be locally finite, though not vice versa. Similarly, any finite determinate theory will be definite. As can be easily verified, a determinate

⁴Thus, any constraint $A \Rightarrow \mathbf{f}$ will be a Horn rule in this sense.

causal theory is definite if and only if, for any literal l , there is no more than a finite number of rules with the head l . Consequently, our definition of definite theories turns out to be equivalent to that given in [McCain and Turner, 1997].

By a *determinate subrelation* of a causal relation $\Rightarrow$ we will mean the least causal relation generated by the set of all Horn rules from $\Rightarrow$. Then we have

Theorem 4.2. *Any causal relation is nonmonotonically equivalent to its determinate subrelation.*

By Theorem 4.1 we can conclude now that any causal theory is equivalent to some determinate theory.

Example. The causal theory

$$\{p \Rightarrow p, \neg p \Rightarrow \neg p, \neg q \Rightarrow \neg q, \mathbf{t} \Rightarrow p \vee q\}$$

has a single causally explained interpretation $\{p, \neg q\}$. The last rule $\mathbf{t} \Rightarrow p \vee q$ is not Horn, but it implies a constraint $\neg(p \vee q) \Rightarrow \mathbf{f}$ (by Reduction), and it can be verified that the determinate causal theory obtained by replacing the rule with this constraint will be equivalent to the original theory.

Actually, there is a simple algorithm of transforming any causal theory Δ into an equivalent determinate theory Δ_d . It consists in two steps:

1. For any causal rule $A \Rightarrow B$ from Δ , a constraint $A \wedge \neg B \Rightarrow \mathbf{f}$ is added to Δ_d ;
2. If $\{A_i \Rightarrow B_i\}$ is a (minimal) set of rules from Δ such that $\bigwedge B_i \models l$, for some literal l , then a rule $\bigwedge A_i \Rightarrow l$ is added to Δ_d .

Then the following result can be proved:

Lemma 4.3. Δ is equivalent to Δ_d .

Unfortunately, the above algorithm is not modular. Moreover, it does not preserve local finiteness: there are locally finite causal theories such that their determinate counterparts are not locally finite. Still, in many simple cases it gives a convenient recipe for transforming an arbitrary causal theory into an equivalent determinate theory.

Now we are going to show the second part of our main result, namely that causal relations constitute a maximal logic suitable for reasoning in causal theories. To begin with, we introduce

Definition 4.4. Causal theories Δ and Γ are *causally equivalent* if $\Rightarrow_\Delta$ coincides with $\Rightarrow_\Gamma$.

Two theories are causally equivalent if each can be obtained from the other using the postulates of causal relations. In other words, they are equivalent in the ‘causal logic’ associated with the latter. For example, a theory $\{A \vee B \Rightarrow C\}$ is causally equivalent to the theory $\{A \Rightarrow C, B \Rightarrow C\}$ (by Strengthening and Or). Speaking generally, any causal theory is causally equivalent to a theory with rules of the form $\bigwedge l_i \Rightarrow \bigvee l_j$, where l_i, l_j are literals.

Now, by Theorem 4.1, we immediately obtain

Corollary 4.4. *Causally equivalent theories are nonmonotonically equivalent.*

The reverse implication in the above corollary does not hold, and a deep reason for this is that the causal equivalence

is a monotonic (logical) notion, and hence, unlike the non-monotonic equivalence, it is preserved under addition of new causal rules. For example, though any causal theory is equivalent to a determinate one, they may give different results if we add some causal rules to them.

Example. (continued) Let us add a rule $\mathbf{t} \Rightarrow \neg p \vee q$ both to the causal theory in the preceding example, and to its determinate counterpart. In the first case we obtain a theory which implies $\mathbf{t} \Rightarrow q$ (by And), and hence it will have two explained interpretations $\{p, q\}$ and $\{\neg p, q\}$. For the reduced determinate theory, however, the addition of this rule will give $\{p, q\}$ as the only explained interpretation.

What we need is a stronger, monotonic counterpart of the notion of equivalence that would be preserved under addition of new causal rules. This immediately suggests the following definition.

Definition 4.5. Two causal theories Δ and Γ will be said to be *strongly equivalent* if, for any set Φ of causal rules, $\Delta \cup \Phi$ is nonmonotonically equivalent to $\Gamma \cup \Phi$.

Strongly equivalent theories are ‘equivalent forever’, that is, they are interchangeable in any larger causal theory without changing the nonmonotonic semantics. Consequently, strong equivalence can be seen as an equivalence with respect to the monotonic logic of causal theories. And the next result shows that this logic is precisely the logic of causal relations.

Theorem 4.5. *Two causal theories are strongly equivalent if and only if they are causally equivalent.*

The above result (and its proof) has the same pattern as the corresponding results about strong equivalence of logic programs and default theories (see [Lifschitz et al., 2001; Turner, 2001]).

The above result implies, in effect, that causal relations are maximal inference relations that are adequate for reasoning with causal theories: any postulate that is not valid for causal relations can be ‘falsified’ by finding a suitable extension of two causal theories that would determine different causally explained interpretations, and hence would produce different nonmonotonic conclusions.

5 Causation versus abduction

Causal inference has numerous connections with other non-monotonic formalisms. Thus, [Turner, 1999] describes the relations with circumscription, autoepistemic and default logic. In this section we will describe a correspondence between causal inference and abductive reasoning. In addition to specific results, this will give us a broader perspective on the representation capabilities of causal relations.

A general *abductive framework* can be defined as a pair $\mathbb{A} = (\text{Cn}, \mathcal{A})$, where Cn is a supraclassical consequence relation, while $\mathcal{A}$ is a distinguished set of *abducibles*. A proposition A is *explainable* in an abductive framework $\mathbb{A}$ if there exists a consistent set of abducibles $a \subseteq \mathcal{A}$ such that $A \in \text{Cn}(a)$.

A general correspondence between abductive frameworks and input-output logics has been established in [Bochman, 2002], based on the identification between the above notion

of explainability and input-output derivability. By this correspondence, abducibles are representable by ‘reflexive’ propositions satisfying input-output rules $A \Rightarrow A$, while abductive frameworks themselves correspond exactly to input-output logics satisfying an additional postulate of Abduction (see below). By these results, input-output logics allow to give a syntax-independent representation of abductive reasoning.

5.1 The abductive subrelation

We will begin with the following definitions:

Definition 5.1. • A proposition A is an *abducible* in a causal relation $\Rightarrow$ if $A \Rightarrow A$;

- A causal relation is *abductive* if it satisfies

(Abduction) If $A \Rightarrow B$, then $A \Rightarrow C \Rightarrow B$, for some abducible C .

Causal inference in abductive causal relations is always mediated by abducibles. Consequently, propositions caused by worlds are caused, in effect, by the abducibles that hold in these worlds:

$$\mathcal{C}(\alpha) = \mathcal{C}(\alpha \cap \mathcal{A}),$$

where $\mathcal{A}$ is the set of abducibles. As we mentioned, abductive causal relations correspond to a certain class of abductive systems. Moreover, it has been shown in [Bochman, 2002] that the nonmonotonic semantics of such causal relations describes, in effect, the *explanatory closure*, or *completion*, in associated abductive frameworks (see [Console *et al.*, 1991]).

It turns out that any causal relation $\Rightarrow$ includes a greatest abductive subrelation. The latter can be defined as follows:

$$A \Rightarrow^a B \equiv (\exists C)(A \Rightarrow C \Rightarrow C \Rightarrow B)$$

Theorem 5.1. *If $\Rightarrow$ is a causal relation, then $\Rightarrow^a$ is the greatest abductive causal relation included in $\Rightarrow$.*

The abductive subrelation of a causal relation preserves many properties of the latter. For example, both have the same constraints and abducibles.

Recall that in causally explained interpretations any proposition is caused by other propositions, the latter are also caused by accepted propositions, and so on. Clearly, if our ‘causal resources’ are limited, such a causal chain should stop somewhere. More exactly, it should reach abducible (self-explanatory) propositions. This indicates that in many cases the nonmonotonic semantics associated with a causal theory should be determined by the corresponding abductive subrelation. Below we will make this claim precise.

Definition 5.2. A causal relation will be called *weakly abductive* if it is nonmonotonically equivalent to its abductive subrelation.

The next definition will give us an important sufficient condition for weak abductivity.

Definition 5.3. A causal relation $\Rightarrow$ is *well-founded* if any infinite sequence $\{A_0, A_1, A_2, \dots\}$ of propositions such that $A_{n+1} \Rightarrow A_n$, for every $n \geq 0$, contains an abducible.

The above definition describes a variant of a standard notion of well-foundedness with respect to the (transitive) causal order. It should be clear that any causal relation in

a finite language should be well-founded. Moreover, let us say that a causal relation is *finitary* if it is generated by some finite set of causal rules. Then we have

Lemma 5.2. *Any finitary causal relation is well-founded.*

Finally, the next result shows that all such causal relations will be weakly abductive.

Theorem 5.3. *Any well-founded causal relation is weakly abductive.*

It turns out that well-foundedness is not the only condition that is sufficient for weak abductivity. Thus, adequately acyclic causal theories (see [McCain and Turner, 1998] for a definition) are not in general well-founded, but they also satisfy this property.

Theorem 5.4. *Any adequately acyclic causal relation is weakly abductive.*

The above results show that in many cases of interest, the nonmonotonic semantics of causal theories can also be computed using abduction. Still, as a general conclusion, we can say that causal inference constitutes a proper generalization of abductive reasoning beyond the situations where facts are explainable by reference to a fixed set of self-explanatory abducibles.

6 Conclusions and perspectives

Summing up the main results, we can argue that causal relations constitute a powerful formalism for nonmonotonic reasoning, especially suitable for representing action domains. It has been shown, in particular, that it gives an exact description of the logic underlying causal theories. We have seen also that it allows to give a syntax-independent representation of abductive reasoning. If we add to this also the natural correspondences with other nonmonotonic formalisms, such as default logic and circumscription (see [Turner, 1999]), we can safely conclude that causal inference covers a significant part of nonmonotonic reasoning.

Viewed from this perspective, there is still much to be done in order to realize the opportunities created by the formalism. There are two kinds of problems that need to be resolved in this respect.

The nonmonotonic semantics for causal theories is based on the principle of universal causation which is obviously very strong. The principle implies, for example, that if we have no causal rules for a certain proposition, it should be false in all explainable interpretations. As a result, adding a new propositional atom to the language makes any causal theory inconsistent, since we have no causal rules for it. This makes causal theories extremely sensitive to the underlying language in which they are formulated. One way out has been suggested in [Lifschitz, 1997]; it amounts to restricting the principle of universal causation to a particular subset of *explainable* propositions. This approach, however, is purely syntactical and hence retains language dependence.

More subtle, yet perceptible difficulties arise also in representing indeterminate situations in causal theories. Thus, since any causal theory is reducible to a determinate theory, causal rules with disjunctive heads are ignored in the nonmonotonic semantics; more exactly, they are informative only

to the extent that they imply corresponding non-causal constraints or Horn rules. This does not mean, however, that we cannot represent indeterminate information in causal theories. Actually, one of the main contributions of [McCain and Turner, 1997] consisted in showing how we can do this quite naturally in common cases (see also [Lin, 1996]). Still, there is yet no systematic understanding whether and how an indeterminate information can be represented by Horn causal rules.

A more general problem concerns the role and place of causal inference in general nonmonotonic reasoning. Though the former covers many areas of nonmonotonic reasoning, it does not cover them all. Thus, it does not seem suitable for solving the qualification problem in representing actions. Speaking generally, causal reasoning appears to be independent of the kind of nonmonotonicity described by preferential inference from [Kraus *et al.*, 1990]. This naturally suggests that the two kinds of nonmonotonic reasoning with conditionals could be combined into a single formalism, a grand uniform theory of nonmonotonic reasoning. Actually, this idea is not new; it has been explored more than ten years ago in [Geffner, 1992]. It remains to be seen whether current studies of causal reasoning can contribute to viability of such a general theory.

Acknowledgments. The author is grateful to Vladimir Lifschitz, Hudson Turner, David Makinson and Leendert van der Torre, as well as to the anonymous referees, for their instructive comments and suggestions.

References

- [Bochman, 2002] A. Bochman. Input-output logics: Production inference, nonmonotonicity, and abduction, 2002. (Draft).
- [Console *et al.*, 1991] L. Console, D. Theseider Dupre, and P. Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1:661–690, 1991.
- [Geffner, 1992] H. Geffner. *Default Reasoning. Causal and Conditional Theories*. MIT Press, 1992.
- [Giunchiglia *et al.*, 2001] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories, 2001. *Artificial Intelligence* (to appear).
- [Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
- [Lifschitz *et al.*, 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
- [Lifschitz, 1997] V. Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.
- [Lin, 1996] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings AAAI-96*, pages 670–676, 1996.
- [Makinson and der Torre, 2000] D. Makinson and L. Van der Torre. Input/Output logics. *Journal of Philosophical Logic*, 29:383–408, 2000.
- [McCain and Turner, 1997] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings AAAI-97*, pages 460–465, 1997.
- [McCain and Turner, 1998] N. McCain and H. Turner. Satisfiability planning with causal theories. In *Proceedings KR-98*, Trento, 1998.
- [Schwind, 1999] C. Schwind. Causality in action theories. *Linköping Electronic Articles in Computer and Information Science*, 4(4), 1999.
- [Thielscher, 1997] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89:317–364, 1997.
- [Turner, 1999] H. Turner. A logic of universal causation. *Artificial Intelligence*, 113:87–123, 1999.
- [Turner, 2001] H. Turner. Strong equivalence for logic programs and default theories (made easy). In T. Eiter, W. Faber, and M. Truszczyński, editors, *Proceedings LP-NMR’01*, pages 81–92, Vienna, 2001. Springer. LNAI 2173.

Responsibility and Blame: A Structural-Model Approach

Hana Chockler

School of Engineering and Computer Science
Hebrew University
Jerusalem 91904, Israel.
Email: hanac@cs.huji.ac.il

Joseph Y. Halpern*

Department of Computer Science
Cornell University
Ithaca, NY 14853, U.S.A.
Email: halpern@cs.cornell.edu

Abstract

Causality is typically treated an all-or-nothing concept; either A is a cause of B or it is not. We extend the definition of causality introduced by Halpern and Pearl 2001a to take into account the *degree of responsibility* of A for B . For example, if someone wins an election 11–0, then each person who votes for him is less responsible for the victory than if he had won 6–5. We then define a notion of *degree of blame*, which takes into account an agent’s epistemic state. Roughly speaking, the degree of blame of A for B is the expected degree of responsibility of A for B , taken over the epistemic state of an agent.

1 Introduction

There have been many attempts to define *causality* going back to Hume 1739, and continuing to the present (see, for example, [Collins *et al.*, 2003; Pearl, 2000] for some recent work). While many definitions of causality have been proposed, all of them treat causality is treated as an all-or-nothing concept. That is, A is either a cause of B or it is not. As a consequence, thinking only in terms of causality does not at times allow us to make distinctions that we may want to make. For example, suppose that Mr. B wins an election against Mr. G by a vote of 11–0. Each of the people who voted for Mr. B is a cause of him winning. However, it seems that their degree of responsibility should not be as great as in the case when Mr. B wins 6–5.

In this paper, we present a definition of responsibility that takes this distinction into account. The definition is an extension of a definition of causality introduced by Halpern and Pearl 2001a. Like many other definitions of causality going back to Hume 1739, this definition is based on counterfactual dependence. Roughly speaking, A is a cause of B if, had A not happened (this is the counterfactual condition, since A did in fact happen) then B would not have happened. As is well known, this naive definition does not capture all the subtleties involved with causality. In the case of the 6–5 vote, it

is clear that, according to this definition, each of the voters for Mr. B is a cause of him winning, since if they had voted against Mr. B, he would have lost. On the other hand, in the case of the 11–0 vote, there are no causes according to the naive counterfactual definition. A change of one vote does not make no difference. Indeed, in this case, we do say in natural language that the cause is somewhat “diffuse”.

While in this case the standard counterfactual definition may not seem quite so problematic, the following example (taken from [Hall, 2003]) shows that things can be even more subtle. Suppose that Suzy and Billy both pick up rocks and throw them at a bottle. Suzy’s rock gets there first, shattering the bottle. Since both throws are perfectly accurate, Billy’s would have shattered the bottle had Suzy not thrown. Thus, according to the naive counterfactual definition, Suzy’s throw is not a cause of the bottle shattering. This certainly seems counter to intuition.

Both problems are dealt with the same way in [Halpern and Pearl, 2001a]. Roughly speaking, the idea is that A is a cause of B if B counterfactually depends on C under some contingency. For example, voter 1 is a cause of Mr. B winning even if the vote is 11–0 because, under the contingency that 5 of the other voters had voted for Mr. G instead, voter 1’s vote would have become critical; if he had then changed his vote, Mr. B would not have won. Similarly, Suzy’s throw is the cause of the bottle shattering because the bottle shattering counterfactually depends on Suzy’s throw, under the contingency that Billy doesn’t throw. (There are further subtleties in the definition that guarantee that, if things are modeled appropriately, Billy’s throw is not a cause. These are discussed in Section 2.)

It is precisely this consideration of contingencies that lets us define degree of responsibility. We take the degree of responsibility of A for B to be $1/(N + 1)$, where N is the minimal number of changes that have to be made to obtain a contingency where B counterfactually depends on A . (If A is not a cause of B , then the degree of responsibility is 0.) In particular, this means that in the case of the 11–0 vote, the degree of responsibility of any voter for the victory is $1/6$, since 5 changes have to be made before a vote is critical. If the vote were 1001–0, the degree of responsibility of any voter would be $1/501$. On the other hand, if the vote is 5–4, then the degree of responsibility of each voter for Mr. B for Mr. B’s victory is 1; each voter is critical. As we would expect, those

*Supported in part by NSF under grant CTC-0208535 and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by ONR under grant N00014-01-1-0795.

voters who voted for Mr. G have degree of responsibility 0 for Mr. B's victory, since they are not causes of the victory. Finally, in the case of Suzy and Billy, even though Suzy is the only cause of the bottle shattering, Suzy's degree of responsibility is $1/2$, while Billy's is 0. Thus, the degree of responsibility measures to some extent whether or not there are other potential causes.

When determining responsibility, it is assumed that everything relevant about the facts of the world and how the world works (which we characterize in terms of what are called *structural equations*) is known. For example, when saying that voter 1 has degree of responsibility $1/6$ for Mr. B's win when the vote is 11–0, we assume that the vote and the procedure for determining a winner (majority wins) is known. There is no uncertainty about this. Just as with causality, there is no difficulty in talking about the probability that someone has a certain degree of responsibility by putting a probability distribution on the way the world could be and how it works. But this misses out on important component of determining what we call here *blame*: the epistemic state. Consider a doctor who treats a patient with a particular drug resulting in the patient's death. The doctor's treatment is a cause of the patient's death; indeed, the doctor may well bear degree of responsibility 1 for the death. However, if the doctor had no idea that the treatment had adverse side effects for people with high blood pressure, he should perhaps not be held to blame for the death. Actually, in legal arguments, it may not be so relevant what the doctor actually did or did not know, but what he *should have known*. Thus, rather than considering the doctor's actual epistemic state, it may be more important to consider what his epistemic state should have been. But, in any case, if we are trying to determine whether the doctor is to blame for the patient's death, we must take into account the doctor's epistemic state.

We present a definition of blame that considers whether agent a performing action b is to blame for an outcome φ . The definition is relative to an epistemic state for a , which is taken, roughly speaking, to be a set of situations before action b is performed, together with a probability on them. The degree of blame is then essentially the expected degree of responsibility of action b for φ (except that we ignore situations where φ was already true or b was already performed). To understand the difference between responsibility and blame, suppose that there is a firing squad consisting of ten excellent marksmen. Only one of them has live bullets in his rifle; the rest have blanks. The marksmen do not know which of them has the live bullets. The marksmen shoot at the prisoner and he dies. The only marksman that is the cause of the prisoner's death is the one with the live bullets. That marksman has degree of responsibility 1 for the death; all the rest have degree of responsibility 0. However, each of the marksmen has degree of blame $1/10$.¹

While we believe that our definitions of responsibility and blame are reasonable, they certainly do not capture all the connotations of these words as used in the literature. In the philosophy literature, papers on responsibility typically are concerned with *moral responsibility* (see, for example, [Zim-

merman, 1988]). Our definitions, by design, do not take into account intentions or possible alternative actions, both of which seem necessary in dealing with moral issues. For example, there is no question that Truman was in part responsible and to blame for the deaths resulting from dropping the atom bombs on Hiroshima and Nagasaki. However, to decide whether this is a morally reprehensible act, it is also necessary to consider the alternative actions he could have performed, and their possible outcomes. While our definitions do not directly address these moral issues, we believe that they may be helpful in elucidating them.

The rest of this paper is organized as follows. In Section 2 we review the basic definitions of causal models based on structural equations, which are the basis for our definitions of responsibility and blame. In Section 3, we review the definition of causality from [Halpern and Pearl, 2001a], and show how it can be modified to give a definition of responsibility. In Section 3.3, we give our definition of blame. In Section 4, we discuss the complexity of computing responsibility and blame. Proofs of the theorems can be found in the full paper, available at <http://www.cs.cornell.edu/home/halpern/papers/blame.ps>.

2 Causal Models: A Review

In this section, we review the details of the definitions of causal models from [Halpern and Pearl, 2001a].

A *signature* is a tuple $S = \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle$, where $\mathcal{U}$ is a finite set of *exogenous* variables, $\mathcal{V}$ is a set of *endogenous* variables, and the function $\mathcal{R} : \mathcal{U} \cup \mathcal{V} \rightarrow \mathcal{D}$ associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a nonempty set $\mathcal{R}(Y)$ of possible values for Y from the range $\mathcal{D}$. Intuitively, the exogenous variables are ones whose values are determined by factors outside the model, while the endogenous variables are ones whose values are ultimately determined by the exogenous variables. A *causal model* over signature S is a tuple $M = \langle S, \mathcal{F} \rangle$, where $\mathcal{F}$ associates with every endogenous variable $X \in \mathcal{V}$ a function F_X such that $F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U) \times (\times_{Y \in \mathcal{V} \setminus \{X\}} \mathcal{R}(Y))) \rightarrow \mathcal{R}(X)$. That is, F_X describes how the value of the endogenous variable X is determined by the values of all other variables in $\mathcal{U} \cup \mathcal{V}$. If the range $\mathcal{D}$ contains only two values, we say that M is a *binary causal model*.

We can describe (some salient features of) a causal model M using a *causal network*. This is a graph with nodes corresponding to the random variables in $\mathcal{V}$ and an edge from a node labeled X to one labeled Y if F_Y depends on the value of X . Intuitively, variables can have a causal effect only on their descendants in the causal network; if Y is not a descendant of X , then a change in the value of X has no affect on the value of Y . For ease of exposition, we restrict attention to what are called *recursive* models. These are ones whose associated causal network is a directed acyclic graph (that is, a graph that has no cycle of edges). Actually, it suffices for our purposes that, for each setting $\vec{u}$ for the variables in $\mathcal{U}$, there is no cycle among the edges of causal network. We call a setting $\vec{u}$ for the variables in $\mathcal{U}$ a *context*. It should be clear that if M is a recursive causal model, then there is always a unique solution to the equations in M , given a *context*.

¹We thank Tim Williamson for this example.

The equations determined by $\{F_X : X \in \mathcal{V}\}$ can be thought of as representing processes (or mechanisms) by which values are assigned to variables. For example, if $F_X(Y, Z, U) = Y + U$ (which we usually write as $X = Y + U$), then if $Y = 3$ and $U = 2$, then $X = 5$, regardless of how Z is set. This equation also gives counterfactual information. It says that, in the context $U = 4$, if Y were 4, then X would be $u + 4$, regardless of what value X , Y , and Z actually take in the real world.

While the equations for a given problem are typically obvious, the choice of variables may not be. For example, consider the rock-throwing example from the introduction. In this case, a naive model might have an exogenous variable U that encapsulates whatever background factors cause Suzy and Billy to decide to throw the rock (the details of U do not matter, since we are interested only in the context where U 's value is such that both Suzy and Billy throw), a variable ST for Suzy throws ($ST = 1$ if Suzy throws, and $ST = 0$ if she doesn't), a variable BT for Billy throws, and a variable BS for bottle shatters. In the naive model, whose graph is given in Figure 1, BS is 1 if one of ST and BT is 1. (Note that the graph omits the exogenous variable U , since it plays no role. In the graph, there is an arrow from variable X to variable Y if the value of Y depends on the value of X .)

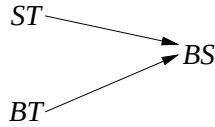


Figure 1: A naive model for the rock-throwing example.

This causal model does not distinguish between Suzy and Billy's rocks hitting the bottle simultaneously and Suzy's rock hitting first. A more sophisticated model might also include variables SH and BH , for Suzy's rock hits the bottle and Billy's rock hits the bottle. Clearly BS is 1 iff one of BH and BT is 1. However, now, SH is 1 if ST is 1, and $BH = 1$ if $BH = 1$ and $SH = 0$. Thus, Billy's throw hits if Billy throws and Suzy's rock doesn't hit. This model is described by the following graph, where we implicitly assume a context where Suzy throws first, so there is an edge from SH to BH , but not one in the other direction.

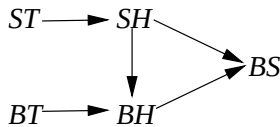


Figure 2: A better model for the rock-throwing example.

Given a causal model $M = (\mathcal{S}, \mathcal{F})$, a (possibly empty) vector $\vec{X}$ of variables in $\mathcal{V}$, and vectors $\vec{x}$ and $\vec{u}$ of values for the variables in $\vec{X}$ and $\mathcal{U}$, respectively, we can define a new causal model denoted $M_{\vec{X} \leftarrow \vec{x}}$ over the signature $\mathcal{S}_{\vec{X}} = (\mathcal{U}, \mathcal{V} - \vec{X}, \mathcal{R}|_{\mathcal{V} - \vec{X}})$. Formally, $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}_{\vec{X}}, \mathcal{F}_{\vec{X} \leftarrow \vec{x}})$, where $F_Y^{\vec{X} \leftarrow \vec{x}}$ is obtained from F_Y by setting the values of

the variables in $\vec{X}$ to $\vec{x}$. Intuitively, this is the causal model that results when the variables in $\vec{X}$ are set to $\vec{x}$ by some external action that affects only the variables in $\vec{X}$; we do not model the action or its causes explicitly. For example, if M is the more sophisticated model for the rock-throwing example, then $M_{ST \leftarrow 0}$ is the model where Suzy doesn't throw.

Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$, is called a *primitive event*. A *basic causal formula* is one of the form $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$, where

- φ is a Boolean combination of primitive events;
- $Y_1, \dots, Y_k$ are distinct variables in $\mathcal{V}$; and
- $y_i \in \mathcal{R}(Y_i)$.

Such a formula is abbreviated as $[\vec{Y} \leftarrow \vec{y}] \varphi$. The special case where $k = 0$ is abbreviated as φ . Intuitively, $[Y_1 \leftarrow y_1, \dots, Y_k \leftarrow y_k] \varphi$ says that φ holds in the counterfactual world that would arise if Y_i is set to y_i , $i = 1, \dots, k$. A *causal formula* is a Boolean combination of basic causal formulas.

A causal formula φ is true or false in a causal model, given a *context*. We write $(M, \vec{u}) \models \varphi$ if φ is true in causal model M given context $\vec{u}$. $(M, \vec{u}) \models [\vec{Y} \leftarrow \vec{y}](X = x)$ if the variable X has value x in the unique (since we are dealing with recursive models) solution to the equations in $M_{\vec{Y} \leftarrow \vec{y}}$ in context $\vec{u}$ (that is, the unique vector of values for the exogenous variables that simultaneously satisfies all equations $F_Z^{\vec{Y} \leftarrow \vec{y}}$, $Z \in \mathcal{V} - \vec{Y}$, with the variables in $\mathcal{U}$ set to $\vec{u}$). We extend the definition to arbitrary causal formulas in the obvious way.

3 Causality and Responsibility

3.1 Causality

We start with the definition of cause from [Halpern and Pearl, 2001a].

Definition 3.1 We say that $\vec{X} = \vec{x}$ is a cause of φ in $(M, \vec{u})$ if the following three conditions hold:

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \varphi$.

AC2. There exist a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x}', \vec{w}')$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}) \models Z = z^*$ for $Z \in \vec{Z}$, then

- $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}'] \neg \varphi$. That is, changing $(\vec{X}, \vec{W})$ from $(\vec{x}, \vec{w})$ to $(\vec{x}', \vec{w}')$ changes φ from true to false.
- $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}', \vec{Z}' \leftarrow \vec{z}^*] \varphi$ for all subsets $\vec{Z}'$ of $\vec{Z}$. That is, setting $\vec{W}$ to $\vec{w}'$ should have no effect on φ as long as $\vec{X}$ has the value $\vec{x}$, even if all the variables in an arbitrary subset of $\vec{Z}$ are set to their original values in the context $\vec{u}$.

AC3. $(\vec{X} = \vec{x})$ is minimal, that is, no subset of $\vec{X}$ satisfies AC2.

AC1 just says that A cannot be a cause of B unless both A and B are true, while AC3 is a minimality condition to prevent, for example, Suzy throwing the rock and sneezing from

being a cause of the bottle shattering. Eiter and Lukasiewicz 2002b showed that one consequence of AC3 is that causes can always be taken to be single conjuncts. Thus, from here on in, we talk about $X = x$ being the cause of φ , rather than $\vec{X} = \vec{x}$ being the cause. The core of this definition lies in AC2. Informally, the variables in $\vec{Z}$ should be thought of as describing the “active causal process” from X to φ . These are the variables that mediate between X and φ . AC2(a) is reminiscent of the traditional counterfactual criterion, according to which $X = x$ is a cause of φ if change the value of X results in φ being false. However, AC2(a) is more permissive than the traditional criterion; it allows the dependence of φ on X to be tested under special *structural contingencies*, in which the variables $\vec{W}$ are held constant at some setting $\vec{w}'$. AC2(b) is an attempt to counteract the “permissiveness” of AC2(a) with regard to structural contingencies. Essentially, it ensures that X alone suffices to bring about the change from φ to $\neg\varphi$; setting $\vec{W}$ to $\vec{w}'$ merely eliminates spurious side effects that tend to mask the action of X .

To understand the role of AC2(b), consider the rock-throwing example again. In the model in Figure 1, it is easy to see that both Suzy and Billy are causes of the bottle shattering. Taking $\vec{Z} = \{ST, BS\}$, consider the structural contingency where Billy doesn’t throw ($BT = 0$). Clearly $[ST \leftarrow 0, BT \leftarrow 0]BS = 0$ and $[ST \leftarrow 1, BT \leftarrow 0]BS = 1$ both hold, so Suzy is a cause of the bottle shattering. A symmetric argument shows that Billy is also the cause.

But now consider the model described in Figure 2. It is still the case that Suzy is a cause in this model. We can take $\vec{Z} = \{ST, SH, BS\}$ and again consider the contingency where Billy doesn’t throw. However, Billy is *not* a cause of the bottle shattering. For suppose that we now take $\vec{Z} = \{BT, BH, BS\}$ and consider the contingency where Suzy doesn’t throw. Clearly AC2(a) holds, since if Billy doesn’t throw (under this contingency), then the bottle doesn’t shatter. However, AC2(b) does not hold. Since $BH \in \vec{Z}$, if we set BH to 0 (it’s original value), then AC2(b) requires that $[BT \leftarrow 1, ST \leftarrow 0, BH \leftarrow 0](BS = 1)$ hold, but it does not. Similar arguments show that no other choice of $(\vec{Z}, \vec{W})$ makes Billy’s throw a cause.

3.2 Responsibility

The definition of responsibility in causal models extends the definition of causality.

Definition 3.2 The degree of responsibility of $X = x$ for φ in $(M, \vec{u})$, denoted $dr((M, \vec{u}), (X = x), \varphi)$, is 0 if $X = x$ is not a cause of φ in $(M, \vec{u})$; it is $1/(k + 1)$ if $X = x$ is a cause of φ in $(M, \vec{u})$ and there exists a partition $(\vec{Z}, \vec{W})$ and setting $(x', \vec{w}')$ for which AC2 holds such that (a) k variables in $\vec{W}$ have different values in $\vec{w}'$ than they do in the context $\vec{u}$ and (b) there is no partition $(\vec{Z}', \vec{W}')$ and setting $(x'', \vec{w}'')$ satisfying AC2 such that only $k' < k$ variables have different values in $\vec{w}''$ than they do in the context $\vec{u}$.

Intuitively, $dr((M, \vec{u}), (X = x), \varphi)$ measures the minimal number of changes that have to be made in $\vec{u}$ in order to make φ counterfactually depend on X . If no partition of $\mathcal{V}$ to

$(\vec{Z}, \vec{W})$ makes φ counterfactually depend on $(X = x)$, then the minimal number of changes in $\vec{u}$ in Definition 3.2 is taken to have cardinality ∞ , and thus the degree of responsibility of $X = x$ is 0. If φ counterfactually depends on $X = x$, that is, changing the value of X alone falsifies φ in $(M, \vec{u})$, then the degree of responsibility of $X = x$ in φ is 1. In other cases the degree of responsibility is strictly between 0 and 1. Note that $X = x$ is a *cause* of φ iff the degree of responsibility of $X = x$ for φ is greater than 0.

Example 3.3 Consider the voting example from the introduction. Suppose there are 11 voters. Voter i is represented by a variable X_i , $i = 1, \dots, 11$; the outcome is represented by the variable O , which is 1 if Mr. B wins and 0 if Mr. B wins. In the context where Mr. B wins 11–0, it is easy to check that each voter is a cause of the victory (that is $X_i = 1$ is a cause of $O = 1$, for $i = 1, \dots, 11$). However, the degree of responsibility of $X_i = 1$ for $O = 1$ is just $1/6$, since at least five other voters must change their votes before changing X_i to 0 results in $O = 0$. But now consider the context where Mr. B wins 6–5. Again, each voter who votes for Mr. B is a cause of him winning. However, now each of these voters have degree of responsibility 1. That is, if $X_i = 1$, changing X_i to 0 is already enough to make $O = 0$; no other variables need to change.

Example 3.4 It is easy to see that Suzy’s throw has degree of responsibility $1/2$ for the bottle shattering both in the naive model described in Figure 1 and the “better” model of Figure 2 in the context where both Suzy and Billy throw. In both cases, we must consider the contingency where Billy does not throw. Although Suzy is the only cause of the bottle shattering in the latter model, her degree of responsibility is still only $1/2$, since the bottle would have shattered even if she hadn’t thrown. This is a subtlety not captured by the notion of causality.

Interestingly, in a companion paper [Chockler *et al.*, 2003] we apply our notion of responsibility to program verification. The idea is to determine the degree of responsibility of the setting of each state for the satisfaction of a specification in a given system. For example, given a specification of the form $\Diamond p$ (eventually p is true), if p is true in only one state of the verified system, then that state has degree of responsibility 1 for the specification. On the other hand, if p is true in three states, each state only has degree of responsibility $1/3$. Experience has shown that if there are many states with low degree of responsibility for a specification, then either the specification is incomplete (perhaps p really did have to happen three times, in which case the specification should have said so), or there is a problem with the system generated by the program, since it has redundant states.

The degree of responsibility can also be used to provide a measure of the degree of fault-tolerance in a system. If a component is critical to an outcome, it will have degree of responsibility 1. To ensure fault tolerance, we need to make sure that no component has a high degree of responsibility for an outcome. Going back to the example of $\Diamond p$, the degree of responsibility of $1/3$ for a state means that the system is robust to the simultaneous failures of at most two states.

3.3 Blame

The definitions of both causality and responsibility assume that the context and the structural equations are given; there is no uncertainty. We are often interested in assigning a degree of *blame* to an action. This assignment depends on the epistemic state of the agent before the action was performed. Intuitively, if the agent had no reason to believe that his action would result in a certain outcome, then he is not to blame for the outcome (even if in fact his action caused the outcome).

To deal with the fact that we are considering two points in time—before the action was performed and after—we add superscripts to variables. We use a superscript 0 to denote the value of the random variable before the action was performed and the superscript 1 to denote the value of the random variable after. Thus, $Y^0 = 1$ denotes that the random variable Y has value 1 before the action is performed, while $Y^1 = 2$ denotes that it had value 1 afterwards. If φ is a Boolean combination of (unscripted) random variables, we use φ^0 and φ^1 to denote the value of φ before and after the action is performed, respectively.

There are two significant sources of uncertainty for an agent who is contemplating performing an action:

- what the true situation is; for example, a doctor may be uncertain about whether a patient has high blood pressure.
- how the world works; for example, a doctor may be uncertain about the side effects of a given medication.

In our framework, the “true situation” is determined by the context and “how the world works” is determined by the structural equations. Thus, we model an agent’s uncertainty by a pair $(\mathcal{K}, \text{Pr})$, where $\mathcal{K}$ is a set of pairs of the form $(M, \vec{u})$, where M is a causal model and $\vec{u}$ is a context, and Pr is a probability distribution over $\mathcal{K}$. Following [Halpern and Pearl, 2001b], who used such epistemic states in the definition of *explanation*, we call a pair $(M, \vec{u})$ a *situation*.

Roughly speaking, the degree of blame that setting X to x has for φ is the expected degree of responsibility of $X = x$ for φ , taken over the situations $(M_{X \leftarrow x, \vec{Y} \leftarrow \vec{y}}, \vec{u})$, where $(M, \vec{u}, \vec{Y} \leftarrow \vec{y}) \in \mathcal{K}$. Our actual definition of blame is just this definition, except that, when computing the expectation, we do not count situations in $\mathcal{K}$ where φ was already true or X was already x . To understand why, suppose that we are trying to compute the degree of blame of Suzy’s throwing the rock for the bottle shattering. Assume that we are interested in a period in a bottle in the period between time 0 and time 1, and the bottle was actually shattered at time 1. We certainly don’t want to say that Suzy’s throw was to blame if Suzy didn’t throw between time 0 and time 1 or if the bottle was already shattered at time 0. So suppose that Suzy does in fact throw between time 0 and time 1, and at time 0, she considers the following four situations to be equally likely:

- $(M_1, \vec{u}_1)$, where the bottle was already shattered before Suzy’s throw;
- $(M_2, \vec{u}_2)$, where the bottle was whole before Suzy’s throw, and Suzy and Billy both hit the bottle simultaneously (as described in the model in Figure 1);

- $(M_3, \vec{u}_3)$, where the bottle was whole before Suzy’s throw, and Suzy’s throw hit before Billy’s throw (as described in the model in Figure 2); and
- $(M_4, \vec{u}_4)$, where the bottle was whole before Suzy’s throw, and Billy did not throw.

To compute the degree of blame assigned to Suzy’s throwing the rock for the bottle shattering, we ignore $(M, \vec{u}_1)$, because the bottle is already shattered in $(M, \vec{u}_1)$ before Suzy’s action. The degree of responsibility of Suzy’s throw for the bottle shattering is $1/2$ in $(M_2, \vec{u}_2)$ and $(M_3, \vec{u}_3)$, and is 1 in $(M_4, \vec{u}_4)$. It is easy to see that the degree of blame is $\frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot 1 = \frac{1}{2}$.

Definition 3.5 The degree of blame of setting X to x for φ relative to epistemic state $(\mathcal{K}, \text{Pr})$, denoted $\text{db}(\mathcal{K}, \text{Pr}, X \leftarrow x, \varphi)$, is $\sum_{\{(M, \vec{u}) \in \mathcal{K}: (M, \vec{u}) \models X^0 \neq x \wedge \neg \varphi^0\}} \text{dr}((M, \vec{u}), X^1 = x, \varphi^1) \text{Pr}((M, \vec{u}))$.

Example 3.6 Consider again the example of the firing squad with ten excellent marksmen. Suppose that marksman 1 knows that exactly one marksman has a live bullet in his rifle. Thus, he considers 10 situations possible, depending on who has the bullet. Let p_i be his prior probability that marksman i has the live bullet. Then the degree of blame of his shot for the death is p_i . The degree of responsibility is either 1 or 0, depending on whether or not he actually had the live bullet. Thus, it is possible for the degree of responsibility to be 1 and the degree of blame to be 0 (if he ascribes probability 0 to his having the live bullet, when in fact he does), and it is possible for the degree of responsibility to be 0 and the degree of blame to be 1 (if he mistakenly ascribes probability 1 to his having the bullet when he in fact does not).

Example 3.7 The previous example suggests that both degree of blame and degree of responsibility may be relevant in a legal setting. Another issue that is relevant in legal settings is whether to consider actual epistemic state or to consider what the epistemic state should have been. The former is relevant when considering intent. To see the relevance of the latter, consider a patient who dies as a result of being treated by a doctor with a particular drug. Assume that the patient died due to the drug’s adverse side effects on people with high blood pressure and, for simplicity, that this was the only cause of death. Suppose that the doctor was not aware of the drug’s adverse side effects. (Formally, this means that he does not consider possible a situation with a causal model where taking the drug causes death.) Then, relative to the doctor’s actual epistemic state, the doctor’s degree of blame will be 0. However, a lawyer might argue in court that the doctor should have known that treatment had adverse side effects for patients with high blood pressure (because this is well documented in the literature) and thus should have checked the patient’s blood pressure. If the doctor had performed this test, he would of course have known that the patient had high blood pressure. With respect to the resulting epistemic state, the doctor’s degree of blame for the death is quite high. Of course, the lawyer’s job is to convince the court that the latter epistemic state is the appropriate one to consider when assigning degree of blame.

4 The Complexity of Computing Responsibility and Blame

In this section we present complexity results for computing the degree of responsibility and blame for general recursive models.

4.1 Complexity of responsibility

Complexity results for computing causality were presented by Eiter and Lukasiewicz 2002a; 2002b. They showed that the problem of detecting whether $X = x$ is an actual cause of φ is Σ_2^P -complete for general recursive models and NP-complete for binary models [Eiter and Lukasiewicz, 2002b]. (Recall that Σ_2^P is the second level of the polynomial hierarchy and that binary models are ones where all random variables can take on exactly two values.) There is a similar gap between the complexity of computing the degree of responsibility and blame in general models and in binary models.

For a complexity class A , $\text{FP}^{A[\log n]}$ consists of all functions that can be computed by a polynomial-time Turing machine with an oracle for a problem in A , which on input x asks a total of $O(\log |x|)$ queries (cf. [Papadimitriou, 1984]). We show that computing the degree of responsibility of $X = x$ for φ in arbitrary models is $\text{FP}^{\Sigma_2^P[\log n]}$ -complete. In [Chockler et al., 2003], we show that computing the degree of responsibility in binary models is $\text{FP}^{\text{NP}[\log n]}$ -complete.

Since there are no known natural $\text{FP}^{\Sigma_2^P[\log n]}$ -complete problems, the first step in showing that computing the degree of responsibility is $\text{FP}^{\Sigma_2^P[\log n]}$ -complete is to define an $\text{FP}^{\Sigma_2^P[\log n]}$ -complete problem. We start by defining one that we call MAXQSAT_2 .

Recall that a *quantified Boolean formula* [Stockmeyer, 1977] (QBF) has the form $\forall X_1 \exists X_2 \dots \psi$, where $X_1, X_2, \dots$ are propositional variables and ψ is a propositional formula. A QBF is *closed* if it has no free propositional variables. TQBF consists of the closed QBF formulas that are true. For example, $\forall X \exists Y (X \Rightarrow Y) \in \text{TQBF}$. As shown by Stockmeyer 1977, the following problem QSAT_2 is Σ_2^P -complete:

$$\text{QSAT}_2 = \{\exists X \forall Y \psi(X, Y) \in \text{TQBF} : \psi \in 3\text{-CNF}\}.$$

That is, QSAT_2 is the language of all true QBFs of the form $\exists X \forall Y \psi$, $\exists \vec{X} \forall \vec{Y} \psi$, where ψ is a Boolean formula in 3-CNF.

A *witness* f for a true closed QBF $\exists \vec{X} \forall \vec{Y} \psi$ is an assignment f to $\vec{X}$ under which $\forall \vec{Y} \psi$ is true. We define MAXQSAT_2 as the problem of computing the maximal number of variables in $\vec{X}$ that can be assigned 1 in a witness for $\exists \vec{X} \forall \vec{Y} \psi$. Formally, given a QBF $\Phi = \exists \vec{X} \forall \vec{Y} \psi$, define $\text{MAXQSAT}_2(\Phi)$ to be k if there exists a witness for Φ that assigns exactly k of the variables in $\vec{X}$ the value 1, and every other witness for Φ assigns at most $k' \leq k$ variables in $\vec{X}$ the value 1. If $\Phi \notin \text{QSAT}_2$, then $\text{MAXQSAT}_2(\Phi) = -1$.

Theorem 4.1 MAXQSAT_2 is $\text{FP}^{\Sigma_2^P[\log n]}$ -complete.

The proof of Theorem 4.1 shows explicitly how to reduce the computation of any $\text{FP}^{\Sigma_2^P[\log n]}$ problem to MAXQSAT_2 . Given a polynomial-time Turing machine M with oracle in

Σ_2^P that on an input of size n makes $O(\log n)$ oracle queries, we construct a formula $\Phi = \exists \vec{X} \forall \vec{Y} \psi$, where ψ is a Boolean formula in 3-CNF such that given $\text{MAXQSAT}_2(\Phi)$ we can compute the output of M in polynomial time. Essentially, the formula ψ describes the output of M for all possible sequences of answers for oracle queries and $\text{MAXQSAT}_2(\Phi)$ gives the correct sequence of answers. Since the total number of oracle queries is $O(\log n)$, the number of all possible sequences of answers is polynomial in the size of the input, and thus the size of Φ is also polynomial in the size of the input. The construction is somewhat complicated; see the full paper for details.

Similarly to MAXQSAT_2 , we define $\text{MINQSAT}_2(\exists \vec{X} \forall \vec{Y} \psi)$ to be the minimum number of variables in $\vec{X}$ that can be assigned 1 in a witness for $\exists \vec{X} \forall \vec{Y} \psi$ if there is such a witness, and $|\vec{X}| + 1$ otherwise. It is easy to see that MINQSAT_2 has the same complexity as MAXQSAT_2 , since

$$\text{MAXQSAT}_2(\exists \vec{X} \forall \vec{Y} \psi) = |\vec{X}| - \text{MINQSAT}_2(\exists \vec{X} \forall \vec{Y} \bar{\psi}),$$

where $\bar{\psi}$ is obtained from ψ by replacing each propositional variable $X \in \vec{X}$ with its negation.

Using a reduction from MINQSAT_2 , we can prove the desired complexity result.

Theorem 4.2 *The degree of responsibility is $\text{FP}^{\Sigma_2^P[\log n]}$ -complete for general recursive causal models.*

Proof: Due to lack of space, we present the proof here in a somewhat abridged form. Membership in $\text{FP}^{\Sigma_2^P[\log n]}$ can be proved using an argument similar to the one used in [Chockler et al., 2003] to prove membership of the degree of responsibility for binary causal models in $\text{FP}^{\text{NP}[\log n]}$.

The proof that computing the degree of responsibility is $\text{FP}^{\Sigma_2^P[\log n]}$ -hard essentially follows from an argument in [Eiter and Lukasiewicz, 2002a] showing that QSAT_2 can be reduced to the problem of detecting causality. In fact, their argument actually provides a reduction from MINQSAT_2 to the degree of responsibility. Given a QBF of the form $\exists \vec{X} \forall \vec{Y} \psi$, Eiter and Lukasiewicz construct a causal model M whose endogenous variables include $\vec{X}$, $\vec{Y}$, and a fresh variable X^* . They consider a context u in which all variables in $\vec{X}$ get the value 0. They show that $\forall \vec{X} \exists \vec{Y} \psi$ is true iff $X^* = 0$ is a cause of ψ in (M, u) . If $X^* = 0$ is indeed a cause, then the set $\vec{W}$ in AC2 must be a subset of $\vec{X}$. That is, if there is a partition $(\vec{W}, \vec{Z})$ and a setting $(x', \vec{w}')$ satisfying AC2 showing that $X^* = 0$ is a cause of ψ , then $\vec{W}$ must be a subset of $\vec{X}$. Moreover, the variables in $\vec{W}$ that change value from 0 to 1 in $\vec{w}'$ are precisely those such that an assignment f that assigns 1 to just those variables makes $\forall \vec{Z} \psi$ true. It then easily follows that $\text{MINQSAT}_2(\exists \vec{X} \forall \vec{Y} \psi) = i < |\vec{X}| + 1$ iff $\text{dr}((M, u), X^* = 0, \psi) = 1/(i + 1)$, and $\text{MINQSAT}_2(\exists \vec{X} \forall \vec{Y} \psi) = |\vec{X}| + 1$ iff $\text{dr}((M, u), X^* = 0, \psi) = 0$. $\square$

4.2 Complexity of blame

Given an epistemic state $(\mathcal{K}, \text{Pr})$, where $\mathcal{K}$ consists of N possible situations, each with at most n random variables, the straightforward way to compute $\text{db}(\mathcal{K}, \text{Pr}, X \leftarrow x, \varphi)$ is by computing $\text{dr}((M, \vec{u}), X^1 = x, \varphi^1)$ for each $(M, \vec{u}) \in \mathcal{K}$ such that $(M, \vec{u}) \models X^0 \neq x \wedge \neg\varphi^0$, and then computing the expected degree of responsibility with respect to these situations, as in Definition 3.5. Recall that the degree of responsibility in each model is determined by using a binary search thus uses $O(\log n)$ queries in each model in $\mathcal{K}$. Since the number of models is N , we get a polynomial time algorithm with $N \log n$ oracle queries. The type of oracle depends on whether the models are binary or general. For binary models it is enough to have an NP oracle, whereas for general models we need a Σ_2^P -oracle. We do not have a matching lower bound but, as we show in the full paper, any binary-search style algorithm for computing the degree of blame requires $\Omega(N \log n)$ oracle queries, for $N = o(n)$.

Acknowledgment We thank Michael Ben-Or and Orna Kupferman for helpful discussions.

References

- [Chockler *et al.*, 2003] H. Chockler, J. Y. Halpern, and O. Kupferman. Causality and responsibility in temporal logic model checking. Unpublished manuscript. Available at <http://www.cs.cornell.edu/home/halpern/papers/resp.ps>, 2003.
- [Collins *et al.*, 2003] J. Collins, N. Hall, and L. A. Paul, editors. *Causation and Counterfactuals*. MIT Press, Cambridge, Mass., 2003.
- [Eiter and Lukasiewicz, 2002a] T. Eiter and T. Lukasiewicz. Causes and explanations in the structural-model approach: tractable cases. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI 2002)*, pages 146–153, 2002.
- [Eiter and Lukasiewicz, 2002b] T. Eiter and T. Lukasiewicz. Complexity results for structure-based causality. *Artificial Intelligence*, 142(1):53–89, 2002.
- [Hall, 2003] N. Hall. Two concepts of causation. In J. Collins, N. Hall, and L. A. Paul, editors, *Causation and Counterfactuals*. MIT Press, Cambridge, Mass., 2003.
- [Halpern and Pearl, 2001a] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, pages 194–202, 2001. The full version of the paper is available at <http://www.cs.cornell.edu/home/halpern>.
- [Halpern and Pearl, 2001b] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part II: Explanations. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 27–34, 2001. The full version of the paper is available at <http://www.cs.cornell.edu/home/halpern>.
- [Hume, 1739] D. Hume. *A Treatise of Human Nature*. John Noon, London, 1739.
- [Papadimitriou, 1984] C.H. Papadimitriou. The complexity of unique solutions. *Journal of ACM*, 31:492–500, 1984.
- [Pearl, 2000] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, 2000.
- [Stockmeyer, 1977] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.
- [Zimmerman, 1988] M. Zimmerman. *An Essay on Moral Responsibility*. Rowman and Littlefield, Totowa, N.J., 1988.

Causes and Explanations Revisited

James D. Park

Computer Science Department
University of California, Los Angeles, CA 90095
jd@cs.ucla.edu

Abstract

This paper reconsiders the notions of actual cause and explanation in functional causal models. We demonstrate that isomorphic causal models can generate intuitively different causal pronouncements. This occurs because psychological factors not represented in the model determine what criteria we use to determine causation. This partially explains the difficulty encountered in previous attempts to define actual cause. Freed from trying fit all examples to match intuition directly (which is not possible using only the information in causal models), we provide definitions for causation matching the different causal criteria we intuitively apply. This formulation avoids difficulties associated with previous definitions, and allows a more refined discussion of what constitutes a cause in a given situation. The definitions of actual cause also allow for more refined formulations of explanation.

1 Introduction

Identifying an actual cause in a specific situation is a normal part of every day human reasoning. From the specifics of the situation, we sift through the events, identifying those that actually caused the event in question. This is different than general claims about causal tendencies. In fact, the causal nature of events can actually run counter to the general causal tendency of the variables. For example, buying a lottery ticket has the general tendency to decrease wealth. However, in certain circumstances (the numbers on the ticket correspond to the winning numbers), it can cause a significant increase wealth. People are able to make these situation dependent pronouncements fairly easily.

Automating causal determinations has a number of applications, including natural language processing, diagnosis, and planning. Unfortunately, formulating a successful notion of actual cause has proved elusive. Previous attempts were often hindered by trying to embed them in formalisms that are not appropriate for modeling causality (See [Pearl, 2000, Chapter 10] for a discussion of these). Recently, functional causal models [Pearl, 2000] have proven to be an effective formalism for modeling causal relationships. There

have been previous attempts to use functional causal models as a basis for defining actual cause, but they have been shown to suffer from serious defects of their own [Hopkins & Pearl, 2003]. In this paper, we identify (at least partially) why previous methods have failed, and provide new definitions that avoid many of their problems. These definitions allow for a more refined determination of actual causation. Additionally, they provide a useful basis for producing explanations.

2 Review of causal models

In this section we review the basic definitions from [Pearl, 2000] that we will use. First we need some notation. Upper case letters (e.g. X, Y, Z) represent variables. Lower case letters (e.g. x, y, z) represent a particular value of the corresponding variable. Bold face upper case letters (e.g. $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$) represent sets of variables, and bold face lower case letters (e.g. $\mathbf{x}, \mathbf{y}, \mathbf{z}$) an instantiation of the corresponding variables.

A *causal model* is a triple $(\mathbf{U}, \mathbf{V}, \mathcal{F})$ where $\mathbf{U}$ is the set of background (or exogenous) variables (i.e. their values are determined outside the model), and $\mathbf{V}$ is the set of endogenous variables (i.e. those whose value is determined by model variables). $\mathcal{F} = \{f_X \mid X \in \mathbf{V}\}$ is a set of functions where f_X maps from $\mathbf{U} \cup (\mathbf{V} \setminus X)$ to X . Together, for any setting of the background variables, $\mathcal{F}$ determines the values of all variables in $\mathbf{V}$. The *parents* of a variable X are the variables that directly influence the value of X (i.e. the set of variables that is non-trivial in f_X). The causal diagram of a model is a directed graph with vertices $\mathbf{U} \cup \mathbf{V}$ where there is an edge from vertex X to vertex Y iff X is a parent of Y in the model. We will restrict our attention to models with acyclic causal diagrams. A submodel of causal model M under intervention $\mathbf{X} = \mathbf{x}$ is a causal model $M_{\mathbf{x}} = (\mathbf{U}, \mathbf{V}, \mathcal{F}_{\mathbf{x}})$ where $\mathcal{F}_{\mathbf{x}} = \{f_Y \mid Y \in \mathbf{V} \setminus \mathbf{X}\} \cup \{\mathbf{X} = \mathbf{x}\}$. This allows us to talk about what would have happened if $\mathbf{X}$ had been $\mathbf{x}$. We will write $\mathbf{Y}_{\mathbf{x}}(\mathbf{u})$ to mean the value that $\mathbf{Y}$ takes on in the submodel $M_{\mathbf{x}}$ in context $\mathbf{U} = \mathbf{u}$. For a propositional formula φ , defined over primitives of the form $X = x$, $\varphi_{\mathbf{x}}(\mathbf{u})$ represents the value of the formula when the primitives are evaluated in model $M_{\mathbf{x}}$ in context $\mathbf{u}$.

3 Previous work

Formulating the notion of cause has been a topic of philosophical debate for many years. Unfortunately, all of the philo-

sophical works of which we are aware either suffer from serious problems, or are not precise enough to be falsifiable (See [Halpern & Pearl, 2001a] for a review of some of these problems).

There have been at least two previous attempts to formulate actual cause in functional causal models. The first was the causal beam approach [Pearl, 2000]. This notion of actual cause seemed to be too restrictive, classifying certain events as noncauses that intuitively clearly should have been causes (See for example the critique in [Halpern & Pearl, 2001a]). The other [Halpern & Pearl, 2001a] (which we will call ASC for Alternate situation Strong Cause) tried to remedy those problems, but introduced several others. [Hopkins & Pearl, 2003] provides a thorough analysis of its problems.

4 Why so hard?

The many formulations encountered, and the problems they exhibit beg the question: why is ascribing cause so difficult to formalize? Consider the following example from [Halpern & Pearl, 2001a]: Two arsonists light matches at opposite ends of the forrest. Both start fires, and the forrest burns down. Either arsonist alone was sufficient to burn down the forrest. Now, we ask whether arsonist 1 lighting his match caused the forrest to burn down. Clearly it did. The reason being that it was sufficient to burn it down, and participated in the actual causal process that brought it about.

Consider another example: A prisoner is scheduled for execution. The execution proceeds as follows: The executioner selects a gun from among several, which may or may not have a bullet in the chamber. The prisoner is given the option of death by lethal injection (which is always fatal), or face execution by gunshot. If the prisoner selects gunshot, the executioner fires the weapon. If the weapon he selected had a bullet, it kills the prisoner. If not, the execution fails and the prisoner goes free. In the situation we consider, the prisoner chooses execution by lethal injection. As it turns out, the selected gun was loaded so the prisoner would have died even if he had chosen gunshot.

Now, we ask if the prisoner's choice of execution method caused his death. Obviously it did not. The reason being that in this circumstance, regardless of his choice, he would still die. That is, his choice had no impact on the outcome.

Surprisingly, the obvious causal models in these two examples are in fact isomorphic as are the specific situations considered. The effect (forrest burns down $\leftrightarrow$ death) occurs in all but one possible instantiation of its direct causes (neither arsonist lights their match $\leftrightarrow$ execution by gunshot is selected and the gun is not loaded) and both sufficient conditions occurred in the situation under consideration. In other words, the information that rendered one a cause and the other a non-cause is not contained in the causal model, or the specific situation. We achieved intuitively different results because we tested different notions of cause. In the forest fire example, we tested the ability to produce the fire, i.e. whether it was capable of bringing about the result. In the second example, we tested the consequentiality of the choice of execution methods, i.e. whether an alternate decision would produce a different result. The choice of what test to apply seems to be

based on psychological factors outside of the causal model.

We believe that the difficulties associated with previous definitions are at least in part due to the fact that they failed to distinguish between different notions of cause.¹ In fact, we would argue that like many problems in philosophy, one reason cause eludes satisfactory definition is because the intuitive concept is fundamentally imprecise. What criteria to apply is often a subjective decision, witnessed by the ease with which one can often convince others that an alternate criteria is the appropriate one. For example, one may argue that it is inappropriate to say that arsonist 1 caused the fire since regardless of his action, the fire would still have occurred.

While the choice of criteria for determining cause is imprecise and subject to factors not representable in a functional causal model, the same is not true once a criteria has been chosen. Each criteria is precise and completely decidable based on the causal model and the state of its variables. Formulating and illustrating appropriate definitions of these criteria is the focus of the next section.

5 A new proposal

In this section, we introduce basic definitions which we then compose to form criteria for evaluating causal relationships. As there are several ways in which an event may be considered a cause, we provide definitions for the different aspects. We believe that all causal tests are based on two basic ideas, which are related to, though distinct from, necessity and sufficiency. By combining these two criteria we can produce the different criteria used for making causal judgments.

First, we need some background and auxiliary definitions. Foremost among them is what we allow to be causes and effects.

Definition 1 *An instantiation $\mathbf{x}$, of endogenous variables $\mathbf{X}$ is an event in context $\mathbf{u}$ if $\mathbf{X}(\mathbf{u}) = \mathbf{x}$. A propositional formula φ defined over the endogenous variables of the model is an effect in context $\mathbf{u}$ if $\varphi(\mathbf{u})$.*

Any potential cause of an effect must be an event. Note that we force events and effects to be true in the situation being considered. This is merely a convenience meant to simplify subsequent definitions.

Since we are concerned with determining cause in a particular context, the causal diagram of a model is not very informative. A variable X that in general is necessary to determine the value of another variable Y (and hence has an edge from X to Y in the diagram), may become irrelevant in a particular context. That is, a particular value of the background variables $\mathbf{u}$ may make the value of Y unresponsive to the value of X . This observation motivates the following definition:

Definition 2 *The context diagram of causal world $M(\mathbf{u})$ is the directed graph whose vertices are the endogenous variables of M , and includes each edge (X, Y) where the function f_Y is nontrivial in X when the background variables are fixed to $\mathbf{u}$.*

The context diagram simplifies reasoning about causal relationships in a specific context.

¹[Hall, 2002] makes a similar argument, although the details differ from ours.

Another preliminary notion we will need is sufficiency.²

Definition 3 An event x is sufficient to cause effect φ in context u if for all instantiations w of the nondescendants W of X in the context diagram of $M(u)$, $\varphi_{xw}(u)$.

Note that it must hold only when the nondescendants are manipulated. Descendants are excluded from manipulation since we believe that the values they take on, including interplay between them, are an integral component of determining causation. Allowing them to be manipulated would upset and skew that process.

Sufficiency is not really strong enough for our purposes. For example, consider this example from [Hall, 2002]:

Suzy and Billy both pick up rocks and throw them at a bottle. Suzy's rock gets there first, shattering the bottle. Since both throws are perfectly accurate, Billy's would have shattered the bottle had it not been preempted by Suzy's throw

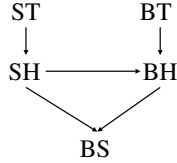


Figure 1: Rock throwing example

Figure 1 shows the causal diagram of a model of the story (the background variables, as well as the functional relationship between variables should be obvious in this story, as well as the others, so will not be explicated). BT (ST) represents Billy (Suzy) throwing BH (SH) represents Billy's (Suzy's) rock hitting the bottle, and BS stands for the bottle shattering. Billy throwing is sufficient to cause the bottle to shatter, but it doesn't actually participate in the causal process that made the bottle shatter. Suzy's action preempted Billy's. We expect not only sufficiency of the cause, but that in some sense, the event actually participates in the causal process. Defining this notion has been one of the primary difficulties in talking about actual cause. The following two definitions formalize these notions in a way that avoids the problems associated with previous approaches.

Definition 4 Let Z be the children of X in the context diagram of u , and let Y be some non-empty subset of Z . Let W be the intersection of the descendants of X (including X) and the ancestors of Y (excluding Y). Then event xy is a child of event x in context u if $W \subseteq X$.

A child event can be thought of as a next step in an unfolding causal process. The criteria that $W \subseteq X$ is merely to enforce the intuition that before a variable takes on a value, its direct

causes must be fixed. That is, a new value can not be added, if the relevant parents have not already been included.

Definition 5 An event x is a strongly sufficient cause of effect φ in context u if either of the following conditions hold:

- For all instantiations z of $Z = V \setminus X$, $\varphi_{xz}(u)$
- x is a sufficient cause of φ and there exists some child event y of x which is a strongly sufficient cause of φ in context u

Effectively, what we require is that the event be sufficient for the effect, and that as the causal process progresses from the event, the sufficiency is maintained. This prevents events that are preempted from being strongly sufficient. For example, while $\{BT=T\}$ is sufficient for $\{BS=T\}$, the only child event, $\{BT=T, BH=F\}$ is not, so the event is not strongly sufficient. Suzy throwing, on the other hand, is strongly sufficient, witnessed by the sequence $\{ST=T\}$, $\{ST=T, SH=T\}$, $\{ST=T, SH=T, BH=F\}$, $\{ST=T, SH=T, BH=F, BS=T\}$.

Strong sufficiency is the first of the two primary notions used to evaluate causality in specific situations. In fact, strong sufficiency by itself corresponds to one of the primary notions of causation. Strong sufficiency corresponds to our intuitions about physical causality. The cause sets in motion a chain of events capable of bringing about the effect. Intuitively, we expect causation to be transitive. If x causes y and y causes z , we expect x to cause z .³ It is not hard to see that strong sufficiency meets that expectation. In fact, strong sufficiency turns out to be the only transitive relation among our definitions of cause.

The other primary consideration when testing actual causality is the notion of consequentiality. In addition to participating in the causal process, we often require that an event be in some sense consequential in order to be considered a cause. That consideration motivates the following definition.

Definition 6 Let Z be the nondescendants of X in the context diagram of u . Then, an event x is potentially consequential with respect to effect φ in context u if for some instantiations x' of X , and z of Z , $\varphi_{xz}(u)$ and $\neg\varphi_{x'z}(u)$. Event x is actually consequential with respect to effect φ if for some instantiation x' of X , $\neg\varphi_{x'}(u)$.

Like strong sufficiency, actual consequentiality alone is a common criteria for causation. For example, in the execution example from Section 4 the choice of lethal injection was not considered a cause of his death precisely because it was not consequential. If instead the gun had not been loaded, which would make the decision consequential, we would consider it a cause. Note that a consequential event necessarily participates in the actual causal process. That is, for every actually consequential event, there is a minimal strongly sufficient event that contains some member of the consequential event.

Potential consequentiality on the other hand is not of itself enough for an event to be considered a cause. However, combining it with strong sufficiency produces a very common causal criteria.

²Note that this definition differs from the definition for sufficiency presented in [Pearl, 2000]. That definition was used for a different purpose, and is not appropriate in this context. It is more closely related to, although distinct from the notion of sustenance [Pearl, 2000].

³In fact, this intuition is so strong that some theories of causation start with the transitivity of cause as a starting point [Lewis, 1986].

Test Strong Sufficiency	Test Consequentiality		
	Actual	Potential	No
Yes	Strong	Sustaining	Strongly Sufficient
No	Consequential	—	—

Table 1: Different types of actual cause can be tested by combining tests for strong sufficiency and consequentiality.

Definition 7 An event x sustains effect φ in context u if x is strongly sufficient and potentially consequential for φ .

The notion of sustenance defined here is a modified version of the one that appeared in [Pearl, 2000]. It differs in two ways. First, it requires strong sufficiency as opposed to merely sufficiency. Second, it fixes the variables W , against which x must be tested, while the previous definition has W as one of the parameters.

Sustenance is a very commonly applied criteria for causation. Most of the examples in [Halpern & Pearl, 2001a] which were deemed causes correspond to our notion of sustaining cause. In the forrest fire example in Section 4, arsonist 1 was considered a cause of the fire because it sustained the forrest fire.

A more restrictive causal criteria is also sometimes useful.

Definition 8 An event x is a strong cause of φ in context u if x is a strongly sufficient and actually consequential cause of φ .

A strong cause can be thought of as a complete explication of why the effect occurred.

Often we talk about an event helping to cause an event even though it is neither consequential nor sufficient. It is instead part of some cause. That consideration motivates the following definition.

Definition 9 An event xy (y may be empty) is a contributory cause of effect φ in context u if x is potentially consequential, and for some event z , xz is a minimal sustaining cause of φ .

Contributory causation is the closest in spirit to the ASC definition. If we insist on minimality, it reduces to singletons just as that definition does. Additionally, in all but one example included in [Halpern & Pearl, 2001a], contributory causation matches the pronouncements of the ASC definition. We consider that example among others in the next section.

6 Examples

The single example presented in [Halpern & Pearl, 2001a] where the ASC definition differs from our definition of contributory causation comes from the following story (along with some commentary), quoted from [Hall, 2002]:

You are standing at a switch in the railroad tracks. Here comes the train: If you flip the switch, you'll send the train down the left-hand track; if you leave where it is, the train will follow the right-hand track. Either way, the train will arrive at the same point, since the tracks reconverge up ahead. Your action is not among the causes of this arrival; it merely helps to determine how the arrival

is brought about (namely, via the left-hand track, or via the right-hand track).

Two models which produced different causal pronouncements were given, demonstrating the impact that model choice has on the determination of causation. The model that classified it as a cause is the one we are interested in. Figure 2 shows the model. F represents the state of the switch, LT (RT) whether the train goes down the left (right) track, and A represents whether the train arrives. The justification given was

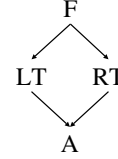


Figure 2: A possible model for the train story

that under normal circumstances, we may not call it a cause, but if something abnormal were to happen (for example, one of the tracks malfunctions, or is destroyed) then the action would be a cause. Our new definitions say that flipping the switch is a strongly sufficient cause (In the words of Hall it “helps to determine how the arrival is brought about”), but it is not even potentially consequential, so is not a contributory cause. This seems a more satisfactory result.

Unlike the ASC definition, these definitions do not always reduce to singleton causes even if minimality is enforced. Additionally, the different criteria produce causes that correspond to different ways that people think about cause.

For example, consider a situation where 5 people vote regarding a ballot measure. The votes are tallied, and the measure passes if 3 or more vote for it. The associated diagram appears in Figure 3. Suppose that 4 vote for the measure, and 1 votes against. According to the ASC definition, each in-

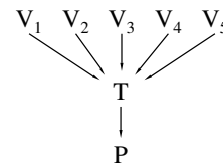


Figure 3: A voting example. The V_i correspond to the votes cast, T the vote tally, and P whether the measure passed.

dividual who votes for the measure is an actual cause of the measure passing. In the causal beam approach, nothing qualifies as a cause of the measure passing. In this approach, it depends what type of cause we are considering.

Three votes for the measure constitute a strong cause of the measure passing, while two do not. This corresponds to our notion that a majority of the votes is required to cause it to pass.

Two votes for the measure constitute a consequential cause. Changing those votes to vote against the measure would make

the measure fail. This corresponds to the way we often talk about causes in elections. Often election outcomes are attributed to the voting behavior of some minority of the electorate (in U.S. politics these include “Reagan Democrats”, “soccer moms”, “angry white men”, “African Americans”, etc). While they are far from sufficient to win an election, we say that they caused the election result because if they had voted significantly differently the outcome of the election would have been different. On the other hand, most people understand that in the vast majority of elections their single vote did not cause the outcome. This is reflected in our definitions by the fact that none of the criteria yield a single voter as a cause in this situation. A single vote for the measure is a contributory cause however, as one would expect.

[Halpern & Pearl, 2001a] provides a number of examples that have proved problematic for other methods. As most of the examples correspond either to issues of sustenance or consequentiality⁴ (and with the exception of the train example considered above, all of the positive examples are contributory causes, and the negative examples are not), we will not review them all here. We would like to revisit one example that was considered troublesome. The first version of the story is as follows: A machine severed Fred’s finger while he was at work. He was rushed to the hospital and it was reattached. A month later his finger worked just as before. We would of course not consider getting his finger cut off a cause of it being fully function later (because it is not consequential). Now, suppose we add another component to the story: It is possible that Larry the loanshark may be waiting outside for Fred, so that he can cut off his finger to make an example of him for not paying his loan promptly. If Larry cuts off the finger, he will destroy it so that it can not be reattached. If Fred severs his finger at the factory, he will be rushed to the hospital, so Larry will not have an opportunity to do it himself. The causal diagram appears in Figure 4. Now, consider the

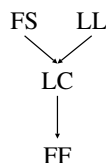


Figure 4: Finger severing example. FS represents severing his finger. LL represents whether Larry is lurking. LC represents whether Larry cuts off his finger and FF represents whether his finger is fully functional

situation in which Larry turns out not to be waiting outside, and Fred does sever his finger. Was Fred cutting his finger off a cause of his finger being fully functional the next month? According to the ASC definition it is. This was considered disturbing because adding a provision for a contingency that did not occur changed the the event from noncause to cause.

⁴In fact, most of the difficult examples play off of the conflicting interpretation of cause between sustenance and consequentiality. That is, the event in question is typically either consequential or sustaining, but not both.

In light of our new definitions, we do not see that particular consideration as a problem. According to our definition, it again depends on what you are looking for. It is sustaining, but it is not consequential. Typically, when we are testing if an event caused something to remain unchanged, we are looking for a consequential cause. That is clearly the case in this example. However, it is still reasonable to say that cutting off his finger at the factory was a sustaining cause, because that event makes certain the effect, regardless of what Larry does or does not do.

Our definitions also handle correctly the examples in [Hopkins & Pearl, 2003] avoiding the problems of ASC, but because of space limitations we will not review them all here. We will consider a single example of the problem to demonstrate how these definitions avoid it. Consider the following story: Before an execution, the captain loads one of the guns of the two members of his firing squad, putting a blank in the other gun. Both gunmen then fire simultaneously and the prisoner dies. In the current situation, the captain put the bullet in the gun of gunman 1, and both men shoot. The model consists of the loader, and the two gunmen as direct causes of the the variable representing whether the prisoner is alive. Now, we ask if gunman 2 shooting caused the prisoner’s death. According to the ASC definition, it did. Gunman 2 did not cause the prisoner’s death according to any of our definitions of cause. It is clearly not consequential, nor sufficient, so it can not be a cause. It is not a contributory cause either because it is not a member of any minimal sufficient cause. According to these definitions, gunman 1 shooting is a consequential cause, gunman 1 shooting and the captain loading his gun is a strong cause, and gunman 1 shooting and the captain loading his gun are both contributory causes.

7 Explanations

If we have a probability distribution quantifying our uncertainty about what context will actually occur, we can use that distribution, combined with our model to produce explanations for an observed effect.

Causation forms an integral part of evaluating explanations (see [Chajewska & Halpern, 1997; Pearl, 2000; Halpern & Pearl, 2001b] for critiques of methods that ignore causal considerations when producing causes). In spirit, we agree with the intuitive notion of explanation put forth in [Halpern & Pearl, 2001b]. That is, an explanation is fundamentally just a hypothesized event which if true would be a possible cause of the observed phenomenon. Our new view of actual causation, however, makes the particulars somewhat different. Our definitions of actual cause form the basis for evaluating possible explanations. We allow any instantiation of endogenous variables to be an explanation and compare them based on how well they explain the effect. According to our view, there are different types of explanation, just as there are different types of causes. Which type of explanation to choose depends on its intended purpose. This is in contrast to the approach using ASC as the basis for causal explanation. We believe that this approach provides more flexibility and is more general than ASC based explanations.

By way of example, consider the following situation: We

Explanation Variable	Posterior =0	Posterior =1	Explanatory Power =0	Explanatory Power =1
X_1	.52	.48	.35	.65
X_2	.67	.33	.40	.60
X_3	.74	.26	.42	.58

Table 2: Posteriors and explanatory power for possible explanations in the circuit example according to ASC definition.

have a circuit consisting of a single 3 input parity gate, with inputs X_1 , X_2 and X_3 , and output Y (i.e. $Y = 1$ when an odd number of the inputs are 1). Each input has an associated background variable, independent of the others, where $X_j = U_j$ and $\Pr(U_j = 1) = 1/(j + 2)$. So, typically $Y = 0$ and $X_j = 0$. Now, suppose that we observe that $Y = 1$, and we want an explanation in terms of the X variables. The first question is what constitutes an explanation (or even a partial explanation in the ASC definition). According to the ASC based definition, the only possible explanations (or partial explanations) are singletons.⁵ Now, how can we compare between them? The ASC approach provides three measures of the quality of an instantiation $\mathbf{x}$ as an explanation of effect φ in light of evidence $\mathbf{e}$ (which must entail φ). The first, named goodness is $\Pr(\mathbf{x} \text{ causes } \varphi | \mathbf{x}\mathbf{e})$.⁶ The second, the posterior probability is $\Pr(\mathbf{x} | \mathbf{e})$. The third, explanatory power is $\Pr(\mathbf{x} \text{ causes } \varphi | \mathbf{x})$. The goodness of all explanations in this example are 1. Also, the explanatory power, $\Pr(\mathbf{x} \text{ causes } \varphi | \mathbf{x}) = \Pr(\mathbf{x}\varphi | \mathbf{x})$ since every instantiation is a weak cause according to the ASC definition. Table 2 shows the explanatory power and posterior probability of the different possible explanations. It is not clear how to choose between them in this case. One thing to notice about this example is that every variable is consequential in every context, so notions of consequential causation (or by extension, contributory causation) become less interesting. We would argue that in this case, what we are really interested in is sustenance. Our definitions allow that to be considered easily. For a particular instantiation we can calculate the probabilities $\Pr(\mathbf{x} \text{ sustains } \varphi | \mathbf{e})$, $\Pr(\mathbf{x} \text{ strongly causes } \varphi | \mathbf{e})$, $\Pr(\mathbf{x} \text{ is a consequential cause of } \varphi | \mathbf{e})$ from the model and the probability distribution over the background variables. For a particular causal criteria $c(\mathbf{x}, \varphi, \mathbf{u})$, the probability that $\mathbf{x}$ is a cause of φ given $\mathbf{e}$ using criteria c is just

$$\Pr(\mathbf{x} \text{ "c caused" } \varphi | \mathbf{e}) = \frac{\sum_{\mathbf{u} \in \{\mathbf{u} | c(\mathbf{x}, \varphi, \mathbf{u}) \wedge \mathbf{E}(\mathbf{u}) = \mathbf{e}\}} \Pr(\mathbf{u})}{\sum_{\mathbf{u} \in \{\mathbf{u} | \mathbf{E}(\mathbf{u}) = \mathbf{e}\}} \Pr(\mathbf{u})}.$$

The probability of causation can then be used as the measure of explanation quality. In our circuit example, for example, $\Pr(X_1 = 1, X_2 = 0, X_3 = 0 \text{ sustains } Y = 1 | Y = 1) = .444$, which is the maximal probability for sustenance. This explanation (which is not allowed in the ASC version) seems more satisfying than any of the single variable explanations.

⁵Note that this is not always the case. This example was specifically crafted to exhibit such behavior.

⁶In the ASC definition of explanation, it "causes" would mean is a weak cause. We write it in a general form, because these measures are applicable to the new method as well

This is not to say that the measures used in [Halpern & Pearl, 2001b] are not useful. In fact they can be very useful. We simply note that our definitions allow a further analysis that is useful in other situations. They prove especially useful when we are looking for a sufficient or sustaining explanations. Because our definitions can handle different notions of cause, the explanations they produce can be more informative than the explanations formed when using the methods in [Halpern & Pearl, 2001b], where they are restricted to one weak form of cause.

8 Conclusion

We demonstrated that the intuitive choice of causal criteria is subjective, and is influenced psychological factors not representable in a causal model. The criteria, once chosen however can be evaluated from the information contained in the causal model. We presented new criteria covering the different notions of causation. Fundamental among these is the definition of strong sufficiency, which provides a simple and intuitively satisfying definition for actually participating in the active causal process that generated the effect. These definitions avoid the problems of previous attempts, and allow for a more refined discussion of actual causality. Additionally, they provide a foundation for evaluating explanations, which proves to be more satisfactory than previous attempts to use functional causal models for explanation.

Acknowledgment

This work has been partially supported by MURI grant N00014-00-1-0617

References

- [Chajewska & Halpern, 1997] Chajewska, U., and Halpern, J. 1997. Defining explanation in probabilistic systems. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, 62–71.
- [Hall, 2002] Hall, N. 2002. Two concepts of causation. In *Causation and Counterfactuals*. MIT Press.
- [Halpern & Pearl, 2001a] Halpern, J., and Pearl, J. 2001a. Causes and explanations: A structural-model approach—part I: Causes. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, 194–202.
- [Halpern & Pearl, 2001b] Halpern, J., and Pearl, J. 2001b. Causes and explanations: A structural-model approach—part II: Explanations. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 27–34.
- [Hopkins & Pearl, 2003] Hopkins, M., and Pearl, J. 2003. Clarifying the usage of structural models for commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*.
- [Lewis, 1986] Lewis, D. 1986. Causation. In *Philosophical Papers*, volume II, 159–213. Oxford University Press.
- [Pearl, 2000] Pearl, J. 2000. *Causality*. Cambridge University Press.

COGNITIVE MODELING

GHOST: experimenting conflicts countermeasures in the pilot's activity

Frédéric Dehais, Catherine Tessier and Laurent Chaudron

Onera-DCSD

31055 Toulouse cedex 04 - France

{dehais,tessier,chaudron}@cert.fr

Abstract

An approach for designing countermeasures to cure conflict in aircraft pilots' activities is presented, both based on Artificial Intelligence and Human Factors concepts.

The first step is to track the pilot's activity, i.e. to reconstruct what he has actually done thanks to the flight parameters and reference models describing the mission and procedures. The second step is to detect conflict in the pilot's activity, and this is linked to what really matters to the achievement of the mission. The third step is to design accurate countermeasures which are likely to do better than the existing onboard devices. The three steps are presented and supported by experimental results obtained from private and professional pilots.

1 Introduction

The review of civilian and military reports reveals that a conflictual situation is a precursor to the loss of aircrews' situation awareness and is a major cause of air accidents. Conflict may occur through different forms: open conflict between the different operators (Air Philippines crash, April 1999), representation conflict about the aircraft position (Korean air, August 1997), resource conflict (Streamline, May 2000), knowledge conflict (Crossair, January 2000), or more recently, a conflict between automated systems (TCAS) and human operators (air collision between a Tupolev 154 and a Boeing 757 over Switzerland, July 2002). Therefore, the idea is to detect conflict so as to propose accurate on-line countermeasures to pilots.

The first step is to track the pilot's activity, i.e. to reconstruct what he has actually done thanks to the flight parameters and reference models describing the mission and procedures. The second step is to detect conflict in the pilot's activity, and this is linked to what really matters to the achievement of the mission. The third step is to design accurate countermeasures which are likely to do better than the existing onboard devices. The three steps are presented and supported by experimental results obtained both from private and military pilots.

2 Track the pilot's activity

2.1 Reference model

The approach to capture the pilot's activity is based on situation tracking classically used for system diagnosis, monitoring and intelligence [Ghallab, 1996; Chaudron *et al.*, 1997; Rota and Thonnat, 2000]. Whatever the formalism used to represent knowledge, situation tracking rests on a matching between observed facts and known situations (reference models), and its object is to carry out postdictions, predictions or to diagnose the current state of the system.

To represent knowledge, we use the concept of *propositional attitude*, "which covers the notions of belief, desire, intention, [...] which have in common the fact of being identified by their propositional contents" [Tessier *et al.*, 2000].

Definition: a propositional attitude (PA) is a pair (K, T) where K is a logical cube, i.e. a conjunction of first order literals [Chaudron *et al.*, 2003], and T is the time interval within which the properties captured by K are true.

The pilot's knowledge is divided into three categories of PAs, which constitute the reference model:

- Propositional attitude *goal* describes a goal of the mission. Ex: `pa_goal(goal(5), <Landing(francazal)>, T)` means that *the fifth goal of the mission is to land at Francazal during time interval T*.

- Propositional attitude *plan* describes the properties the pilot is committed to satisfy to reach a PA-goal as long as he carries out this goal.

Ex: `pa_plan(goal(5), <VOR.frequency(116), VOR.heading.deflection(true), gear_down(true), flaps_down(2), visibility(true)>, T)`

means that *to satisfy goal 5, the pilot has to select the accurate radio frequency, fly a correct route, lower gears and flap deflection 2 and have a good visibility during time interval T*.

- A *crucial* propositional attitude describes the hard constraints associated to one or several goals.

Ex: `pa_crucial(goal(5), <visibility(true)>, T)` means that to succeed in landing at Francazal, the pilot must have a correct visibility during T.

Remark: the model does not implement deontic logic [Traum and Allen, 1994] to capture the pilot's obligations; at each time step, the system will check whether the pilot meets the different constraints on the PA-goals.

2.2 Observed activity

The pilot's activity is observed and analyzed through his actions on the airplane interface: during the experiments, the flight parameters are recorded every second. The parameters are then translated into symbolic knowledge and aggregated to build *observation propositional attitudes*.

The numerical thresholds that are used for numerical to symbolic translation are adjusted during interviews with the pilots to build an accurate model for each pilot. For example, if the pilot has to fly 2500 ft, he defines his own flight envelope where he feels secure [Amalberti and Wioland, 1997]. After numerical to symbolic translation, parameter altitude is characterized as `altitude(false)` (i.e. *out of flight envelope*) or `altitude(true)` (i.e. *within flight envelope*).

The same kind of analysis is performed with parameters such as the route, the fuel level, etc.

Therefore, propositional attitude *observation* describes perceived and deduced facts from the pilot's activity:

Ex: `pa_obs(<VOR_frequency(116),`
`VOR_heading_deflection(true), gear_down(true),`
`flaps_down(3), visibility(false)>, <1500, 1500>)`
means that *at time 1500, the pilot has selected the accurate radio frequency, is flying a correct route, has lowered gears and flap deflection 3 and has a bad visibility.*

3 Detect conflict

3.1 Conflict

Considering research in the field of A.I. and particularly on multi-agent systems, the focus has been brought much on how to avoid, solve or get rid of conflicts, especially *via* negotiation [Sycara, 1989; Rosenschein and Zlotkin, 1994], argumentation [Jung and Tambe, 2000] or constraint satisfaction [Conry *et al.*, 1991; Khedro and Genesereth, 1994; Hannebauer, 2000]. Such approaches, where (i) conflict is likened to formal inconsistency and (ii) the internal knowledge of the agents is clearly identified, are not suited to our purpose. Indeed in aeronautics, uncertainty on data is high and the human behavior is extremely variable and unpredictable. Moreover, while in air, aircrews frequently face incoherent situations due to the uncertain environment (e.g. weather, failures) or because of their own errors. Nevertheless, these situations are not necessarily conflictual, and pilots are used to going on with their missions with such inconsistencies. An inconsistency becomes conflictual only if it leads to a critical flight phase and *matters* to the mission or to the aircrew's safety.

Therefore we are very close to Easterbrook's definition of conflict [Easterbrook, 1991]: "*[...] viewpoints are free to differ, and only conflict when that difference matters for some reason, leading to interference. [...] A conflict, then, is simply a difference that matters.*" This point of view has been emphasized in social sciences [Festinger, 1957; Lewin *et al.*, 1939] and in A.I. applied to social modeling [Castelfranchi, 2000], where human conflict is seen as the result of an impossibility to succeed in a goal that matters.

Definition: a *conflict* is a set of propositional attitudes that is

not coherent¹ and the fact that it is not coherent matters.

The inconsistency of a set of PAs *matters* either because there is a need:

- to satisfy the rationality² of the agent(s) involved (pilot, copilot ...);
- or to build a coherent set of knowledge;
- or to decide on further goals.

All these different reasons are implemented in the reference model as *crucial* propositional attitudes (see section 2.1).

Therefore a *conflict* is an incoherent set of knowledge that matters through one or several *crucial propositional attitudes*.

3.2 Conflict detection

Conflict detection is a tracking process that monitors the pilot's PAs at each time step t . It is a two-step process: (1) detect the sets of PAs that are not coherent at time t and (2) detect the conflicts given the crucial PAs that concern time t . One could have designed a single-step process, however it is of great interest to track PAs that are not coherent and do not conflict: indeed, contrary to the minimal inconsistent sets approach [De Kleer, 1986], we aim at capturing all the inconsistencies and analyzing how they are managed by the pilots for "lessons learned" and further man-artefact interaction model tuning.

(1) Detect the sets of PAs that are not coherent

Predicate `not_hold_together` detects incoherent sets of PAs Inc_t at time t .

Ex: given the set of PAs

- `pa_goal(goal(5), <Landing(francalaz)>, <1000, t1>)`
 - `pa_plan(goal(5), <VOR_frequency(116),`
`VOR_heading_deflection(true), gear_down(true),`
`flaps_down(2), visibility(true)>, <1000, t1>)`
 - `pa_obs(<VOR_frequency(116),`
`VOR_heading_deflection(true), gear_down(true),`
`flaps_down(3), visibility(false)>, <1500, 1500>)`
- two incoherent sets of PAs are detected:

→ `not_hold_together(pa_plan(5, flaps_down(2)),`
`pa_obs(flaps_down(3), <1500, 1500>);`
→ `not_hold_together(pa_plan(5, visibility(true),`
`pa_obs(visibility(false)), <1500, 1500>).`

(2) Detect the conflicts

The crucial PAs that concern time t are then considered.

Definition: let Inc_t an incoherent set of PAs at time t . If Inc_t includes PAs or properties that matter through one or several crucial PAs, then Inc_t is a conflict.

Predicate `matters` screens sets Inc_t thanks to the crucial PAs, so as to determine whether they are conflicts or not.

Ex: for the previous example, the crucial PA that holds at time 1500 is:

- `pa_crucial(goal(5), <visibility(true)>, <1000, t1>)`

Predicate `matters` detects that one of the two incoherent

¹in relation to the formalism used to represent the properties in the PAs.

²According to Festinger [Festinger, 1957] who postulates that an individual always aims at being coherent when interacting with his environment.

sets of PAs is conflictual:

```
matters(pa.plan(goal(5),visibility(true)),  
pa_obs(visibility(false),<1500,1500>)
```

The other set of PAs relative to flaps deflection is not a conflict because a landing with flap deflection 3 is possible even though less recommended.

3.3 Preliminary experiment and results

A preliminary experiment designed to test conflict tracking [Dehais, 2002] was conducted with ten pilots on a research flight simulator. The pilots were proposed a flight scenario whose aim was to follow a flight plan at a constant altitude and then land with instruments. The difficult point in this scenario was to manage the fuel consumption correctly considering refuelling areas.

This experiment has validated conflict detection and interestingly enough has shown that a psychological *perseveration* phenomenon appeared with conflict : the pilots who faced conflictual situations did not come to the right decision to solve their conflict, on the contrary they got themselves tangled up in it, and all the more so since the temporal pressure was increasing. The most surprising thing was that all the necessary data to come to the accurate decision were available and displayed on the onboard equipments.

Ex: one of the pilots flew toward waypoint 5 instead of flying first over waypoint 6 for refuelling, and started circling around, until he ran out of fuel (see figure 1).

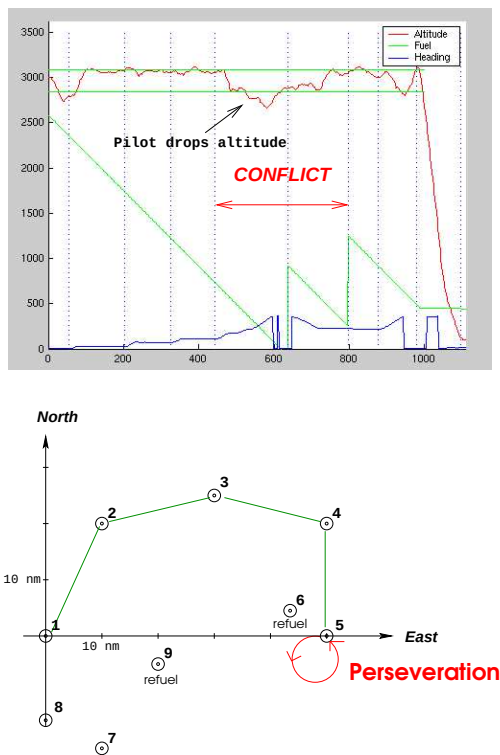


Figure 1: Conflict and perseveration

During the debriefing, the pilot explained that he was con-

vinced that waypoint 5 was the refuelling area (in spite of the map displayed in the cockpit); he reached it and noticed that the fuel level did not increase. Thinking he had missed waypoint 5, he decided to fly back to waypoint 5. He did this manoeuvre again and again till he ran out of fuel, getting more and more stressed.

4 Send accurate countermeasures

4.1 Guidelines

The findings of the preliminary experiments are akin to a recently published report of the BEA³ (the French national institute for air accident analysis) that reveals that pilots' erroneous attitudes of perseveration have been responsible for more than 40 percent of casualties in air crash (in civilian aeronautics). This behavior, called the "perseveration syndrome", is studied in neuropsychology [Vand Der Kolk, 1994; Pastor, 2000] and psychology [Beauvois and Joule, 1999]: it is known to summon up all the pilot's mental efforts toward a unique objective (excessive focus on a single display or focus of the pilot's reasoning on a single task). Once entangled in perseveration, the pilot does anything to succeed in his objective even if it is dangerous in terms of security, and worse, he neglects any kind of information that could question his reasoning (like alarms or data on displays).

Therefore the idea is to design countermeasures in order to break this mechanism and get the pilot out of the conflict. The design of the countermeasures is grounded on the following theoretical and empirical results:

- conflictual situations lead pilots to persevere;
- additional information (alarms, data on displays) designed to warn pilots are often unnoticed when pilots persevere.

Our conjectures are then:

- with an accurate on-line conflict detection, it is possible to analyze and predict the subject of the pilot's perseveration;
- instead of adding information (classical alarms), it is more efficient to *remove the information* on which the pilot is focused and which makes him persevere, and to display a message to explain his error instead.

Ex: in the experiment described afterwards, pilots persevere in trying a dangerous landing at Francalaz despite the bad visibility, with a particular focus on an instrument called the H.S.I.⁴ to locate Francalaz airport. The countermeasures will consist in removing the H.S.I. during a few seconds, to display two short messages instead, one after the other ("*Landing is dangerous*"... "*look at central display*") and next to send an explanation of the conflict on the central display ("*Low visibility on Francalaz, fly back to Blagnac*"). The idea here is to shock the pilot's attentional mechanisms with the short disappearance of the H.S.I., to introduce a cognitive conflict ("*if I land, I crash*") to affect the pilot's reasoning, and to propose a solution on the central display.

³<http://www.bea-fr.org>

⁴The H.S.I. is a display that gives the route to follow and indicates any discrepancy from the selected route.

4.2 GHOST

Ghost is an experimental environment designed to test countermeasures to cure the pilot's perseveration. It is composed of:

- Flightgear⁵, an open source flight simulator, which means that many modifications can be made, e.g. implementing new displays. To fly the airplane, pilots have a stick, rudder and a keyboard. Almost all the airports and beacons (NDB and VOR⁶) of the world are modeled, and the air traffic control is simulated. During the experiment, the flight simulator interface is displayed on a giant screen;
- Atlas⁷, a freeware designed to follow the pilot's route. The airplane trajectory, cities, airports and beacons are displayed with an adjustable focus;
- a wizard of Oz interface (see figure2) we have designed and implemented, which allows a human operator to trigger events (failures, weather alteration and the countermeasures) from an external computer via a local connection (TCP/IP protocol, sockets communication).

As far as the countermeasures are concerned, several actions are available to the wizard of Oz:

- *replace* a display on which the pilot is focused by a black one (time of reappearance is adjustable);
- *blink* a display (frequency and time of blinking is adjustable);
- *fade* a display and *brighten* it again;
- *send a message* to a blinked, removed or faded display;
- *send a message* to the central display for explicit conflict solving.

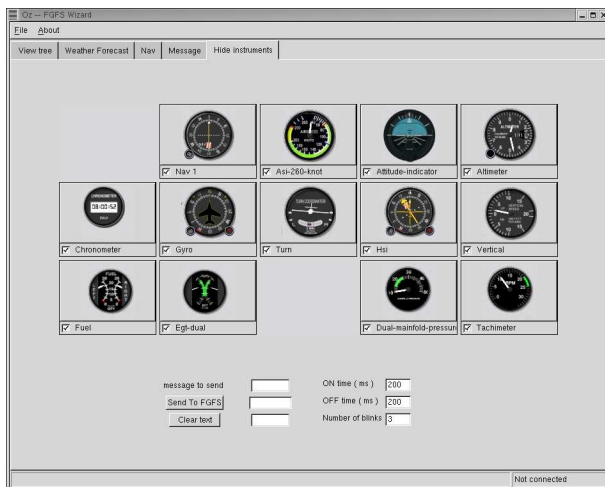


Figure 2: The wizard of Oz interface - countermeasures

4.3 Experimental scenarios

As conflict appears when a desired goal cannot be reached, we have designed three experimental scenarios where a cru-

⁵<http://www.flightgear.org/>

⁶NDB and VOR are two kind of radionavigation beacons used to guide pilots.

⁷<http://atlas.sf.net>

cial goal of the mission cannot be achieved:

- scenario 1 is a navigation task from Toulouse-Blagnac airport to Franczal airport including three waypoints (VOR 117.70, NDB 331 and NDB 423). An atmospheric depression is positioned in such a way that the pilot cannot see the landing ground but at the last moment when it is too late to land on it.
- scenario 2 is a navigation task from Toulouse-Blagnac airport to Franczal airport including three waypoints (VOR 117.70, NDB 415 and NDB 423). The visibility is decreased from the runway threshold: from far away, the landing ground is visible but as the pilot gets closer to it, it disappears totally in a thick fog.
- scenario 3 is a navigation from Toulouse-Blagnac back to Toulouse-Blagnac including three waypoints (VOR 117.70, NDB 331 and NDB 423). An atmospheric depression is positioned over waypoint 2 (NDB 331), and as the pilot flies over this waypoint, the left engine fails.

In scenarios 1 and 2 the pilot's conflict can be summarized as *"should I try a dangerous landing at Franczal or should I fly back to Blagnac for a safer landing?"*. If our conjectures hold, pilots will persevere and try to land at Franczal.

In scenario 3, the pilot's conflict can be summarized as *"should I go on with the mission despite the failure or should I land at Franczal and therefore abort the mission?"*

4.4 Experimental Results

21 experiments were conducted with Ghost in December 2002 (see figure 3). The pilots' experiences ranged from novice (5 hours, small aircraft) to very experienced (3500 hours, military aircraft). Conflicts were detected with the conflict tracker (described above) and were inputs for the wizard of Oz to elaborate the countermeasures.

Results for scenarios 1 and 2: "impossible landing"

In these scenarios, the pilots faced the decision of mission abortion, i.e. not to land at Franczal and fly back to Blagnac. Both scenarios were tested within two different contexts: without countermeasures and with countermeasures.

- Context 1: no countermeasures

7 pilots tested scenario 1 or 2 without any countermeasures (see next table). "Circuits" corresponds to the number of circuits performed by the pilot round Franczal before crashing or landing.

Pilot	Scenario	Circuits	Results
Pilot1	1	3	crash
Pilot2	1	3	crash
Pilot3	1	1	chance landing
Pilot4	1	1	chance landing
Pilot5	2	1	crash
Pilot6	2	1	chance landing
Pilot7	2	1	crash

The results suggest that without any countermeasures, none of the pilots came to the right decision (fly back to Blagnac): they all persevered at trying to land at Franczal. Four of them hit the ground, and the other three had a "chance landing", which means that while they were flying round Franczal, the runway appeared between two fog banks and

they succeeded in achieving a quick landing. During the debriefing, all of them admitted they had come to an erroneous and dangerous decision.

- Context 2: with countermeasures

12 pilots tested scenario 1 or 2 with countermeasures (see next table). “Circuits” corresponds to the number of circuits performed by the pilot round Francazal before a countermeasures is triggered by the wizard of Oz.

Pilot	Scenario	Circuits	Results
Pilot8	1	3	crash on Francazal
Pilot9	1	2	back to Blagnac
Pilot10	1	2	back to Blagnac
Pilot11	1	2	back to Blagnac
Pilot12	2	3	back to Blagnac
Pilot13	2	2	back to Blagnac
Pilot14	2	2	back to Blagnac
Pilot15	2	2	back to Blagnac
Pilot16	1	2	chance landing Francazal
Pilot17	1	2	back to Blagnac
Pilot18	1	2	back to Blagnac
Pilot19	1	5	crash on Francazal

The results show the efficiency of the countermeasures to cure perseveration: 9 pilots out of 12 changed their minds thanks to the countermeasures, and flew back safely to Blagnac. During the debriefing, all the pilots confirmed that the countermeasures were immediately responsible for their change of mind. Moreover, the short disappearance of the data due to the countermeasures did not cause any stress to them. 4 military pilots found that the solutions proposed by the countermeasures were close to what a human co-pilot would have proposed.

The results with Pilot19 suggest that the more a pilot perseveres, the more difficult it is to get him out of perseveration. During the debriefing, Pilot19 told us that he was obsessed by the idea of landing, and that he became more and more stressed as long as he was flying round Francazal. He then declared that he did not notice any countermeasures. Pilot16 and Pilot8 also persevered despite the countermeasures: they declared they knew they were not flying properly and that they had done that on purpose because they wanted to test the flight simulator.

Results for scenario 3: “failure”

Only 2 pilots tested scenario 3. One pilot experimented this scenario without any countermeasures: he did not notice the failure and hit the ground. The second pilot was warned through the countermeasures that he had a failure and that there was a landing ground at vector 80: the pilot immediately performed an emergency landing on this landing ground. Other experiments are currently being conducted with this scenario.

Other results

During the experiments, 9 pilots made errors, e.g. selection of a wrong radio frequency, erroneous altitude, omission to retract flaps and landing gear. To warn them, the wizard of Oz blinked the display on which they were focusing and displayed the error (e.g.: “gear still down”). In each case,

the countermeasure was successful: the pilots performed the correct action at once. During the debriefing, the pilots declared that this kind of alarm was very interesting, and much more efficient and stressless than a classical audio or visual alarm because they could identify at once what the problem was.

These experiments have shown also the significance of the message contents on the blinked display. For example, as far as the erroneous landing at Francazal is concerned, the initial message was “Don’t land”, but the pilots did not take it into account, thinking it was a bug of the simulator. When the message was changed to “Fly back to Blagnac” or “Immediate overshoot” it was understood and taken into account at once.

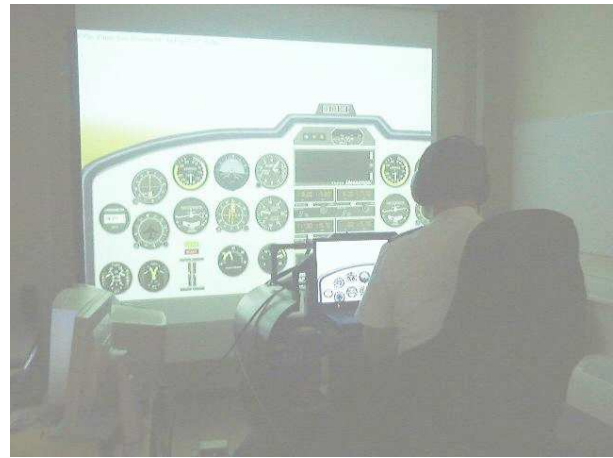


Figure 3: GHOST

5 Conclusion and perspectives

We have presented an approach to detect and cure conflicts in aircraft pilots’ activities. A first experiment has validated the conflict detection tool and has shown that pilots facing conflicts have a trend to persevere in erroneous behaviors. This agrees with the observations made by experts of the pilots’ behaviors in real air accidents. These first results have led us to design GHOST, an experimental environment dedicated to test countermeasures designed to get pilots out of perseveration. A second experiment was conducted with twenty-one pilots in this new experimental environment: it has proved the efficiency of the countermeasures to cure perseveration and has also confirmed the relevance of conflict detection. Further experiments are to be conducted to go on tuning the countermeasures according to the pilots’ feedback.

The next step is to design and implement the whole countermeasures closed-loop, i.e to remove the wizard of Oz and perform an automated conflict and perseveration management. This is currently done thanks to (see figure 4):

- the conflict detection tool (see section 3).
- Kalmansymbo(aéro), a situation tracker and predictor [Tessier, 2003] which is based on predicate/transition timed Petri nets for procedure and pilot’s activity modeling. Kalmansymbo assesses the current situation thanks to the pa-

rameters received in real-time from Flightgear and predicts what is likely to happen. Both the current situation and the predictions are inputs for the conflict detection tool, which allows possible conflicts to be better anticipated.

- a tool for building and sending accurate countermeasures back to the pilot via the Flightgear cockpit.

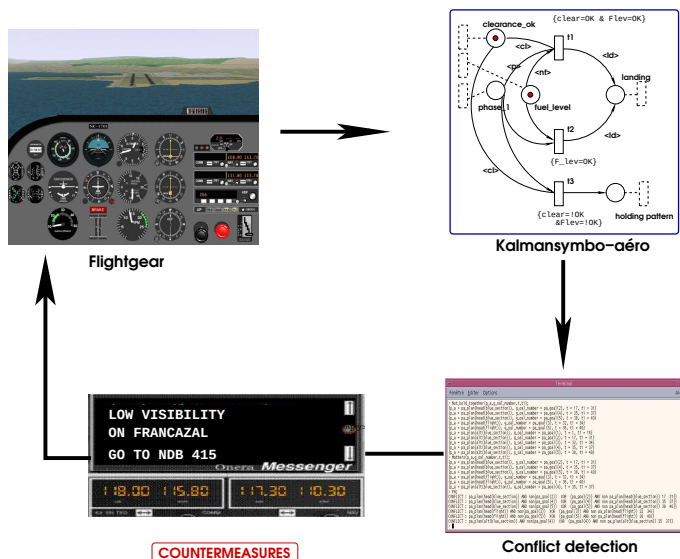


Figure 4: The countermeasures closed-loop

Acknowledgments

Many thanks to Olivier Grisel and Fabrice Schwach for their wonderful work on Flightgear and the wizard of Oz.

References

- [Amalberti and Wioland, 1997] R. Amalberti and L. Wioland. Human error in aviation. In *Aviation safety conference*, Rotterdam, August 1997.
- [Beauvois and Joule, 1999] J.L. Beauvois and R.V. Joule. A radical point of view on dissonance theory. In Eddie Harmon-Jones and Judson Mills, editors, *Cognitive Dissonance : Progress on a Pivotal Theory in Social Psychology*. 1999.
- [Castelfranchi, 2000] C. Castelfranchi. Conflict ontology. In H.-J. Müller and R. Dieng, editors, *Computational conflicts - Conflict modelling for distributed intelligent systems*, pages 21–40. Springer Verlag, 2000.
- [Chaudron et al., 1997] L. Chaudron, C. Cossart, N. Maille, and C. Tessier. A purely symbolic model for dynamic scene interpretation. *International Journal on Artificial Intelligence Tools*, 6(4):635–664, 1997.
- [Chaudron et al., 2003] L. Chaudron, N. Maille, and M. Boyer. The CUBE lattice model and its applications. *Applied Artificial Intelligence*, 17(3):207–242, March 2003.
- [Conry et al., 1991] S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on SMC*, 21(6):1462–1477, 1991.
- [De Kleer, 1986] J. De Kleer. An assumption-based truth maintenance system. *Artificial intelligence*, 28, 1986.
- [Dehais, 2002] F. Dehais. Modelling cognitive conflict in pilot’s activity. In *STAIRS 2002: Starting Artificial Intelligence Researchers Symposium*, pages 45–54, July 2002.
- [Easterbrook, 1991] S. M. Easterbrook. Handling conflict between domain descriptions with computer supported negotiation. *Knowledge Acquisition: An International Journal*, 3(4):255–289, 1991.
- [Festinger, 1957] L. Festinger. *A theory of cognitive dissonance*. CA : Stanford University Press, 1957.
- [Ghallab, 1996] M. Ghallab. On chronicles: representation, on-line recognition and learning. In L.C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the Fifth National Conference on Artificial Intelligence*, San Francisco, California, 1996.
- [Hannebauer, 2000] M. Hannebauer. Their problems are my problems, the transition between internal and external conflicts. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting agents - Conflict management in multi-agent systems*, Multiagent systems, artificial societies and simulated organizations 1, pages 63–109. Kluwer Academic Publishers, 2000.
- [Jung and Tambe, 2000] H. Jung and M. Tambe. Conflicts in agent teams. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting agents - Conflict management in multi-agent systems*, Multiagent systems, artificial societies and simulated organizations 1, pages 153–167. Kluwer Academic Publishers, 2000.
- [Khedro and Genesereth, 1994] T. Khedro and M. Genesereth. Modeling multiagent cooperation as distributed constraint satisfaction problem solving. In *11th ECAI*, pages 249–253, 1994.
- [Lewin et al., 1939] K. Lewin, R. Lippitt, and R. K. White. Patterns of aggression behavior in experimentally created “social climates”. *Journal of Social Psychology*, 10:271–299, 1939.
- [Pastor, 2000] J. Pastor. Cognitive performance modeling - Assessing reasoning with the EARTH methodology. In *COOP’2000*, Sophia Antipolis, May 2000.
- [Rosenschein and Zlotkin, 1994] J. S. Rosenschein and G. Zlotkin. *Rules of encounter. Designing convention for automated negotiation among computers*. Artificial Intelligence. MIT Press, 1994.
- [Rota and Thonnat, 2000] N. Rota and M. Thonnat. Activity recognition from video sequences using declarative models. In *ECAI’00*, pages 673–677, Berlin, 2000.
- [Sycara, 1989] K. Sycara. Multiagent compromise via negotiation. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 119–137. Pitman Publishing, 1989.
- [Tessier et al., 2000] C. Tessier, L. Chaudron, H. Fiorino, and H.-J. Müller. Agents’ conflicts: new issues. In C. Tessier, L. Chaudron, and H.-J. Müller, editors, *Conflicting agents - Conflict management in multi-agent systems*, Multiagent systems, artificial societies and simulated organizations 1, pages 1–30. Kluwer Academic Publishers, 2000.
- [Tessier, 2003] C. Tessier. Towards a commonsense estimator for activity tracking. In *AAAI Spring symposium*, Stanford University, Palo Alto CA, USA, March 2003.
- [Traum and Allen, 1994] D.R. Traum and J.F. Allen. Discourse obligations in dialogue processing. In James Pustejovsky, editor, *Proceedings of the Thirty-Second Meeting of the Association for Computational Linguistics*, pages 1–8, San Francisco, 1994. Morgan Kaufmann.
- [Vand Der Kolk, 1994] B. Vand Der Kolk. The body keeps the score: memory and the evolving psychobiology of post traumatic stress disorder. *Harvard Review of Psychiatry*, 1:253–265, 1994.

Dynamic Bayesian modeling of the cerebral activity

Vincent Labatut, Josette Pastor, Serge Ruff

Inserm u455

Pavillon Riser, CHU Purpan

31059 Toulouse CEDEX 3, France

{vincent.labatut; josette.pastor; serge.ruff}@toulouse.inserm.fr

Abstract

Conventional methods used for the interpretation of activation data provided by functional neuroimaging techniques provide useful insights on what the networks of cerebral structures are, and when and how much they activate. However, they do not explain how the activation of these large-scale networks derives from the cerebral information processing mechanisms involved in cognitive functions. At this global level of representation, the human brain can be considered as a dynamic biological system. Dynamic Bayesian networks seem currently the most promising modeling paradigm. Our modeling approach is based on the anatomical connectivity of cerebral regions, the information processing within cerebral areas and the causal influences that connected regions exert on each other. The capabilities of the formalism's current version are illustrated by the modeling of a phonemic categorization process, explaining the different cerebral activations in normal and dyslexic subjects. The simulation data are compared to experimental results [Ruff *et al.*, 2001].

1 Introduction

In Neurology and Neuropsychology, the diagnosis of the neurological causes of cognitive disorders, as well as the understanding and the prediction of the clinical outcomes of focal or degenerative cerebral lesions, necessitate knowing the link between brain and mind, that is what the cerebral substratum of a cognitive or a sensorimotor function is and how the substratum's activity can be interpreted in cognitive terms, i.e. in terms of information processing.

Studies in humans and animals [Bressler, 1995; Démonet *et al.*, 1994] have shown that sensorimotor or cognitive functions are the offspring of the activity of oriented large-scale networks of anatomically connected cerebral regions (Figure 1). In humans, functional neuroimaging techniques provide activation data, which are indirect measures of the brain's electrical or metabolic

activity during a task performance. Statistical analyses of the activation data allow determining *where* [Fox and Raichle, 1985], i.e. in which areas, and/or *when* [Giard *et al.*, 1995] during the task performance, the activation reaches local extrema. Through the study of covariation between local activations, they give a sketch of *what* the network of cerebral areas involved in the cognitive function is [Herbster *et al.*, 1996]. A known oriented anatomical link between 2 areas allows determining *why* the activation of one area can affect the other one [Büchel and Friston, 1997]. Above methods allow identifying the substratum of a cognitive function and the activation level and dynamics of the substratum during the function performance. They do not give any clue of *how* the cognitive processes participating in the function are implemented by the substratum and *how* the activation derives from the processing. That is, they do not allow *interpreting* neuroimaging data as the result of information processing at the integrated level of large-scale networks.

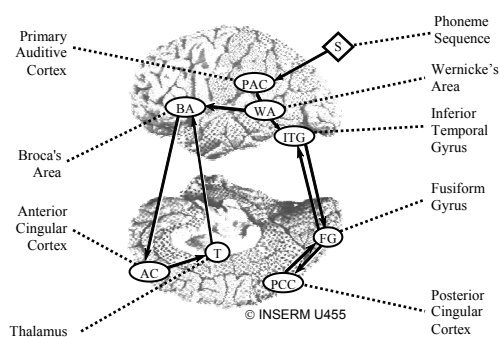


Figure 1: Large-scale network involved in phoneme monitoring, according to results from [Démonet, *et al.*, 1994]

Interpretative models, linking a networked structure activity to the realization of a function, are at the core of Computational Neurosciences. Most existing works in the domain are based on formal neural networks, with varying levels of biological plausibility, from physiology [Wang and Buzsaki, 1996], hardly interpretable in terms of information processing, to more or less biologically

plausible models of how basic cognitive functions emerge from neuronal activation [Grossberg *et al.*, 2002], and to purely functional models [Cohen *et al.*, 1990], not concerned with cerebral plausibility. Although these models answer the *how*, they do not meet two major requirements for an interpretative approach of functional neuroimaging data. The models are not explicit enough to be directly used for clinical purpose, and they cannot evolve quickly and easily with new findings in neuroscience, such as the integration of more detailed knowledge on the substratum, which often necessitates a complete rebuilding of the formal network.

The causal connectivity approach [Pastor *et al.*, 2000] aims at answering the *how* and satisfying the constraints. However, the underlying formalism, causal qualitative networks based on interval calculus, limits severely the biological plausibility of the models, since it cannot represent major cerebral features, such as learning or the non-linearity and the uncertainty of cerebral processes. Dynamic Bayesian networks only meet the three major constraints: temporal evolution, uncertainty and nonlinearity [Labatut and Pastor, 2001]. The utility of graphical probabilistic formalisms for cognitive modeling has also been demonstrated in the representation of visuomotor mechanisms with Bayesian networks [Ghahramani and Wolpert, 1997].

Hereafter, we describe how the interpretation of functional images for a clinical purpose can be tackled. Section 2 presents our viewpoint on large-scale cerebral networks. After a short introduction to dynamic Bayesian networks, section 3 describes the characteristics of our formalism. Section 4 illustrates the formalism's capabilities by an example. We conclude with some perspectives.

2 Representation of Large-Scale Cerebral Networks

2.1 Structural and Functional Nodes

The function implemented by a large-scale network depends on three properties: the network's structure [Goldman-Rakic, 1988], the functional role of each node (E.g. Wernicke's area (Figure 1), which is supposed to realize the early stages of phoneme processing), and the properties of the links (length, role: inhibitory or excitatory, ...). In each network, regions, which are the *structural nodes*, are information processors and connecting oriented fibers are information transmitters [Leiner and Leiner, 1997].

All neurons in a region do not have the same structure or the same role. Similar neurons constitute generally local populations that realize a specific function. For example, the inhibitory role of GABAergic neurons on other neuronal populations may explain the fact that every visual stimulus is not perceived in high frequency stimulation [Pastor, *et al.*, 2000]. Therefore, each region is itself a network of smaller neuronal populations (*functional nodes*), connected through neuronal fibers. These

nodes are information processors that implement *functional primitives*, which may all be different.

A large-scale network has therefore neurophysiologically constrained, oriented edges and possibly differentiated nodes. The explicit representation of the nodes' function allows the direct expression of hypotheses on cerebral processing, and their easy modification in order to follow the evolution of knowledge in neurosciences. This cannot be dealt with by formal neural networks' implicit modeling that requires modifying the whole network architecture to implement functional changes. Hereafter, a structural or a functional structure will be indifferently named a *cerebral zone*.

2.2 Information Representation and Processing

The cerebral information processed by a neuronal population can be seen as the abstraction of the number and the pattern of the neurons firing for this information. It can be represented both by an energy level and by a category. Energy is indirectly represented by the imprecise activation data provided by neuroimaging techniques. The category representation is in agreement with the "topical" organization of the brain, which reflects category maps of the input stimuli, and can persist from primary cortices to nonprimary cortices and subcortical structures [Alexander *et al.*, 1992], through transmission fibers [Leiner and Leiner, 1997]. The energy and the category of a stimulus can also be easily extracted from its psychophysical properties.

Modeling cerebral processes necessitates an explicit and discrete representation of time, both for taking into account the dynamics of cerebral mechanisms (transmission delays, response times...), and for complying with sampled functional neuroimaging data.

According to a definition of causality inspired by Hume [Hume, 1740] and consistent with Pearl's probabilistic causality [Pearl, 2001], information processing in a large-scale network can be considered as mediated through *causal mechanisms*. Causality is defined by three properties: spatial and temporal contiguity, temporal consistency, and statistical regularity [Labatut and Pastor, 2001]. In other words, two entities A and B are causally linked if they are contiguous relatively to the system they belong to, if the beginning of A precedes temporally the beginning of B, and if most of the times, A provokes B. In the brain, oriented anatomical links provide spatial and temporal contiguity between cerebral nodes, cerebral events are temporally consistent (a firing zone provokes the activation of downstream zones), and there is a statistical regularity in the response of a specific neuronal population to a given stimulus.

3 Description of the Formalism

3.1 Dynamic Bayesian Networks

In summary, the brain can be viewed as a network whose nodes are differentiated dynamic and adaptive informa-

tion processors and oriented edges convey causality. Moreover, cerebral mechanisms, which are the abstraction, at the level of a neuronal population, of the chemical and electrical mechanisms at the cell levels, are often nonlinear. Causal dynamic Bayesian networks are the paradigm that meets best the constraints derived from these properties [Labatut and Pastor, 2001].

A causal Bayesian network consists of a directed acyclic graph where nodes represent random variables and edges represent causal relationships between the variables [Pearl, 1988]. A conditional probability is associated with each relationship between a node and its parents. If the node is a root, the probability distribution is a prior. When some nodes' values are observed, posterior probabilities for the hidden nodes can be computed thanks to inference algorithms such as the junction tree algorithm [Jensen, 1996].

In a dynamic Bayesian network (DBN), the evolution of random variables through time is considered. Time is seen as a series of intervals called time slices [Dean and Kanazawa, 1988]. For each slice, a submodel represents the state of the modeled system. DBNs are used to model Markovian processes, i.e. processes where a temporally limited knowledge of the past is sufficient to predict the future. The choice of the inference algorithm, generally an extension of the junction tree algorithm [Murphy, 1999], depends on the DBN's structure, the nature of its variables (discrete or continuous), and relationships (linear or nonlinear).

Activation data and/or the subject's responses to the stimuli are the only observable variables we have. Therefore, they must be integrated in our models. One may reasonably consider that the hidden variables, describing the successive states of the cerebral network, constitute a Markov chain, and that observable variables depend only on them. Moreover, the variables are continuous and their relationships may be nonlinear. This is typically the description of a type of DBNs called fully nonlinear state space models. Specific and recent algorithms allowing dealing with nonlinearity exist for this type of structures. Their general principle is to linearize the model in order to apply the classic Kalman filter. These algorithms differ on the used linearization method: first-order Taylor approximations for the extended Kalman filter [Julier and Uhlmann, 1997; Norgaard *et al.*, 2000] or polynomial approximations for the unscented Kalman filter [Julier and Uhlmann, 1997], the divided difference filter (DDF) [Norgaard, *et al.*, 2000], and others [Van Der Merwe and Wan, 2001]. The algorithms based on polynomial approximations seem to give more reliable results [Norgaard, *et al.*, 2000]. Their computational complexity is $O(L^3)$, where L is the state dimension [Van Der Merwe and Wan, 2001]. They offer equivalent qualities, but those of the DDF are more accurate according to its author [Norgaard, *et al.*, 2000].

3.2 Formal definition

Static and Dynamic Networks

A *static network* is the graphical representation of a large-scale network, whose nodes are cerebral zones and edges are the oriented axon bundles connecting zones. Due to anatomical loops, it is often cyclic. The DBN is the acyclic temporal expansion of the static network. Each node of the DBN is the *processing entity* related to a cerebral zone, i.e. the mathematical expression, at a given time slice, of information processing in the zone. Each edge is the *propagation entity*, whose orientation is its corresponding axon bundle's orientation. When deriving the DBN from the static network, values are given to the temporal parameters, according to known physiology results (e.g. the transmission speed in some neural fibers). That is, the length of the time slices is fixed, and a delay representing the average propagation time in the bundle's fibers is associated to the propagation entity.

Information Representation

Cerebral information is the *flowing entity* that is computed at each spatial (cerebral zone) and temporal (time slice) step, by a processing entity. It is a two-dimensional data. The first part, the *magnitude*, stands for the cerebral energy needed to process the information in the zone. It is represented by a real random variable in the DBN. For the second part, the type, which represents the cerebral category the zone attributes to the information, the representation is based on the *symbol* and *categorical field* concepts.

A *symbol* represents a "pure" (i.e. not blurred with noise or another symbol) category of information. For example, when the information represents a linguistic stimulus, a symbol may refer to a non ambiguous phoneme. For cerebral information, the symbol represents, in each zone, the neuronal subpopulation being sensitive to (i.e. that fires for) the corresponding category. It may be, in the primary auditory cortex, the subpopulation sensitive to a specific frequency interval. A *categorical field* is a set of symbols describing stimuli of the same semantic class. The "color" categorical field contains all the color symbols, but it cannot contain phonemes.

A type concerns several symbols, due to the presence of noise or because of some compound information. Let $\mathbb{S}$ be the set of all existing symbols. We assume that a type T is defined for only one categorical field. Let S_T be the subset of $\mathbb{S}$, corresponding to this categorical field. The type T is an application from S_T to $[0,1]$, with the property $\sum_{s \in S_T} T(s) = 1$, i.e. it describes a symbol repartition for a specific categorical field.

In a stimulus, this repartition corresponds to the relative importance of each symbol compounding the information carried by the stimulus. Inside the model, $T(s)$ stands for the proportion of s -sensitive neurons in the population that fired for the information whose type is T . Unlike the magnitude, the

type is not represented by a random variable. Indeed, it is not necessary to represent its uncertainty (and hence to make the computational complexity harder) since we cannot compare it to neuroimaging data.

At time t and node X , the information is represented by the type T_X^t and the magnitude M_X^t at the output of X .

Propagation and Processing

For a zone X , both the cerebral propagation mechanisms (i.e. the relationships towards the zone) and the processing (spatial and temporal integration of the inputs, and processing as such) are described by a pair of functions, the type f_{T_X} and the magnitude functions f_{M_X} . In the general case where n zones $Y_1, \dots, Y_n$ are inputs to X , let $\partial_1, \dots, \partial_n$ be the corresponding delays of these relationships. In the DBN, the general form of the magnitude functions is:

$$M_X^t = f_{M_X}(M_{Y_1}^{t-\partial_1}, \dots, M_{Y_n}^{t-\partial_n}, M_X^{t-1}, u_X) \quad (1)$$

where f_M can be a nonlinear function. The random variable $u_X \sim N(0, \sigma^2)$ models uncertainty in the cerebral processing.

The type function is any combination of the incoming types and of the previous type that respects our type definition. If all types are defined on the same categorical field S , the type function can be the linear combination:

$$T_X^t(s) = c_{Y_1} T_{Y_1}^{t-\partial_1}(s) + \dots + c_{Y_n} T_{Y_n}^{t-\partial_n}(s) + c_X T_X^{t-1}(s), \quad \forall s \in S \quad (2)$$

where S is the categorical field of T_X^t , $T_{Y_1}^{t-\partial_1}, \dots, T_{Y_n}^{t-\partial_n}$, and T_X^{t-1} ; and with $\sum c_i = 1$ in order to keep the property

$$\sum_{s \in S} T_X^t(s) = 1.$$

The functions' definition, as well as the setting of the parameters' values (e.g. the value of a firing threshold), utilize mostly results in neuropsychology or in neurophysiology. The existence of *generic models*, that is, non instantiated, *reusable*, models of functional networks, is assumed. For example, primary cortices may implement the same mechanisms, although they are parameterized so that they can process different types of stimuli [Pastor, *et al.*, 2000].

4 Example

The model, presented hereafter, is based on an experimental study [Ruff, *et al.*, 2001] that focused on the differences between normal and dyslexic subjects during a passive phonemic categorization process.

Six patients and six controls were submitted to a passive hearing of stimuli that are mixes of the two phonetically close syllables /pa/ and /ta/. The *pivot* is noted *dev0* and the *deviants* are 4 different mixes of /pa/ and /ta/, noted *dev2M*, *dev1M*, *dev1P*, *dev2P* (Table 1). The measurements were made with fMRI. An experiment is constituted of 5 blocks, corresponding to the pivot and the de-

viant. Each block contains 6 sequences of 4 sounds, 3 pivots and the block's deviant, in a random order.

We focus on a single region, a part of the right temporal superior gyrus involved in the early processing of auditory stimuli and activated differently in controls and dyslexic subjects. Phylogeny is in favor of the existence of specialized phonemic processors in this area (Figure 2). Since their location is unknown, they cannot constitute separate structural nodes. They are supposed to have the same building functional nodes. According to our genericity hypothesis, the processors' structure and parameters are based on a previously released visual cortex model [Pastor, *et al.*, 2000]. The Input Gating Nodes (*IGN*•) express the phoneme processors' sensitivity to the stimulus. The Output Gating Nodes (*OGN*•) send information to the downstream areas. Intra and inter (lateral) inhibitions (*IN*• and *LIN*•) are assumed between the /pa/ and /ta/ processors. *LIN*• make the activation of an *IGN*• cause an inhibition in the opposite *IGN*•. Each Firing Threshold Node (*FTN*•) is modulated by an *OGN*• that can lower it. Since only one activation measure is provided by fMRI for the area, it is represented by the sole *AN* node in the static model. *Stim* stands for the stimulus.

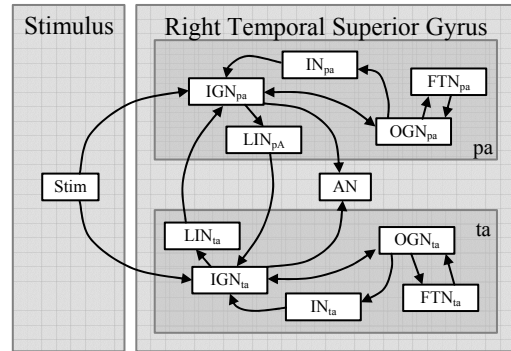


Figure 2: Static network used to model the cerebral phonemic categorization process.

Except for the parameterization of the *IGN*• nodes, which reflects the specialization of each phonemic processor to the phoneme category (/pa/ or /ta/), the functions for both the /pa/ and /ta/ parts share exactly the same structure and parameters. Thus, only the *pa* part will be presented. In the following equations, the i^{th} parameter of the function of a node X is noted $a_X^{(i)}$.

The refractory period of the processor's neurons is modeled in *IGN*_{pa} by a sigmoid function ϕ_{IGN} that makes the node sensitive to the incoming stimulus only if the magnitude of the output is already close to zero:

$$M_{IGN_{pa}}^t = a_{IGN_{pa}}^{(1)} g_{IGN_{pa}}(T_{Stim}^{t-2} (1 - \phi_{IGN}(M_{OGN_{pa}}^{t-1}))) M_{Stim_{pa}}^{t-2} + a_{IGN_{pa}}^{(2)} M_{IGN_{pa}}^{t-1} - a_{IGN_{pa}}^{(3)} M_{IN_{pa}}^{t-1} - a_{IGN_{pa}}^{(4)} M_{LIN_{ta}}^{t-1} + u_{IGN_{pa}}^t \quad (3)$$

The categorical field contains two symbols (*pa* and *ta*). The type of a stimulus represents the proportions of the two symbols (Table 1).

name	dev2M	dev1M	dev0	dev1P	dev2P	sens _{pa}	sens _{ta}
pa value	0.7	0.55	0.4	0.25	0.1	0.8	0.2
ta value	0.3	0.45	0.6	0.75	0.9	0.2	0.8

Table 1: Constants for both phonemic categorization models

The sensitivity of each $IGN_{\bullet}$ to the received type is defined by a constant type $sens_{\bullet}$. IGN_{pa} is more sensitive to the symbol pa , and IGN_{ta} to ta . The function $g_{IGN_{pa}}$ in equation (3) is used with the constant $sens_{pa}$ and the incoming stimulus' type T_{Stim}^{t-2} in order to modulate the magnitude of IGN_{pa} :

$$g_{IGN_{pa}}(T_{Stim}^{t-2}) = T_{Stim}^{t-2}(pa)sens_{pa}(pa) + T_{Stim}^{t-2}(ta)sens_{pa}(ta)$$

The types are used only for the input gating; they do not intervene in the rest of the model. The sigmoid ϕ_{OGN} in OGN_{pa} 's magnitude function allows it to fire only if the magnitude coming from the IGN_{pa} is greater than the firing threshold's (FTN_{pa}) one:

$$M'_{OGN_{pa}} = a_{OGN}^{(1)} \phi_{OGN}(M'_{IGN_{pa}} - M'_{FTN_{pa}}) M'_{IGN_{pa}}^{t-1} + a_{OGN}^{(2)} M'_{OGN_{pa}}^{t-1} + u'_{OGN_{pa}} \quad (4)$$

$$M'_{IN_{pa}} = a_{IN}^{(1)} M'_{OGN_{pa}}^{t-1} + a_{IN}^{(2)} M'_{IN_{pa}}^{t-1} + u'_{IN_{pa}} \quad (5)$$

$$M'_{LIN_{pa}} = a_{LIN}^{(1)} M'_{IGN_{pa}}^{t-1} + a_{LIN}^{(2)} M'_{LIN_{pa}}^{t-1} + u'_{LIN_{pa}} \quad (6)$$

$$M'_{FTN_{pa}} = a_{FTN}^{(1)} - (a_{FTN}^{(2)} (a_{FTN}^{(1)} - M'_{FTN_{pa}}^{t-1})) + a_{FTN}^{(3)} M'_{OGN_{pa}}^{t-2} + u'_{FTN_{pa}} \quad (7)$$

AN consists in the sum of the successive $IGNs$ ' activations during one experimental block:

$$M'_{AN} = M'_{IGN_{pa}}^{t-1} + M'_{IGN_{ta}}^{t-1} + M'_{AN}^{t-1} \quad (8)$$

This is a gross approximation of the fMRI data, which models only the part of the information processing mechanisms in the activation building and neglects metabolic processes at the level of the cerebral blood flow. Since, except the Stim and the AN nodes, all nodes represent neuronal activities, the time unit is set to 1 ms. We used the DD2 algorithm [Norgaard, *et al.*, 2000] to perform the simulations.

The hypothesis is that the difference of processing between normal and dyslexic subjects is caused by a disorder in the inhibitory mechanisms. Thus, the two models, one for the average patient and the other for the average control, use the same functions and share the same parameters, except for the inhibition nodes ($IN_{\bullet}$ and $LIN_{\bullet}$). There are no lateral inhibitions in the dyslexic model. It can be interpreted in cognitive terms as the fact that all the processors compete for each stimulus and that no clear category can be built. Also, the dyslexic model's internal inhibitions are slightly stronger than in the normal one, leading to a slowing in the stimulus perception. These two tentative interpretations are good starting points for new experiments.

The differences in the inhibition parameters are sufficient to obtain very different activation data. For con-

trols, the more distant (from the pivotal stimulus, categorically speaking) the deviant is, the stronger the activation is (Figure 3). This is supposed to be caused by a habituation mechanism that lowers the activation, followed by an activation the force of which depends on the "surprise" caused by the deviant. Dyslexic subjects do not correctly categorize the different phonemes, both the pa and the ta parts of the gyrus activate for each block. This illustrates how activation data can be explained thanks to the understanding of the cerebral information processing mechanisms expressed in the models.

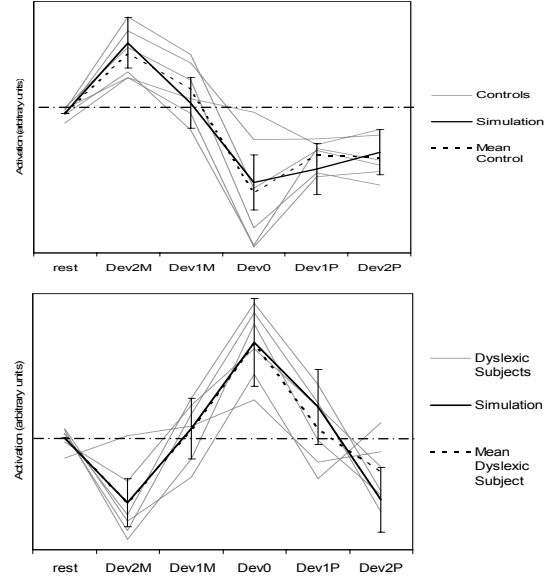


Figure 3: Compared results between simulated data (± 2 standard deviations) and experimental measures.

5 Conclusion

Instead of building a specialized model, designed for a specific function or cerebral network, we have presented a general framework, allowing the interpretation of functional neuroimaging data. This framework has been designed to be open to evolutions of the knowledge in neuropsychology and neurophysiology. Using DBNs allows modeling the brain as a dynamic causal probabilistic network with nonlinear relationships. We have illustrated this with an example concerning a language-related process. Currently, our framework is adapted to automatic processing, which is dominant in cerebral functioning. In function of the stimulus type, nodes can react differently and different networks may be activated, thus implementing different functions. Our future work will focus on the integration of more biological plausibility in the framework. The representation of complex relationships between and inside the zones will allow the representation of controlled processes and contextual modulation of the cerebral activity. The combination of types from different categorical domains and the search for regularities in the combinations will allow the implementation of learning mechanisms. Another essential topic is to make our mod-

els independent of the used data acquisition technique, thanks to *interface* models, able to translate cerebral information processing variables into neuroimaging results. Our long-term goal is to progressively include in our framework various validated models and to build a consistent and general brain theory based on large-scale networks.

References

- [Alexander, *et al.*, 1992] G. E. Alexander, M. R. Delong, and M. D. Crutcher. Do cortical and basal ganglionic motor area use "motor programs" to control movement? *BBS*, 15:656-65, 1992.
- [Bressler, 1995] S. L. Bressler. Large-scale cortical networks and cognition. *Brain Res Rev*, 20(3):288-304, 1995.
- [Büchel and Friston, 1997] C. Büchel and K. J. Friston. Modulation of connectivity in visual pathways by attention: Cortical interactions evaluated with structural equation modelling and fMRI. *Cereb Cortex*, 7(8):768-78, 1997.
- [Cohen, *et al.*, 1990] J. D. Cohen, K. Dunbar, and J. L. McClelland. On the control of automatic processes: A parallel distributed processing account of the Stroop effect. *Psychol Rev*, 97(3):332-61, 1990.
- [Dean and Kanazawa, 1988] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. Proceedings of AAAI, Pages 524-528, 1988.
- [Démonet, *et al.*, 1994] J. F. Démonet, C. Price, R. Wise, and R. S. Frackowiak. A PET study of cognitive strategies in normal subjects during language tasks. Influence of phonetic ambiguity and sequence processing on phoneme monitoring. *Brain*, 117 (Pt 4):671-82, 1994.
- [Fox and Raichle, 1985] P. T. Fox and M. E. Raichle. Stimulus rate determines regional brain blood flow in striate cortex. *Ann Neurol*, 17(3):303-5, 1985.
- [Ghahramani and Wolpert, 1997] Z. Ghahramani and D. M. Wolpert. Modular decomposition in visuomotor learning. *Nature*, 386(6623):392-395, 1997.
- [Giard, *et al.*, 1995] M.H. Giard, J. Lavikainen, K. Reinikainen, F. Perrin, O. Bertrand, J. Pernier, and R. Näätänen. Separate representation of stimulus frequency, intensity and duration in auditory sensory memory: An event-related potential and dipole-model analysis. *J Cogn Neurosci*, 7(2):133-143, 1995.
- [Goldman-Rakic, 1988] P. S. Goldman-Rakic. Topography of cognition: Parallel distributed networks in primate association cortex. *Annu Rev Neurosci*, 11:137-56, 1988.
- [Grossberg, *et al.*, 2002] S. Grossberg, S. Hwang, and E. Mingolla. Thalamocortical dynamics of the McCollough effect: Boundary-surface alignment through perceptual learning. *Vision Res*, 42(10):1259-86, 2002.
- [Herbster, *et al.*, 1996] A. N. Herbster, T. Nichols, M. B. Wiseman, M. A. Mintun, S. T. DeKosky, and J. T. Becker. Functional connectivity in auditory-verbal short-term memory in Alzheimer's disease. *Neuroimage*, 4(2):67-77, 1996.
- [Hume, 1740] D. Hume. *Treatise of human nature*, Oxford University Press (2000), 1740.
- [Jensen, 1996] Finn V. Jensen. *An introduction to Bayesian networks*, Springer, New York, 1996.
- [Julier and Uhlmann, 1997] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. Proceedings of Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando (FL), 1997.
- [Labatut and Pastor, 2001] V. Labatut and J. Pastor. Bayesian modeling of cerebral information processing. Proceedings of AIME, Bayesian models in medicine Workshop, Cascais (Po), 2001.
- [Leiner and Leiner, 1997] H. C. Leiner and A. L. Leiner. How fibers subserve computing capabilities: Similarities between brains and machines. *Int Rev Neurobiol*, 41:535-53, 1997.
- [Murphy, 1999] K. Murphy. Filtering, smoothing and the junction tree algorithm. Technical Report, U.C. Berkeley, Dept. Comp. Sci, 1999.
- [Norgaard, *et al.*, 2000] Magnus Norgaard, Niels K. Poulsen, and Ole Ravn. Advances in derivative-free state estimation for nonlinear systems. Technical Report IMM-REP-1998-15, Technical University of Denmark, Lyngby, 2000.
- [Pastor, *et al.*, 2000] J. Pastor, M. Lafon, L. Travé-Massuyès, J. F. Démonet, B. Doyon, and P. Celsis. Information processing in large-scale cerebral networks: The causal connectivity approach. *Biol Cybern*, 82(1):49-59, 2000.
- [Pearl, 1988] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufmann, 1988.
- [Pearl, 2001] J. Pearl. Bayesianism and causality, or, why I am only a half-Bayesian. In *Foundations of Bayesianism*, Kluwer, 2001.
- [Ruff, *et al.*, 2001] S. Ruff, K. Boulanouar, D. Cardebat, P. Celsis, and J. F. Demonet. Brain correlates of impaired categorical phonetic perception in adult dyslexics. *Neuroimage*, 13(6):S595-S595, 2001.
- [Van Der Merwe and Wan, 2001] R. van der Merwe and E. A. Wan. Efficient derivative-free Kalman filters for online learning. Proceedings of ESANN, Pages 205-210, Bruges, Belgium, 2001.
- [Wang and Buzsaki, 1996] X. J. Wang and G. Buzsaki. Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model. *J Neurosci*, 16(20):6402-13, 1996.