

Incremental Utility Elicitation with the Minimax Regret Decision Criterion

Tianhan Wang

Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, CANADA
tianhan@cs.toronto.edu

Craig Boutilier

Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, CANADA
cebly@cs.toronto.edu

Abstract

Utility elicitation is a critical function of any automated decision aid, allowing decisions to be tailored to the preferences of a specific user. However, the size and complexity of utility functions often precludes full elicitation, requiring that decisions be made without full utility information. Adopting the *minimax regret* criterion for decision making with incomplete utility information, we describe and empirically compare several new procedures for incremental elicitation of utility functions that attempt to reduce minimax regret with as few questions as possible. Specifically, using the (continuous) space of standard gamble queries, we show that myopically optimal queries can be computed effectively (in polynomial time) for several different improvement criteria. One such criterion, in particular, empirically outperforms the others we examine considerably, and has provable improvement guarantees.

1 Introduction

As software for decision making under uncertainty becomes increasingly common, the process of utility elicitation takes on added importance. Tailoring decisions to a specific user requires knowledge of the user's utility function, information that generally cannot be built into software in advance. In domains as varied as travel planning, product configuration, and resource allocation, to name but a few, assessing a user's utility function is an integral part of interactive decision making. Unfortunately, as is well-known among decision analysts, utility functions are unwieldy, complex, and difficult for users to articulate [5]. To mitigate these difficulties, analysts have developed many techniques for easing the burden of elicitation. For example, structuring of multiattribute utility functions reduces the number of parameters that need to be assessed [6]; and the use of *standard gamble queries* and sensitivity analysis allows users to calibrate utilities more easily [5].

More recently, emphasis has been placed on decision making with incomplete utility information. The principle of maximum expected utility (MEU) cannot be used directly in such cases, since the utility function is unknown; thus

new decision criteria are needed. In addition, methods for the automatic generation of queries have been developed that reduce uncertainty or incompleteness in the utility function with minimal effort. Within AI, probabilistic models of utility function uncertainty have been used [4; 2]. By assuming a density over possible utility functions, expectations over this density can be taken to determine the value of a decision; and standard Bayesian updating techniques can be used to account for the responses to queries. A different perspective is taken in work on imprecisely specified multiattribute utility theory (ISMAUT) [8; 7] in which linear constraints on multiattribute utility functions are refined, allowing the set of Pareto optimal decisions to be identified; these constraints are often refined until one action can be proven optimal. Boutilier, Bacchus and Brafman [3] and Blythe [1] adopt a somewhat related perspective, also reasoning with linear constraints on utility functions.

In this work, we adopt a distribution-free model, working with linear constraints on utility functions, much like ISMAUT. Unlike ISMAUT, we allow for decisions to be made (or recommended) even when the incompleteness in utility knowledge prevents us from proving a decision is optimal. In such circumstances, we adopt the *minimax regret* decision criterion. We also propose and examine several methods for generating queries that reduce regret quickly, in contrast to work on ISMAUT (where query *generation* strategies have not been studied in depth). In this sense, our model more closely resembles probabilistic models [4; 2], which rely on the fact that decisions of good *expected* quality can be made with uncertain utility information. Using the minimax regret criterion, we generate decisions whose quality (difference from optimal) can be bounded in the face of incomplete utility information. These bounds can be traded off against query cost or minimum error requirements to guide the query process.

The paper is organized as follows. We outline relevant background in Section 2 and define the minimax regret criterion for decision making with incomplete utility information. We show how decisions of minimax regret can be computed using simple linear programs (LPs) if utility constraints are linear. We also discuss incremental elicitation, focusing on *standard gamble queries* (SGQs) [5], the responses to which impose one-dimensional, linear utility constraints that can be easily handled using LPs. Our key contribution is described

in Section 3, where we develop several myopic elicitation strategies. Assuming linear constraints in each utility dimension (an assumption consistent with the use of SGQs), we show that the minimax regret improvement offered by any response to a SGQ, as a function of the (continuous) query parameter, is piecewise linear (PWL) and weakly monotonic (decreasing or increasing, depending on the response). This fact allows optimal queries under each query strategy to be computed efficiently, in time linear in the number of utility attributes, despite the fact that query space is continuous. We present empirical results comparing the different strategies in Section 4, demonstrating the effectiveness, in particular, of the *maximum expected improvement* strategy.

2 Minimax Regret with Incomplete Utility Information

We assume a system charged with making a decision on behalf of a user in a specific *decision scenario*. By a decision scenario, we refer to a setting in which a fixed set of choices (e.g., actions, policies, recommendations) are available to the system, and the (possibly stochastic) effects of these choices are known. For example, the decisions could be courses of medical treatment with known probabilities for specific outcomes [4]. The system’s task is to take the optimal decision with respect to the user’s utility function over outcomes, or some approximation thereof. The system may have little information about the user’s utility function, so to achieve this, it must find out enough information about this utility function to enable a good decision to be made. We assume that The system has available to it a set of queries it can ask of the user that provide such information. We make these concepts more precise below.

2.1 The Minimax Criterion

Formally, a *decision scenario* consists of a finite set of possible decisions D , a finite set of n possible outcomes (or states) S , and a distribution function $\Pr_d \in \Delta(S)$, for each $d \in D$.¹ The term $\Pr_d(s)$ denotes the probability of outcome s being realized if the system takes decision d . A utility function $u : S \rightarrow [0, 1]$ associates utility $u(s)$ with each outcome s . We often view u as a n -dimensional vector $\mathbf{u}$ whose i th component u_i is simply $u(s_i)$. We assume that utilities are normalized in the range $[0, 1]$ for convenience. The *expected utility* of decision d with respect to utility function $\mathbf{u}$ is:

$$EU(d, \mathbf{u}) = \sum_{i \in S} \Pr_d(s_i) u_i.$$

Note that $EU(d, \mathbf{u})$ is linear in $\mathbf{u}$. The optimal decision d^* w.r.t. $\mathbf{u}$ is that with *maximum expected utility* (MEU).

In general the utility function $\mathbf{u}$ will not be known with certainty at the start of the elicitation process, nor at its end. As in ISMAUT [8; 7], we model this uncertainty by assuming a set of linear constraints $\mathcal{C}$ over the set of possibly utility functions $U = [0, 1]^n$. More precisely, we assume that constraints

¹The extension of our elicitation methods to a set of possible decision scenarios is straightforward.

over unknown utility values u_i are linear. We use $C \subseteq U$ to denote the subspace of U satisfying $\mathcal{C}$.

If a system makes a decision d under such conditions of incomplete utility information, some new decision criterion must be adopted to rank decisions. Following [3], we adopt the *minimax regret decision criterion*.² Define the optimal decision $d_{\mathbf{u}}^*$ with respect to utility vector $\mathbf{u}$ to be

$$d_{\mathbf{u}}^* = \arg \max_{d_i} EU(d_i, \mathbf{u}).$$

If the utility function were known, $d_{\mathbf{u}}^*$ would be the correct decision. The *regret* of decision d_i with respect to $\mathbf{u}$ is

$$R(d_i, \mathbf{u}) = EU(d_{\mathbf{u}}^*, \mathbf{u}) - EU(d_i, \mathbf{u}).$$

i.e., the loss associated with executing d_i instead of acting optimally. Let $C \subseteq U$ be the feasible utility set. Define the *maximum regret* of decision d_i with respect to C to be

$$MR(d_i, C) = \max_{\mathbf{u} \in C} R(d_i, \mathbf{u})$$

and the decision d_C^* with *minimax regret* with respect to C :

$$d_C^* = \arg \min_{d_i} MR(d_i, C).$$

The (*minimax*) *regret level* of feasible utility set C is

$$MMR(C) = MR(d_C^*, C).$$

If the only information we have about a user’s utility function is that it lies in the set C , then d_C^* is a reasonable decision. Specifically, without distributional information over the set of possible utility functions, choosing (or recommending) d_C^* minimizes the worst case loss with respect to possible realizations of the utility function (e.g., if the true $\mathbf{u}$ were chosen by an adversary).

If C is defined by a set $\mathcal{C}$ of linear constraints, then d_C^* as well as $MMR(C)$ can be computed using a set of linear programs [3]. We can compute the *pairwise max regret*, for any pair of decisions d_i and d_j ,

$$PMR(d_i, d_j, C) = \max_{\mathbf{u} \in C} EU(d_j, \mathbf{u}) - EU(d_i, \mathbf{u})$$

using an LP (i.e., maximizing a linear function of the unknown outcome utilities subject to C). Solving $O(|D|^2)$ such LPs, one for each ordered pair of actions, allows us to identify the decision d_C^* that achieves minimax regret and to determine the minimax regret level $MMR(C)$.

2.2 Incremental Elicitation

Given partial knowledge of a utility function in the form of constraint set $\mathcal{C}$, the optimal decision d_C^* may have an unacceptable level in regret. In such a case, a user could be queried in order to reduce this level of uncertainty, thus generally improving decision quality.³

²Minimax regret is often used for decision making under *strict* or unquantified uncertainty [5], but its application to imprecisely known utility functions appears not to have been considered in the decision analysis literature.

³Note that max regret cannot increase with additional utility information: if $C \subseteq C'$, then $MMR(C) \leq MMR(C')$.

A common type of query is a standard gamble w.r.t. outcome s_i , where the user is asked if she prefers s_i to a gamble in which the best outcome $s_\top$ occurs with probability l and the worst $s_\perp$ occurs with probability $1 - l$ [6]. We will designate this query $q_i(l)$ and focus our attention on such *standard gamble queries* or SGQs.⁴ Given a response *yes* to query $q_i(l)$, the constraint $u_i > l$ can be imposed on the user's utility function, thus (in general) refining our knowledge; similarly, a *no* response corresponds to the constraint $u_i \leq l$. A response to any standard gamble query imposes a one-dimensional (i.e., axis parallel) linear constraint on the utility set. Thus if our initial constraint set $\mathcal{C}$ is linear, computing the minimax optimal decision after a sequence of SGQs can be accomplished using the LP method above. Furthermore, if $\mathcal{C}$ consists of a set of bounds on utility values in each dimension—i.e., $\mathcal{C}$ forms a hyper-rectangle within $[0, 1]^n$ —then after any sequence of SGQs, the feasible utility set retains this form.

The interactive decision making context we consider is one in which queries are asked repeatedly until the minimax regret level falls to some acceptable value. At that point the “optimal” decision, that with minimax regret given the current constraints, is recommended. Termination can be based on simple thresholding, or can take into account the cost of a query (which can be weighed against the predicted improvement in decision quality).⁵ Generally, queries will be asked that offer the greatest predicted improvement in decision quality.

Both query selection and termination rely critically on the way in which “predicted improvement in decision quality” is defined for a query. For example, when asking a query $q_i(l)$ with respect to current constraint set $\mathcal{C}$, we obtain two constraint sets $\mathcal{C}_{no}$ and $\mathcal{C}_{yes}$, respectively, given responses, *no* and *yes*. We might then define the improvement in decision quality associated with the query as some function of $MMR(\mathcal{C}_{no})$ and $MMR(\mathcal{C}_{yes})$. Such a method of evaluating queries is *myopic*: the query is evaluated in isolation, without consideration of its impact on the value or choice of future queries.

It is important to note that optimal query choice is inherently nonmyopic—in general, a sequence of several queries may offer much more value than the aggregate myopic values of the component queries. Unfortunately, nonmyopic methods require some form of lookahead, and thus often impose severe computational costs on the process of query selection. For this reason, we focus on the development of several myopic query selection strategies in the next section. This is analogous to the use of myopic methods for the approximation of value of information in cases where uncertainty is quantified probabilistically; while the computation of true value of information requires some form of sequential rea-

⁴Other types of queries could be considered, though we rely on the special nature of SGQs in some of our results. SGQs are used widely in decision analysis [5], and have been the main query type studied in recent Bayesian elicitation schemes [4; 2].

⁵Of course, elicitation can continue until a zero-regret (i.e., optimal) decision is identified: this occurs whenever $\mathcal{C} \subset \mathcal{R}_d$ for some decision d , the region of utility space for which d is optimal. The regions $\mathcal{R}_d$ are convex polytopes within U .

soning, myopic approximations tend to be used frequently in practice [5; 4].

3 Myopic Elicitation Strategies

In this section we describe three myopic strategies for query selection under the minimax regret criterion. Throughout this section we make the following assumptions:

- (a) The initial constraints have the form of upper and lower bounds in each utility dimension (these may be trivial bounds, 0, 1).
- (b) SGQs are asked, assuming some known best and worst outcome. These ensure that each constraint set (after any query) has the same form as the initial set (i.e., a hyper-rectangle). We denote by ub_i and lb_i the current bounds on the utility u_i of outcome s_i .
- (c) For simplicity, we assume a threshold τ is used to implement termination; that is, when the predicted improvement of a query falls below τ , we terminate the process.

We discuss the impact of relaxing these assumptions later.

3.1 Characterization of Regret Reduction

Query selection is complicated by the fact that, in general, there are n types of SGQs that can be asked—one per outcome s_i —and a continuous set of instances of each type i — $q_i(l)$ for each $l \in [lb_i, ub_i]$. Whatever criterion is used to select queries, it must distinguish among queries in this n -dimensional continuous space.

Before describing the three strategies, we characterize the reduction in minimax regret offered by a response to a SGQ. Assume current constraint set $\mathcal{C}$, with bounds ub_i and lb_i in dimension i . Given query $q_i(l)$, with $lb_i < l < ub_i$, a negative response will provide us with a refined set of constraints $\mathcal{C}_{no} = \mathcal{C} \cup \{u_i \leq l\}$ —i.e., it reduces the upper bound ub_i on u_i —with (ideally) reduced regret. A positive response gives us a similarly refined constraint set $\mathcal{C}_{yes}$.

Focusing on the negative response, note that we'll end up with different $\mathcal{C}_{no}$ sets depending on the query point l . Naturally, the closer the query point l is to lb_i , the more informative a negative response is (since it constrains u_i to be in the interval $[lb_i, l]$). Intuitively, we'd expect minimax regret to be smaller given a tighter query. In fact, we can say more. Define $MMR_{no}(\mathcal{C}, i, l)$ to be $MMR(\mathcal{C}_{no})$ where the *no* is in response to query $q_i(l)$. We have:

Theorem 1 For any $i \leq n$, $MMR_{no}(\mathcal{C}, i, l)$ is a PWL, non-decreasing function of l .

Proof: For space reasons, we provide only an intuitive proof sketch. Define $PMR(d, d', \mathcal{C}, i, l) = PMR(d, d', \mathcal{C} \cup \{u_i = l\})$; that is, $PMR(d, d', \mathcal{C}, i, l)$ is the pairwise max regret of d w.r.t. d' given the current constraints *plus* knowledge that $u_i = l$. It is not hard to see that $PMR(d, d', \mathcal{C}, i, l)$ is a linear function of l : maximizing the regret of d w.r.t. d' is effected by setting each component u_j of the utility function to either its upper bound (if $\Pr_d(s_j) \leq \Pr_{d'}(s_j)$) or lower bound (if $\Pr_d(s_j) > \Pr_{d'}(s_j)$). Since this max is achieved independently in each dimension, the contribution

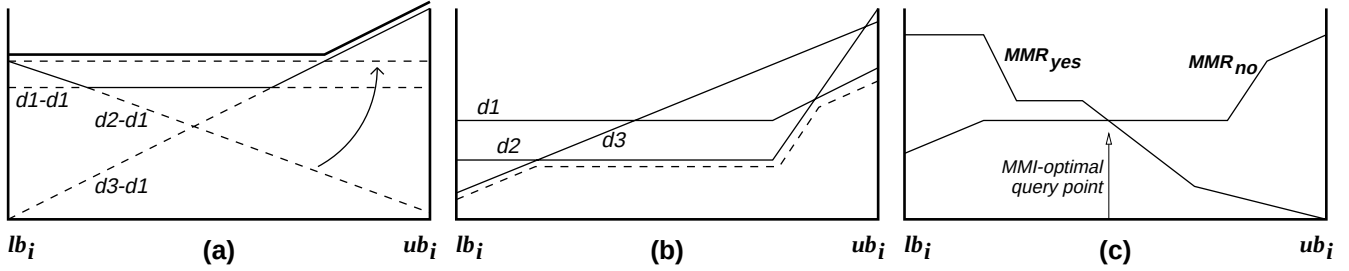


Figure 1: Structure of various functions in dimension i , as a function of l : (a) The PMR for d_1 (w.r.t. d_1, d_2, d_3) are shown; as is $MR(d, \mathcal{C}, i, l)$, the max (thin solid line) of pairwise max regret functions. MR_{no} (thick solid upper line) is obtained by replacing the PMR line $d_2 - d_1$ (with negative slope) by the constant line, and again taking the max. (b) MMR_{no} (dashed line) is the min of the MR_{no} functions for each decision; (c) Intersection of MMR_{no} and MMR_{yes} gives maximin improvement in MMR.

to max regret in dimensions other than i is constant, while the contribution to regret in dimension i given $u_i = l$ is linear in l , with coefficient $\Pr_{d'}(s_i) - \Pr_d(s_i)$. It follows that $MR(d, \mathcal{C}, i, l) = \max_{d'} PMR(d, d', \mathcal{C}, i, l)$ is a PWL convex function of l , since it is the max of these pairwise regret functions (see Figure 1(a)).

Now define $MR_{no}(d, \mathcal{C}, i, l) = MR(d, \mathcal{C} \cup \{u_i \leq l\})$, the max regret of d after obtaining a negative response to query $q_i(l)$. (Note that this differs from $MR(d, \mathcal{C}, i, l)$, which is defined by assuming $u_i = l$, not $u_i \leq l$.) $MR_{no}(d, \mathcal{C}, i, l)$ is also a PWL convex function, obtained from the set of linear PMR -functions that make up $MR(d, \mathcal{C}, i, l)$ as follows: we replace any linear components with negative slope by the constant line $MR(d, \mathcal{C}, i, lb_i)$. Intuitively, if the regret of d w.r.t. d' after learning $u_i = l$ decreases with l , then simply learning that $u_i \leq l$ cannot reduce pairwise max regret, since this weaker constraint does not rule out the maximum regret at $l = lb_i$ (see Figure 1(a) for this intuitive flattening of the regret line for $d_2 - d_1$). This ensures that $MR_{no}(d, \mathcal{C}, i, l)$ is also nondecreasing in l , as illustrated in Figure 1(a).

Finally, note that, by definition of minimax regret, $MMR_{no}(\mathcal{C}, i, l) = \min_d MR_{no}(d, \mathcal{C}, i, l)$. The minimum of a collection of PWL, convex, nondecreasing functions is also PWL and nondecreasing (though not necessarily convex) (see Figure 1(b)). ◀

Note that this proof sketch shows how to construct a finite representation of the function $MMR_{no}(\mathcal{C}, i, l)$ as a finite collection of linear functions and inflection points. By entirely analogous reasoning, we also have:

Theorem 2 For any $i \leq n$, $MMR_{yes}(\mathcal{C}, i, l)$ is a PWL, non-increasing function of l .

3.2 Maximin Improvement

One goal of any query strategy is to determine utility information that reduces regret as quickly as possible. Unfortunately, for any given SGQ $q_i(l)$, the exact reduction in regret cannot typically be predicted in advance, since it differs depending on whether a *yes* or *no* response is obtained. The *maximin improvement* (MMI) query strategy myopically selects queries with the best *worst-case* response. More precisely, let $\mathcal{C}$ be our current constraint set. We define the minimum

improvement of query $q_i(l)$:

$$MI(q_i(l), \mathcal{C}) = MMR(\mathcal{C}) - \max\{MMR_{yes}(\mathcal{C}, i, l), MMR_{no}(\mathcal{C}, i, l)\}.$$

At each stage, the query $q_i(l)$ is asked whose minimum improvement with respect to the current constraint set is maximum. The process stops when no query has minimum improvement greater than threshold τ .

To compute the optimal MMI query point, we find the optimal query point in each dimension i , and ask the SGQ corresponding to the dimension with greatest MMI. The PWL representation of the functions $MMR_{yes}(\mathcal{C}, i, l)$ and $MMR_{no}(\mathcal{C}, i, l)$ described above allows the optimal point in each dimension to be computed readily. The point l that offers MMI in dimension i can be determined by computing the intersection of the two functions: since one is nondecreasing and the other nonincreasing, the maximum point of the function $\max\{MMR_{yes}, MMR_{no}\}$ must lie at the intersection. Note that the intersection must exist since each has the same maximum value $MMR(\mathcal{C})$ (see Figure 1(c)).⁶ Finally, the value of the improvement in regret is the difference between the original minimax regret level and this value.

Computation of the intersection of these functions is straightforward, requiring only the computation of the intersection of the linear segments whose bounds overlap. As such, this can be accomplished in linear time in the maximum number of segments in either function. The number of segments in these functions is (very loosely) bounded by $|D|^{2.7}$. Since we must compute the optimizing point for each utility dimension, the complexity of this algorithm is $O(n|D|^2)$. The algorithm thus scales linearly in the number of outcomes and quadratically with the number of decisions.

3.3 Average and Expected Improvement

One difficulty with the MMI criterion for query selection is that, due to its worst-case nature, we can often find situations in which no query offers positive (minimum) improvement (we will see evidence of this in the next section), despite the

⁶If the intersection occurs where both functions are “flat,” any query point in the intersection can be used.

⁷In practice, the number of segments appears to grow sublinearly in the number of decisions $|D|$.

fact that the current regret level is positive. This occurs when at least one of the responses for every query offers no improvement, thus stalling the query process. Intuitively, just because one response to a query $q_i(l)$ offers no improvement is no reason not to ask the query: the opposite response may still offer immediate improvement.⁸ This suggests an alternative criterion called *maximum average improvement (MAI)*: SGQs are ranked according to the average improvement offered by both positive and negative responses.

Computing the optimal query point according to MAI can also exploit the PWL nature of the functions $MMR_{yes}(\mathcal{C}, i, l)$ and $MMR_{no}(\mathcal{C}, i, l)$. As with MMI, we compute the optimal query point l in each dimension i independently. It is not hard to see that the point of maximum average improvement must occur at an inflection point of one of the two functions. Thus, each dimension can be optimized in time linear in the number of segments in the two functions.

The MAI criterion is not subject to stalling in the sense that MMI is: if $MMR(\mathcal{C})$ is positive, then there exists some query with positive MAI (this will follow from a result discussed below). However, it is subject to a different form of stalling: it may well be the case that the query $q_i(l)$ with MAI occurs at one of the boundary points ub_i or lb_i . In such a case, only one (consistent) response is possible, imposing no additional constraints on the utility function. As such, the constraint set $\mathcal{C}$ will remain unchanged, meaning that the same query remains MAI-optimal at the next stage.

This second type of stalling can be prevented. Suppose that a query $q_i(l)$ is optimal, where $l = ub_i$. We know that $MMR_{yes}(\mathcal{C}, i, ub_i) < MMR_{no}(\mathcal{C}, i, ub_i)$ (i.e., a *yes* offers greater improvement at the point ub_i); but since u_i cannot exceed ub_i , the probability of receiving a *yes* response is zero, so the *yes*-improvement cannot be realized. We can thus make the query $q_i(l)$ appear to be less desirable by accounting for the odds of receiving a specific response. The *maximum expected improvement (MEI)* criterion does just this. We define the expected improvement of a query:

$$EI(q_i(l), \mathcal{C}) = MMR(\mathcal{C}) - [\Pr(yes|q_i(l), \mathcal{C})MMR_{yes}(\mathcal{C}, i, l) + \Pr(no|q_i(l), \mathcal{C})MMR_{no}(\mathcal{C}, i, l)].$$

At each stage, we ask the query with maximum EI.

Computation of expected improvement requires some distribution over responses. For simplicity, we assume a uniform distribution over utility functions and noise-free responses: thus, $\Pr(yes|q_i(l), \mathcal{C}) = (ub_i - l)/(ub_i - lb_i)$ (with the negative probability defined similarly). This assumption also allows for the ready computation of the MEI-optimal query. Again, we optimize each dimension separately. The optimization in dimension i can be effected by doing separate optimizations in the regions defined by the union of the inflection points in the functions $MMR_{yes}(\mathcal{C}, i, l)$ and $MMR_{no}(\mathcal{C}, i, l)$. We defer the details, but note that the function being optimized within each region is a simple quadratic function of l that can be solved analytically. Thus the computational complexity of this criterion is similar to that of MMI and MAI. Fortunately, MEI is not subject to stalling:

⁸Note that, since MMI is myopic, even a nonimproving response may offer information that can be exploited in the future.

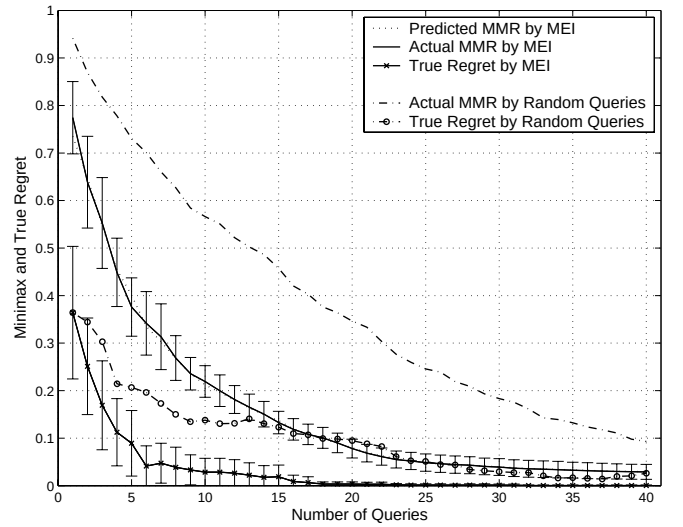


Figure 2: Performance of MEI on three-good problems (40 runs).

Proposition 1 *If $MMR(\mathcal{C}) > 0$, then there exists some query $q_i(l)$ with positive expected improvement; and at least one response to the MEI-optimal query reduces minimax regret.*

The MEI criterion could be adopted using other distributional assumptions, though the optimization required by query selection could become more complicated. It is worth noting that the manner in which we use distributional information is consistent with the worst-case perspective imposed by the minimax criterion. With some distribution over utility functions, we could adopt the perspective of [4; 2], and make decisions by taking expectations with respect to this distribution. However, even in this case, minimax regret allows one to offer *guarantees* on decision quality that a Bayesian approach does not address. The MEI criterion exploits distributional information *only* to guide the querying process, hoping to reach a point more quickly where acceptable guarantees can be provided; the distribution is not used to evaluate decisions *per se*.

While other prior distributions will generally require a different approach to the optimization problem for query selection, it is interesting to observe that queries associated with a *mixture of uniform distributions* can be determined in exactly the same manner. The derivation of the optimal query given such a mixture is straightforward, and these models have the desirable property (like uniform priors) that they are closed under update by query responses. Thus, if our prior beliefs can be approximated well using a mixture of uniforms with a small number of components, MEI-querying can be used directly as described here, without distribution-specific optimization. It is important to note, however, that, even if the approximate priors are used, the decision quality of the MEI strategy is unaffected—only the number of queries required may be adversely impacted.

4 Empirical Results

We evaluated the MMI, MAI, and MEI query criteria on a number of elicitation problems in two different domains. With the MEI criterion, we have also tested its robustness to different assumptions about the prior over utility functions.

We first tested our methods in two bidding scenarios involving simultaneous auctions and combinatorial preferences [2]. In the first scenario, a bidding agent must offer bids for four different goods auctioned simultaneously in four different markets. To discretize the decision space, we assume that the agent can offer three distinct bids—low, medium, and high—for each good. Each of these bid levels corresponds to a precise cost: should the bid win, the user pays the price associated with each bid (with, of course, higher prices associated with higher bid levels). To suppress the need for strategic reasoning, the agent has a fixed, known probability of winning a good associated with each of the three bid levels. The probabilities of winning each good are independent, and increasing in the bid level. With four goods and three bid levels, there are 81 possible decisions (mappings from markets to bids) and 16 outcomes (subsets of goods the user might obtain). The user’s utility function need not be additive with respect to the goods obtained. For instance, the user might value goods g_1 and g_2 in conjunction, but may value neither individually. Thus utility is associated with each of the 16 outcomes. We assume that the overall utility function (accounting for price paid) is quasi-linear; so the price paid is subtracted from the utility of the subset of goods obtained. A smaller scenario with three goods was also run: this has 27 decisions and 8 outcomes.

We first discuss the smaller (3-good) scenario. For each query criterion, we run elicitation using that criterion for 40 steps (or until no query has positive value). For each criterion, 40 trials using random utility functions drawn from $[0, 1]^n$ were run, with elicitation simulated using responses based on that function. For each query in a run, we record: (a) *predicted MMR*—the MMR level that is predicted to hold after asking the optimal query;⁹ (b) *actual MMR*—the MMR level realized once the actual query response is obtained; and (c) the *true regret*—the difference in utility between the minimax decision and the true optimal decision for the underlying utility function. While our algorithms don’t have access to true regret, this measure gives an indication of true decision quality, not just the quality guarantees the algorithms provide.

Figure 2 shows the performance of the MEI criterion (std. error bars are shown on actual MMR and true regret, but are excluded from predicted MMR for legibility). We see that the algorithm quickly converges to a point where the minimax regret guarantees are quite tight: within 20 queries, the average regret guarantee falls below 0.1 (less than 10%); and within forty queries, decision quality is guaranteed to be with 4% of optimal. More interesting, we see that *true regret* falls to under 10% with 5 queries, and to near zero within 20 queries. Thus the actual decision quality associated with acting according to the decision with MMR is generally far better than the MMR guarantee. The MEI criterion seems to select

⁹For example, MMI predicts the maximum regret over all responses, while MEI predicts the expected regret.

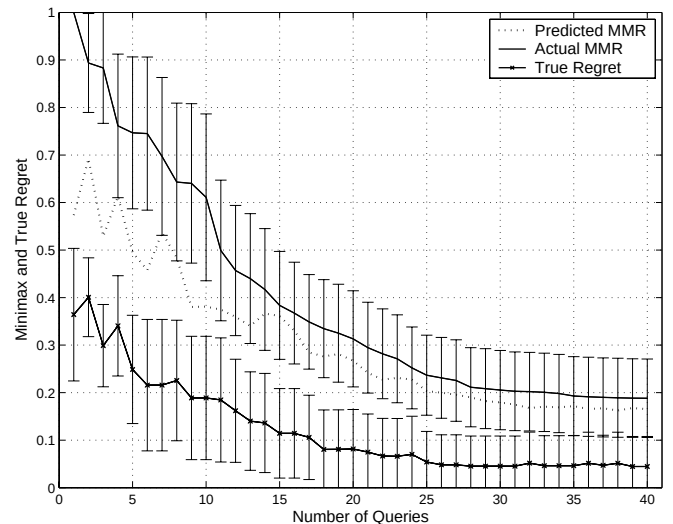


Figure 3: Performance of MAI on three-good problems (40 runs).

suitable queries, allowing optimal decisions and tight regret guarantees to be identified with few interactions. For reference, we also include the performance of randomly selected queries (where both dimension i and query point l are chosen uniformly at random from the feasible region). Because the problem is of relatively low dimension, random queries perform reasonably well, though they have difficulty reducing regret to zero, and do not compete with MEI queries.

Figure 3 shows the same measurements for the MAI criterion. We note that this query strategy does not reduce regret bounds nearly as quickly as the MEI strategy, reaching only an average regret guarantee of 0.18 after 40 queries. True regret is generally much better, but still does not approach the performance of MEI (or even random queries). We note that the MAI criterion often stalls: in such a case, we complete the data with the last minimax regret value. Finally, it is worth noting that the MMI criterion performs extremely badly. We don’t plot its performance, but note that in all runs, it stalls after a maximum of five queries; its average minimax regret bound is 0.8, and average true regret level is 0.3 when it stalls.

The MEI criterion appears to offer much better performance than MAI, MMI, or random querying. Figure 4 shows the performance of MEI, MAI and random querying strategies on the larger four-good (16-outcome, 81-decision) scenario. Again we see that MEI converges quickly and outperforms the other strategies. With the increase in dimensionality, random queries fare worse than MAI-optimal queries. We note that in all experiments, the optimal query (regardless of criterion) can be computed very quickly.

We have also tested the MEI criterion on a travel planning domain (as in the previous tests, MEI seems to dominate the other criteria, so we focus on it). In this domain, an agent must choose a collection of flight segments from a flight database to take a user from a source to a destination city [1]. To make the elicitation problem interesting, we added the following information to the DB: the probability of any flight arrival being delayed by specific amount of time;

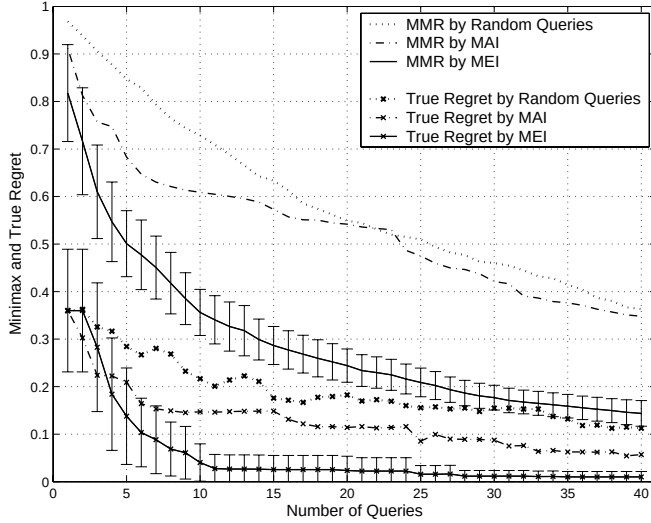


Figure 4: Performance on four-good problems (40 runs).

the probability of missing a connecting flight as a function of connection time and airport; a distribution over ground travel times to a hotel in the destination city as a function of arrival time (reflecting, e.g., arrival in rush hour or off-peak); and the probability of losing a hotel room as a function of arrival time at the hotel. As a result, for any specific flight combination (decision), a joint distribution over these variables (outcome) is obtained. A user's utility function is quasi-linear, given by her utility for a specific outcome over these four attributes less the flight price. The specific formulation discretizes these attributes, so the outcome space is of size 64. In our experiments, the flight DB was designed to allow 20 flights (both direct and indirect) between pairs of cities.

We tested the MEI strategy using a uniform prior over utility space to select queries, with user utility functions drawn from the same uniform distribution. The results for a specific source-destination pair (Toronto-San Francisco) are shown in Figure 5. As before, we see that the MEI strategy easily outperforms random querying, both in terms of the regret guarantees, and the true regret of the decisions it would recommend at each stage. These results are representative of those obtained in other decision scenarios.

We also explored the use of strong prior knowledge to guide the querying process. We repeated the test above, drawing user utility functions from a strongly peaked (truncated) Gaussian distribution over utility space (with diagonal covariance matrix, and variance 0.03 in each dimension). We tested the MEI-criterion using a (hand-chosen) mixture of three uniform distributions over subregions of utility space that very roughly approximated the Gaussian.¹⁰ To test the robustness of MEI to inaccurate priors, we also used MEI using a single uniform prior over all of utility space (despite the fact that the true utility function is drawn from the Gaussian). The results illustrated in Figure 6 demonstrate that MEI can benefit

¹⁰In principle, this mixture could have been fit to the actual prior using, say, EM; but our goal is not accurate modeling of the prior.

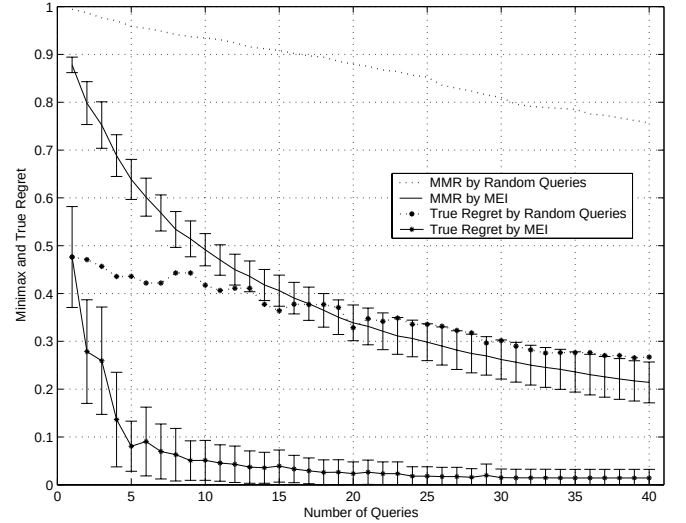


Figure 5: Performance on YYZ-SFO, with utility functions drawn from uniform (40 runs).

considerably from strong prior information. Indeed, minimax regret is reduced very quickly when a reasonable prior is used to select queries; and true regret is reduced to zero in every instance of this scenario within four queries. Random querying does very poorly, indicating that this problem is not easy to solve without sufficient utility information. The robustness of MEI to inaccurate priors is also in evidence. We see that minimax regret and true regret are also reduced very quickly when an uninformative uniform prior is used to guide the MEI-querying process.

5 Concluding Remarks

We have presented a new procedure for decision making with incomplete utility information which recommends decisions that minimize maximum regret. We defined several different myopic query selection criteria, and showed that myopically optimal queries under each criterion can be computed effectively, in polynomial time. The empirical performance of one such criterion, maximum expected improvement, proved to be rather attractive: not only did it provide strong guarantees after few queries, but true decision quality tended to exceed these guarantees significantly.

Our work differs from existing approaches to preference elicitation in several important ways. Like recent Bayesian approaches [4; 2], our approach identifies a concrete decision in the face of utility function uncertainty. Unlike these methods, for the purposes of decision making, we assume only constraints on possible utility functions, not distributions. As a result, the minimax regret criterion is used to identify decisions with guaranteed error bounds on quality. Our use of constraints on utility functions is more closely related to work on ISMAUT [8; 7]. However, the focus in ISMAUT is the identification of Pareto optimal decisions in the face of utility function uncertainty, as opposed to the choice of a specific decision that maximizes some decision criterion. Furthermore,

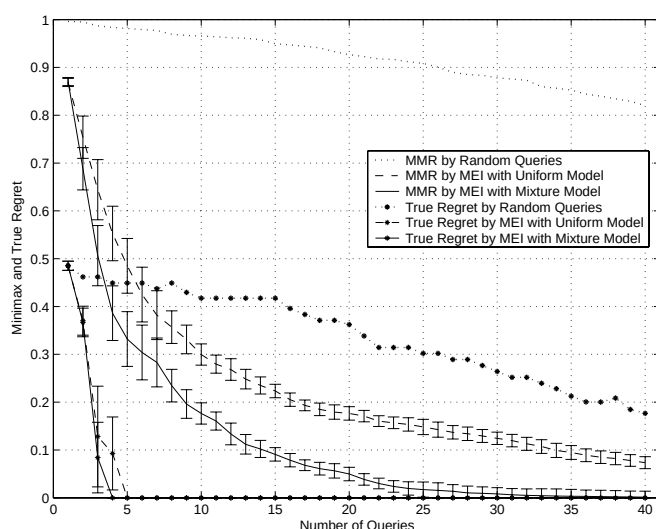


Figure 6: Performance on YYZ-SFO, with utility functions drawn from a strongly peaked Gaussian. MEI querying using both a uniform prior and a 3-component mixture are shown (40 runs).

little attention has been paid to query strategies in ISMAUT, which, in contrast, is our main focus.

There are a number of directions in which this work can be extended. Obviously, scaling issues are of paramount importance. We are currently exploring pruning techniques for removing decisions from consideration that can never be minimax optimal, thus reducing the quadratic dependence on the number of decisions. We are also exploring methods for dealing with more general linear constraints (apart from one-dimensional bounds), as well as more expressive query types. Also of interest are methods for dealing with noisy/inconsistent query responses, and visualization techniques. Finally, we are developing heuristics that simulate some of the effects of nonmyopic elicitation without explicit lookahead. One such technique involves enumerating the vertices of the regions $\mathcal{R}_d$ of utility space in which each decision d is optimal (the regions are convex polytopes). Queries at those points can quickly help rule out suboptimal actions. We hope to combine the computationally attractive methods devised in this paper with more intensive techniques like this to help reduce the number of required queries even further.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS).

References

- [1] Jim Blythe. Visual exploration and incremental utility elicitation. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 526–532, Edmonton, 2002.
- [2] Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 239–246, Edmonton, 2002.
- [3] Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 56–64, Seattle, 2001.
- [4] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 363–369, Austin, TX, 2000.
- [5] Simon French. *Decision Theory*. Halsted Press, New York, 1986.
- [6] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.
- [7] Edward A. Sykes and Chelsea C. White, III. Multiobjective intelligent computer-aided design. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1498–1511, 1991.
- [8] Chelsea C. White, III, Andrew P. Sage, and Shigeru Dozono. A model of multiattribute decisionmaking and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man and Cybernetics*, 14(2):223–229, 1984.

DESCRIPTION LOGICS

Least Common Subsumers and Most Specific Concepts in a Description Logic with Existential Restrictions and Terminological Cycles*

Franz Baader

Theoretical Computer Science
TU Dresden
D-01062 Dresden, Germany
baader@inf.tu-dresden.de

Abstract

Computing least common subsumers (lcs) and most specific concepts (msc) are inference tasks that can support the bottom-up construction of knowledge bases in description logics. In description logics with existential restrictions, the most specific concept need not exist if one restricts the attention to concept descriptions or acyclic TBoxes. In this paper, we extend the notions lcs and msc to cyclic TBoxes. For the description logic $\mathcal{EL}$ (which allows for conjunctions, existential restrictions, and the top-concept), we show that the lcs and msc always exist and can be computed in polynomial time if we interpret cyclic definitions with greatest fixpoint semantics.

1 Introduction

Computing the most specific concept of an individual and the least common subsumer of concepts can be used in the bottom-up construction of description logic (DL) knowledge bases. Instead of defining the relevant concepts of an application domain from scratch, this methodology allows the user to give typical examples of individuals belonging to the concept to be defined. These individuals are then generalized to a concept by first computing the most specific concept of each individual (i.e., the least concept description in the available description language that has this individual as an instance), and then computing the least common subsumer of these concepts (i.e., the least concept description in the available description language that subsumes all these concepts). The knowledge engineer can then use the computed concept as a starting point for the concept definition.

The least common subsumer (lcs) in DLs with existential restrictions was investigated in [Baader *et al.*, 1999]. In particular, it was shown there that the lcs in the small DL $\mathcal{EL}$ (which allows for conjunctions, existential restrictions, and the top-concept) always exists, and that the binary lcs can be computed in polynomial time. Unfortunately, the most specific concept (msc) of a given ABox individual need not exist in languages allowing for existential restrictions or number restrictions. As a possible solution to this problem, Küsters

and Molitor [2001] show how the most specific concept can be approximated in $\mathcal{EL}$ and some of its extensions. Here, we follow an alternative approach: we extend the language by cyclic terminologies with greatest fixpoint (gfp) semantics, and show that the msc always exists in this setting. Of course, then one must also be able to compute the lcs w.r.t. cyclic terminologies with GFP-semantics. For the DL $\mathcal{ALN}$ (which allows for conjunctions, value restrictions, and number restrictions) it was shown in [Baader and Küsters, 1998] that the most specific concept always exists if one adds cyclic concept definitions with GFP-semantics. One reason for Küsters and Molitor to choose an approximation approach rather than an exact characterization of the most specific concept using cyclic definitions was that the impact of cyclic definitions in description logics with existential restrictions was largely unexplored.

The paper [Baader, 2003a] is a first step toward overcoming this deficit. It considers cyclic terminologies in $\mathcal{EL}$ w.r.t. the three types of semantics (greatest fixpoint, least fixpoint, and descriptive semantics) introduced by Nebel [1991], and shows that the subsumption problem can be decided in polynomial time in all three cases. This is in stark contrast to the case of DLs with value restrictions. Even for the small DL $\mathcal{FL}_0$ (which allows conjunctions and value restrictions only), adding cyclic terminologies increases the complexity of the subsumption problem from polynomial (for concept descriptions) to PSPACE. The main tool in the investigation of cyclic definitions in $\mathcal{EL}$ is a characterization of subsumption through the existence of so-called simulation relations on the graph associated with an $\mathcal{EL}$ -terminology, which can be computed in polynomial time [Henzinger *et al.*, 1995].

This characterization of subsumption can be used to characterize the lcs w.r.t. GFP-semantics via the product of this graph with itself (Section 4). This shows that, w.r.t. GFP semantics, the lcs always exists, and that the binary lcs can be computed in polynomial time. (The n -ary lcs may grow exponentially even in $\mathcal{EL}$ without cyclic terminologies [Baader *et al.*, 1999].)

The characterization of subsumption w.r.t. GFP-semantics can be extended to the instance problem in $\mathcal{EL}$. This allows us to show that the msc in $\mathcal{EL}$ with cyclic terminologies interpreted with GFP semantics always exists, and can be computed in polynomial time (Section 5).

In the next section, we introduce $\mathcal{EL}$ with cyclic terminolo-

*Partially supported by the DFG under grant BA 1122/4-3.

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top-concept	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
exist. restriction	$\exists r.C$	$\{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concept definition	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$
individual name	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concept assertion	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
role assertion	$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}$.

gies as well as the lcs and the msc. Then we recall the important definitions and results from [Baader, 2003a]. Section 4 formulates and proves the new results for the lcs, and Section 5 does the same for the msc.

2 Cyclic terminologies, least common subsumers, and most specific concepts

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. The constructors determine the expressive power of the DL. In this paper, we restrict the attention to the DL $\mathcal{EL}$, whose concept descriptions are formed using the constructors top-concept ($\top$), conjunction ($C \sqcap D$), and existential restriction ($\exists r.C$). The semantics of $\mathcal{EL}$ -concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

A *terminology* (or *TBox* for short) is a finite set of concept definitions of the form $A \equiv D$, where A is a concept name and D a concept description. In addition, we require that TBoxes do not contain *multiple definitions*, i.e., there cannot be two distinct concept descriptions D_1 and D_2 such that both $A \equiv D_1$ and $A \equiv D_2$ belongs to the TBox. Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names occurring in the TBox are called *primitive concepts*. Note that we allow for cyclic dependencies between the defined concepts, i.e., the definition of A may refer (directly or indirectly) to A itself. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all definitions $A \equiv D$ in $\mathcal{T}$.

An *ABox* is a finite set of assertions of the form $A(a)$ and $r(a, b)$, where A is a concept name, r is a role name, and a, b are individual names from a set N_I . Interpretations of ABoxes must additionally map each individual name $a \in N_I$ to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of the ABox $\mathcal{A}$ iff it satisfies all its assertions, i.e., $a^{\mathcal{I}} \in A^{\mathcal{I}}$ for all concept assertions $A(a)$ in $\mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all role assertions $r(a, b)$ in $\mathcal{A}$. The interpretation $\mathcal{I}$ is a model of the ABox $\mathcal{A}$ together with the TBox $\mathcal{T}$ iff it is a model of both $\mathcal{T}$ and $\mathcal{A}$.

The semantics of (possibly cyclic) $\mathcal{EL}$ -TBoxes we have defined above is called *descriptive semantic* by Nebel [1991]. For some applications, it is more appropriate to interpret cyclic concept definitions with the help of a fixpoint semantics.

Example 1 To illustrate this, let us recall an example from [Baader, 2003a]: $\text{Inode} \equiv \text{Node} \sqcap \exists \text{edge}.\text{Inode}$.

Here the intended interpretations are graphs where we have nodes (elements of the concept Node) and edges (represented by the role edge), and we want to define the concept Inode of all nodes lying on an infinite (possibly cyclic) path of the graph. In order to capture this intuition, the above definition must be interpreted with greatest fixpoint semantics.

Before we can define greatest fixpoint semantics (gfp-semantics), we must introduce some notation. Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox containing the roles N_{role} , the primitive concepts N_{prim} , and the defined concepts $N_{def} = \{A_1, \dots, A_k\}$. A *primitive interpretation* $\mathcal{J}$ for $\mathcal{T}$ is given by a domain $\Delta^{\mathcal{J}}$, an interpretation of the roles $r \in N_{role}$ by binary relations $r^{\mathcal{J}}$ on $\Delta^{\mathcal{J}}$, and an interpretation of the primitive concepts $P \in N_{prim}$ by subsets $P^{\mathcal{J}}$ of $\Delta^{\mathcal{J}}$. Obviously, a primitive interpretation differs from an interpretation in that it does not interpret the defined concepts in N_{def} . We say that the interpretation $\mathcal{I}$ is *based on* the primitive interpretation $\mathcal{J}$ iff it has the same domain as $\mathcal{J}$ and coincides with $\mathcal{J}$ on N_{role} and N_{prim} . For a fixed primitive interpretation $\mathcal{J}$, the interpretations $\mathcal{I}$ based on it are uniquely determined by the tuple $(A_1^{\mathcal{I}}, \dots, A_k^{\mathcal{I}})$ of the interpretations of the defined concepts in N_{def} . We define

$$\text{Int}(\mathcal{J}) := \{\mathcal{I} \mid \mathcal{I} \text{ is an interpretation based on } \mathcal{J}\}.$$

Interpretations based on $\mathcal{J}$ can be compared by the following ordering, which realizes a pairwise inclusion test between the respective interpretations of the defined concepts: if $\mathcal{I}_1, \mathcal{I}_2 \in \text{Int}(\mathcal{J})$, then

$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \text{ iff } A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2} \text{ for all } i, 1 \leq i \leq k.$$

It is easy to see that $\preceq_{\mathcal{J}}$ induces a *complete lattice* on $\text{Int}(\mathcal{J})$, i.e., every subset of $\text{Int}(\mathcal{J})$ has a least upper bound (lub) and a greatest lower bound (glb). Using *Tarski's fixpoint theorem* [Tarski, 1955] for complete lattices, it is not hard to show [Nebel, 1991] that, for a given primitive interpretation $\mathcal{J}$, there always is a greatest (w.r.t. $\preceq_{\mathcal{J}}$) model of $\mathcal{T}$ based on $\mathcal{J}$. We call this models the *greatest fixpoint model (gfp-model)* of $\mathcal{T}$. *Greatest fixpoint semantics* considers only gfp-models as admissible models.

Definition 2 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and $\mathcal{A}$ an $\mathcal{EL}$ -ABox, let A, B be defined concepts occurring in $\mathcal{T}$, and a an individual name occurring in $\mathcal{A}$. Then,

- A is subsumed by B w.r.t. gfp-semantics ($A \sqsubseteq_{gfp, \mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all gfp-models $\mathcal{I}$ of $\mathcal{T}$.
- a is an instance of A w.r.t. gfp-semantics ($\mathcal{A} \models_{gfp, \mathcal{T}} A(a)$) iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{A}$ that are gfp-models of $\mathcal{T}$.

On the level of concept descriptions, the least common subsumer of two concept descriptions C, D is the least concept

description E that subsumes both C and D . An extensions of this definition to the level of (possibly cyclic) TBoxes is not completely trivial. In fact, assume that A_1, A_2 are concepts defined in the TBox $\mathcal{T}$. It should be obvious that taking as the lcs of A_1, A_2 the least defined concept B in $\mathcal{T}$ such that $A_1 \sqsubseteq_{\text{gfp}, \mathcal{T}} B$ and $A_2 \sqsubseteq_{\text{gfp}, \mathcal{T}} B$ is too weak since the lcs would then strongly depend on what other defined concepts are already present in $\mathcal{T}$. However, a second approach (which might look like the obvious generalization of the definition of the lcs in the case of concept descriptions) is also not quite satisfactory. We could say that the lcs of A_1, A_2 is the least concept description C (possibly using defined concepts of $\mathcal{T}$) such that $A_1 \sqsubseteq_{\text{gfp}, \mathcal{T}} C$ and $A_2 \sqsubseteq_{\text{gfp}, \mathcal{T}} C$. The problem is that this definition does not allow us to use the expressive power of cyclic definitions (with gfp-semantics) when constructing the lcs. For example, consider the TBox $\mathcal{T}$ consisting of the following concept definitions:

$\text{BlueNode} \equiv \text{Blue} \sqcap \text{Node} \sqcap \exists \text{edge}.\text{BlueNode},$
 $\text{RedNode} \equiv \text{Red} \sqcap \text{Node} \sqcap \exists \text{edge}.\text{RedNode}.$

The intended interpretation is similar to the one in Example 1, with the only difference that now nodes may have colors, and we are interested in blue (red) nodes lying on an infinite path consisting of blue (red) nodes. Intuitively, the lcs of BlueNode and RedNode describes nodes lying on an infinite path (without any restriction on their color), i.e., the concept lNode from Example 1 should be a definition of this lcs. However, this cannot be expressed by a simple concept description. It requires a new cyclic definition.

Consequently, to obtain the lcs we must allow the original TBox to be extended by new definitions. We say that the TBox $\mathcal{T}_2$ is a *conservative extension* of the TBox $\mathcal{T}_1$ iff $\mathcal{T}_1 \subseteq \mathcal{T}_2$ and $\mathcal{T}_1$ and $\mathcal{T}_2$ have the same primitive concepts and roles. Thus, $\mathcal{T}_2$ may contain new definitions $A \equiv D$, but then D does not introduce new primitive concepts and roles (i.e., all of them already occur in $\mathcal{T}_1$), and A is a new concept name (i.e., A does not occur in $\mathcal{T}_1$). The name “conservative extension” is justified by the fact that the new definitions in $\mathcal{T}_2$ do not influence the subsumption relationships between defined concepts in $\mathcal{T}_1$ (see [Baader, 2002] for a proof).

Lemma 3 *Let $\mathcal{T}_1, \mathcal{T}_2$ be $\mathcal{EL}$ -TBoxes such that $\mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$, and let A, B be defined concepts in $\mathcal{T}_1$ (and thus also in $\mathcal{T}_2$). Then $A \sqsubseteq_{\text{gfp}, \mathcal{T}_1} B$ iff $A \sqsubseteq_{\text{gfp}, \mathcal{T}_2} B$.*

Definition 4 Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox containing the defined concepts A, B , and let $\mathcal{T}_2$ be a conservative extension of $\mathcal{T}_1$ containing the new defined concept E . Then E in $\mathcal{T}_2$ is a *least common subsumer* of A, B in $\mathcal{T}_1$ w.r.t. *gfp-semantics* (gfp-lcs) iff the following two conditions are satisfied:

1. $A \sqsubseteq_{\text{gfp}, \mathcal{T}_2} E$ and $B \sqsubseteq_{\text{gfp}, \mathcal{T}_2} E$.
2. If $\mathcal{T}_3$ is a conservative extension of $\mathcal{T}_2$ and F a defined concept in $\mathcal{T}_3$ such that $A \sqsubseteq_{\text{gfp}, \mathcal{T}_3} F$ and $B \sqsubseteq_{\text{gfp}, \mathcal{T}_3} F$, then $E \sqsubseteq_{\text{gfp}, \mathcal{T}_3} F$.

In the case of concept descriptions, the lcs is unique up to equivalence, i.e., if E_1 and E_2 are both least common subsumers of the descriptions C, D , then $E_1 \equiv E_2$ (i.e., $E_1 \sqsubseteq E_2$ and $E_2 \sqsubseteq E_1$). In the presence of (possibly cyclic) TBoxes, this uniqueness property also holds (though its formulation is more complicated).

Proposition 5 *Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox containing the defined concepts A, B . Assume that $\mathcal{T}_2$ and $\mathcal{T}_2'$ are conservative extensions of $\mathcal{T}_1$ such that*

- *the defined concept E in $\mathcal{T}_2$ is a gfp-lcs of A, B in $\mathcal{T}_1$;*
- *the defined concept E' in $\mathcal{T}_2'$ is a gfp-lcs of A, B in $\mathcal{T}_1$;*
- *the sets of newly defined concepts in respectively $\mathcal{T}_2$ and $\mathcal{T}_2'$ are disjoint.*

For $\mathcal{T}_3 := \mathcal{T}_2 \cup \mathcal{T}_2'$, we have $E \equiv_{\text{gfp}, \mathcal{T}_3} E'$ (i.e., $E \sqsubseteq_{\text{gfp}, \mathcal{T}_3} E'$ and $E' \sqsubseteq_{\text{gfp}, \mathcal{T}_3} E$).

The notion “most specific concept” can be extended in a similar way from concept descriptions to concepts defined in a TBox.

Definition 6 Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox and $\mathcal{A}$ an $\mathcal{EL}$ -ABox containing the individual name a , and let $\mathcal{T}_2$ be a conservative extension of $\mathcal{T}_1$ containing the defined concept E . Then E in $\mathcal{T}_2$ is a *most specific concept* of a in $\mathcal{A}$ and $\mathcal{T}_1$ w.r.t. *gfp-semantics* (gfp-msc) iff the following two conditions are satisfied:

1. $\mathcal{A} \models_{\text{gfp}, \mathcal{T}_2} E(a)$.
2. If $\mathcal{T}_3$ is a conservative extension of $\mathcal{T}_2$ and F a defined concept in $\mathcal{T}_3$ such that $\mathcal{A} \models_{\text{gfp}, \mathcal{T}_3} F(a)$, then $E \sqsubseteq_{\text{gfp}, \mathcal{T}_3} F$.

Uniqueness up to equivalence of the most specific concept can be formulated and shown like uniqueness of the least common subsumer.

3 Characterizing subsumption

In this section, we recall the characterizations of subsumption w.r.t. gfp-semantics developed in [Baader, 2003a]. To this purpose, we must represent TBoxes and primitive interpretations by description graphs, and introduce the notion of a simulation on description graphs.

Before we can translate $\mathcal{EL}$ -TBoxes into description graphs, we must normalize the TBoxes. In the following, let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox, N_{def} the defined concepts of $\mathcal{T}$, N_{prim} the primitive concepts of $\mathcal{T}$, and N_{role} the roles of $\mathcal{T}$.

We say that the $\mathcal{EL}$ -TBox $\mathcal{T}$ is *normalized* iff $A \equiv D \in \mathcal{T}$ implies that D is of the form

$$P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_\ell.B_\ell,$$

for $m, \ell \geq 0$, $P_1, \dots, P_m \in N_{\text{prim}}$, $r_1, \dots, r_\ell \in N_{\text{role}}$, and $B_1, \dots, B_\ell \in N_{\text{def}}$. If $m = \ell = 0$, then $D = \top$.

As shown in [Baader, 2003a], one can (without loss of generality) restrict the attention to normalized TBox. In the following, we thus assume that all TBoxes are normalized. Normalized $\mathcal{EL}$ -TBoxes can be viewed as graphs whose nodes are the defined concepts, which are labeled by sets of primitive concepts, and whose edges are given by the existential restrictions. For the rest of this section, we fix a normalized $\mathcal{EL}$ -TBox $\mathcal{T}$ with primitive concepts N_{prim} , defined concepts N_{def} , and roles N_{role} .

Definition 7 An $\mathcal{EL}$ -description graph is a graph $\mathcal{G} = (V, E, L)$ where

- V is a set of nodes;

- $E \subseteq V \times N_{role} \times V$ is a set of edges labeled by role names;
- $L: V \rightarrow 2^{N_{prim}}$ is a function that labels nodes with sets of primitive concepts.

The normalized TBox $\mathcal{T}$ can be translated into the following $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{T}} = (N_{def}, E_{\mathcal{T}}, L_{\mathcal{T}})$:

- the nodes of $\mathcal{G}_{\mathcal{T}}$ are the defined concepts of $\mathcal{T}$;
- if A is a defined concept and

$$A \equiv P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_\ell.B_\ell$$

its definition in $\mathcal{T}$, then

- $L_{\mathcal{T}}(A) = \{P_1, \dots, P_m\}$, and
- A is the source of the edges $(A, r_1, B_1), \dots, (A, r_\ell, B_\ell) \in E_{\mathcal{T}}$.

Any primitive interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ can be translated into the following $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{J}} = (\Delta^{\mathcal{J}}, E_{\mathcal{J}}, L_{\mathcal{J}})$:

- the nodes of $\mathcal{G}_{\mathcal{J}}$ are the elements of $\Delta^{\mathcal{J}}$;
- $E_{\mathcal{J}} := \{(x, r, y) \mid (x, y) \in r^{\mathcal{J}}\}$;
- $L_{\mathcal{J}}(x) = \{P \in N_{prim} \mid x \in P^{\mathcal{J}}\}$ for all $x \in \Delta^{\mathcal{J}}$.

Conversely, every $\mathcal{EL}$ -description graph can be viewed as representing either an $\mathcal{EL}$ -TBox or a primitive interpretation.

Simulations are binary relations between nodes of two $\mathcal{EL}$ -description graphs that respect labels and edges in the sense defined below.

Definition 8 Let $\mathcal{G}_i = (V_i, E_i, L_i)$ ($i = 1, 2$) be two $\mathcal{EL}$ -description graphs. The binary relation $Z \subseteq V_1 \times V_2$ is a *simulation* from $\mathcal{G}_1$ to $\mathcal{G}_2$ iff

- (S1) $(v_1, v_2) \in Z$ implies $L_1(v_1) \subseteq L_2(v_2)$; and
- (S2) if $(v_1, v_2) \in Z$ and $(v_1, r, v'_1) \in E_1$, then there exists a node $v'_2 \in V_2$ such that $(v'_1, v'_2) \in Z$ and $(v_2, r, v'_2) \in E_2$.

We write $Z: \mathcal{G}_1 \rightsquigarrow \mathcal{G}_2$ to express that Z is a simulation from $\mathcal{G}_1$ to $\mathcal{G}_2$.

It is easy to see that the set of all simulations from $\mathcal{G}_1$ to $\mathcal{G}_2$ is closed under arbitrary unions. Consequently, there always exists a greatest simulation from $\mathcal{G}_1$ to $\mathcal{G}_2$. If $\mathcal{G}_1, \mathcal{G}_2$ are finite, then this greatest simulation can be computed in polynomial time [Henzinger *et al.*, 1995]. As an easy consequence of this fact, the following proposition is proved in [Baader, 2003a].

Proposition 9 Let $\mathcal{G}_1, \mathcal{G}_2$ be two finite $\mathcal{EL}$ -description graphs, v_1 a node of $\mathcal{G}_1$ and v_2 a node of $\mathcal{G}_2$. Then we can decide in polynomial time whether there is a simulation $Z: \mathcal{G}_1 \rightsquigarrow \mathcal{G}_2$ such that $(v_1, v_2) \in Z$.

Subsumption w.r.t. gfp-semantic corresponds to the existence of a simulation relation such that the subsumee simulates the subsumer:

Theorem 10 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and A, B defined concepts in $\mathcal{T}$. Then the following are equivalent:

1. $A \sqsubseteq_{gfp, \mathcal{T}} B$.

2. There is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{T}}$ such that $(B, A) \in Z$.

The theorem together with Proposition 9 shows that subsumption w.r.t. gfp-semantic in $\mathcal{EL}$ is tractable. The proof of the theorem given in [Baader, 2003a] depends on a characterization of when an individual of a gfp-model belongs to a defined concept in this model.

Proposition 11 Let $\mathcal{J}$ be a primitive interpretation and $\mathcal{I}$ the gfp-model of $\mathcal{T}$ based on $\mathcal{J}$. Then the following are equivalent for any $A \in N_{def}$ and $x \in \Delta^{\mathcal{J}}$:

1. $x \in A^{\mathcal{I}}$.
2. There is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{J}}$ such that $(A, x) \in Z$.

This proposition is also important in the proof of correctness of our characterization of the instance problem (Theorem 17).

4 Computing the lcs

We will show how the characterization of subsumption w.r.t. gfp-semantic given in Theorem 10 can be used to characterize the gfp-lcs (see [Baader, 2002] for more details and proofs). Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox, let $\mathcal{G}_{\mathcal{T}_1} = (N_{def}, E_{\mathcal{T}_1}, L_{\mathcal{T}_1})$ be the corresponding description graph, and let A, B be defined concepts in $\mathcal{T}_1$ (i.e., elements of N_{def}). In principle, the lcs of A, B in $\mathcal{T}_1$ is defined in a TBox whose description graph is the product of $\mathcal{G}_{\mathcal{T}_1}$ with itself.

Definition 12 Let $\mathcal{G}_1 = (V_1, E_1, L_1)$ and $\mathcal{G}_2 = (V_2, E_2, L_2)$ be two description graphs. Their *product* is the description graph $\mathcal{G}_1 \times \mathcal{G}_2 := (V, E, L)$ where

- $V := V_1 \times V_2$;
- $E := \{((v_1, v_2), r, (v'_1, v'_2)) \mid (v_1, r, v'_1) \in E_1 \wedge (v_2, r, v'_2) \in E_2\}$;
- $L(v_1, v_2) := L_1(v_1) \cap L_2(v_2)$.

The description graph $\mathcal{G}_{\mathcal{T}_1} \times \mathcal{G}_{\mathcal{T}_1}$ yields a TBox $\mathcal{T}$ such that $\mathcal{G}_{\mathcal{T}} = \mathcal{G}_{\mathcal{T}_1} \times \mathcal{G}_{\mathcal{T}_1}$. Now, $\mathcal{T}_2 := \mathcal{T}_1 \cup \mathcal{T}$ is a conservative extension of $\mathcal{T}_1$. In fact, $\mathcal{G}_{\mathcal{T}_1} \times \mathcal{G}_{\mathcal{T}_1}$ is based on the same primitive concepts and roles as $\mathcal{G}_{\mathcal{T}_1}$, and the set of defined concepts in $\mathcal{T}$ is $N_{def} \times N_{def}$, which is disjoint from N_{def} .

Lemma 13 (A, B) in $\mathcal{T}_2$ is the gfp-lcs of A and B in $\mathcal{T}_1$.

Computing the (binary) product of two $\mathcal{EL}$ -description graphs can obviously be done in polynomial time, and thus the gfp-lcs can be computed in polynomial time.

Theorem 14 Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox, and let A, B be defined concepts in $\mathcal{T}_1$. Then the gfp-lcs of A, B in $\mathcal{T}_1$ always exists, and it can be computed in polynomial time.

5 Instance and most specific concepts

One motivation for considering cyclic terminologies in $\mathcal{EL}$ is the fact that the most specific concept of an ABox individual need not exist in $\mathcal{EL}$ (without cyclic terminologies). An example is the simple cyclic ABox $\mathcal{A} := \{r(b, b)\}$, where b has no most specific concept, i.e., there is no least $\mathcal{EL}$ -concept description D such that b is an instance of D w.r.t. $\mathcal{A}$ [Küsters and Molitor, 2001]. However, if one allows for cyclic TBoxes with gfp-semantic, then the defined concept B with $B \equiv \exists r.B$ is such a most specific concept.

5.1 The instance problem w.r.t. gfp-semantics

Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ -TBox and $\mathcal{A}$ an $\mathcal{EL}$ -ABox. In the following, we assume that $\mathcal{T}$ is fixed and that all instance problems for $\mathcal{A}$ are considered w.r.t. this TBox.

In this setting, $\mathcal{A}$ can be translated into an $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{A}}$ by viewing $\mathcal{A}$ as a graph and extending it appropriately by the graph $\mathcal{G}_{\mathcal{T}}$ associated with $\mathcal{T}$. The idea is then that the characterization of the instance problem should be similar to the statement of Proposition 11: the individual a is an instance of A in $\mathcal{A}$ iff there is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{A}}$ such that $(A, a) \in Z$.

Before giving an exact definition of $\mathcal{G}_{\mathcal{A}}$, we consider an example that demonstrates that a too simple-minded realization of this idea does not work. Let

$$\begin{aligned}\mathcal{A} &:= \{A(a), P(a)\} \text{ and} \\ \mathcal{T} &:= \{A \sqsubseteq \exists r.A, B \sqsubseteq P \sqcap \exists r.B\}.\end{aligned}$$

The ABox $\mathcal{A}$ itself can be viewed as an $\mathcal{EL}$ -description graph consisting of a single node a with label $\{P\}$. Since $A \sqsubseteq \exists r.A$ is in $\mathcal{T}$ and $A(a)$ is in $\mathcal{A}$, we extend this graph by an r -loop from a to a . Figure 1 shows the graph $\mathcal{G}$ obtained this way as well as the $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{T}}$ corresponding to $\mathcal{T}$.

Obviously, there is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}$ such that $(B, a) \in Z$. However, it is easy to see that a is not an instance of B . The reason for this problem is that node labels and edges in $\mathcal{G}_{\mathcal{T}}$ state facts that must hold for *all* individuals that are instances of the defined concept labeling a given node whereas assertions of the ABox make statements about properties of *particular* named individuals. The construction of $\mathcal{G}$ in the above example mixes these different things, and thus leads to unfounded conclusions.

In order to separate edges and labels coming from ABox assertions from the ones coming from TBox definitions, we do not “identify” the node a with the node A if $A(a)$ belongs to $\mathcal{A}$ (as done in the construction of $\mathcal{G}$ above). Instead, we do a “one-step expansion” of the definition of A . The right-most graph in Figure 1 shows the graph $\mathcal{G}_{\mathcal{A}}$ obtained this way in our example. Obviously, there is no simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{A}}$ such that $(B, a) \in Z$.

Below, we give a formal definition of the $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{A}}$ associated with the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$ in the general case.

Definition 15 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox, $\mathcal{A}$ an $\mathcal{EL}$ -ABox, and $\mathcal{G}_{\mathcal{T}} = (V, E, L)$ be the $\mathcal{EL}$ -description graph associated with $\mathcal{T}$. The $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{A}} = (V_{\mathcal{A}}, E_{\mathcal{A}}, L_{\mathcal{A}})$ associated with $\mathcal{A}$ and $\mathcal{T}$ is defined as follows:

- The nodes of $\mathcal{G}_{\mathcal{A}}$ are the individual names occurring in $\mathcal{A}$ together with the defined concepts of $\mathcal{T}$, i.e., $V_{\mathcal{A}} := V \cup \{a \mid a \text{ is an individual name in } \mathcal{A}\}$.
- The edges of $\mathcal{G}_{\mathcal{A}}$ are the edges of $\mathcal{G}$, the role assertions of $\mathcal{A}$, and additional edges linking the ABox individuals with defined concepts:

$$\begin{aligned}E_{\mathcal{A}} &:= E \cup \{(a, r, b) \mid r(a, b) \in \mathcal{A}\} \cup \\ &\quad \{(a, r, B) \mid A(a) \in \mathcal{A} \text{ and } (A, r, B) \in E\}.\end{aligned}$$

- if $u \in V_{\mathcal{A}}$ is a defined concept, then it inherits its label from $\mathcal{G}_{\mathcal{T}}$, i.e., $L_{\mathcal{A}}(u) := L(u)$ if $u \in V$. Otherwise,

$u \in V_{\mathcal{A}} \setminus V$ is an ABox individual, and then its label is derived from the concept assertions for u in $\mathcal{A}$:

$$L_{\mathcal{A}}(u) := \{P \mid P(u) \in \mathcal{A}\} \cup \bigcup_{A(u) \in \mathcal{A}} L(A).$$

Here P denotes primitive and A denotes defined concepts.

Before we can characterize the instance problem via the existence of certain simulation relations from $\mathcal{G}_{\mathcal{T}}$ to $\mathcal{G}_{\mathcal{A}}$, we must characterize under what conditions a gfp-model of $\mathcal{T}$ is a model of $\mathcal{A}$. In the following we assume that primitive interpretations $\mathcal{I}$ also interpret ABox individuals. We say that the simulation $Z: \mathcal{G}_{\mathcal{A}} \rightsquigarrow \mathcal{G}_{\mathcal{I}}$ respects ABox individuals iff $\{x \mid (a, x) \in Z\} = \{a^{\mathcal{I}}\}$ for all individual names a occurring in $\mathcal{A}$.

Proposition 16 Let $\mathcal{I}$ be a primitive interpretation and $\mathcal{I}$ the gfp-model of $\mathcal{T}$ based on $\mathcal{I}$. Then the following are equivalent:

1. $\mathcal{I}$ is a model of $\mathcal{A}$.
2. There is a simulation $Z: \mathcal{G}_{\mathcal{A}} \rightsquigarrow \mathcal{G}_{\mathcal{I}}$ that respects ABox individuals.

The following characterization of the instance problem is an easy consequence of this proposition and Proposition 11.

Theorem 17 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox, $\mathcal{A}$ an $\mathcal{EL}$ -ABox, A a defined concept in $\mathcal{T}$ and ‘ a ’ an individual name occurring in $\mathcal{A}$. Then the following are equivalent:

1. $\mathcal{A} \models_{\text{gfp}, \mathcal{T}} A(a)$.
2. There is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{A}}$ such that $(A, a) \in Z$.

The theorem together with Proposition 9 shows that the instance problem w.r.t. gfp-semantics in $\mathcal{EL}$ is tractable.

Corollary 18 The instance problem w.r.t. gfp-semantics in $\mathcal{EL}$ can be decided in polynomial time.

5.2 Computing the gfp-msc

Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox and $\mathcal{A}$ an $\mathcal{EL}$ -ABox containing the individual name a . Let $\mathcal{G}_{\mathcal{A}} = (V_{\mathcal{A}}, E_{\mathcal{A}}, L_{\mathcal{A}})$ be the $\mathcal{EL}$ -description graph corresponding to $\mathcal{A}$ and $\mathcal{T}_1$, as introduced in Definition 15. In order to obtain the gfp-msc of a , we view $\mathcal{G}_{\mathcal{A}}$ as the $\mathcal{EL}$ -description graph of an $\mathcal{EL}$ -TBox $\mathcal{T}_2$, i.e., let $\mathcal{T}_2$ be the TBox such that $\mathcal{G}_{\mathcal{A}} = \mathcal{G}_{\mathcal{T}_2}$. By the definition of $\mathcal{G}_{\mathcal{A}}$, the defined concepts of $\mathcal{T}_2$ are the defined concepts of $\mathcal{T}_1$ together with the individual names occurring in $\mathcal{A}$. It is easy to show that $\mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$. To avoid confusion we will denote the defined concept in $\mathcal{T}_2$ corresponding to the individual name b in $\mathcal{A}$ by C_b .

Using the results of the previous subsection, we can show [Baader, 2002] that C_a is the gfp-msc of a .

Lemma 19 The defined concept C_a in $\mathcal{T}_2$ is the gfp-msc of ‘ a ’ in $\mathcal{A}$ and $\mathcal{T}_1$.

Given $\mathcal{T}_1$ and $\mathcal{A}$, the graph $\mathcal{G}_{\mathcal{A}}$ can obviously be computed in polynomial time, and thus the gfp-msc can be computed in polynomial time.

Theorem 20 Let $\mathcal{T}_1$ be an $\mathcal{EL}$ -TBox and $\mathcal{A}$ an $\mathcal{EL}$ -ABox containing the individual name ‘ a ’. Then the gfp-msc of ‘ a ’ in $\mathcal{T}_1$ and $\mathcal{A}$ always exists, and it can be computed in polynomial time.

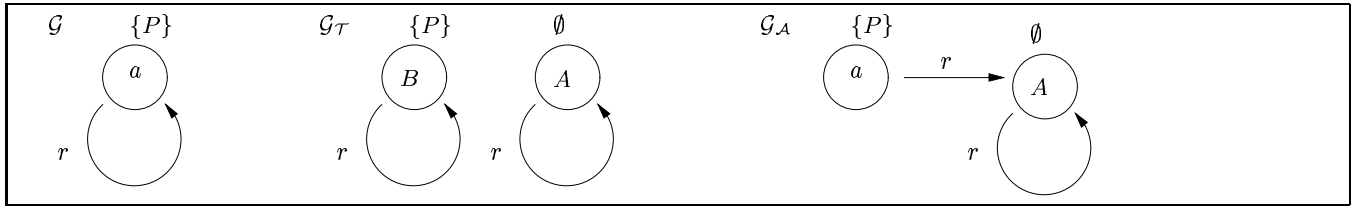


Figure 1: The $\mathcal{EL}$ -description graphs $\mathcal{G}$, $\mathcal{G}_T$, and $\mathcal{G}_A$ of our example.

6 Conclusion

In [Baader, 2003a] we have shown that subsumption in $\mathcal{EL}$ remains polynomial if one allows for cyclic terminologies with greatest fixpoint (gfp) semantics. In Section 5.1 of this paper we have complemented this result by showing that the instance problem in $\mathcal{EL}$ with cyclic terminologies interpreted with gfp-semantics is also polynomial. Thus, all the standard inferences in $\mathcal{EL}$ remain polynomial if one allows for cyclic terminologies with gfp-semantics. Our main motivation for considering cyclic terminologies with gfp-semantics in $\mathcal{EL}$ was that the most specific concept of an ABox-individual then always exists. In fact, we have shown in this paper that both the least common subsumer (lcs) and the most specific concept (msc) can be computed in polynomial time in $\mathcal{EL}$ with cyclic terminologies interpreted with gfp-semantics. Thus, also two of the most important non-standard inferences in DLs [Küsters, 2001] remain polynomial in this context.

It should be noted that there are indeed applications where the expressive power of the small DL $\mathcal{EL}$ appears to be sufficient. In fact, SNOMED, the Systematized Nomenclature of Medicine [Cote et al., 1993] uses $\mathcal{EL}$ [Spackman, 2000; 2001].

Subsumption [Baader, 2003a] and the instance problem [Baader, 2003b] are also polynomial w.r.t. descriptive semantics. For the lcs, descriptive semantics is not that well-behaved: in [Baader, 2003] we have shown that w.r.t. descriptive semantics the lcs need not exist in $\mathcal{EL}$ with cyclic terminologies. In addition, we could only give a sufficient condition for the existence of the lcs. If this condition applies, then the lcs can be computed in polynomial time. In [Baader, 2003b] similar results are shown for the msc w.r.t. descriptive semantics.

One problem left for future research is the question of how to obtain a decidable characterization of the cases in which the lcs (msc) exists w.r.t. descriptive semantics, and to determine whether in these cases it can always be computed in polynomial time.

References

- [Baader, 2002] F. Baader. Least common subsumers, most specific concepts, and role-value-maps in a description logic with existential restrictions and terminological cycles. LTCS-Report 02-07, TU Dresden, 2002. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [Baader, 2003] F. Baader. Computing the least common subsumer in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proc. ICCS'03*, Springer LNAI, 2003.
- [Baader, 2003a] F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. IJCAI'03*, 2003.
- [Baader, 2003b] Franz Baader. The instance problem and the most specific concept in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. LTCS-Report 03-01, TU Dresden, 2003. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [Baader and Küsters, 1998] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$ -concept descriptions. In *Proc. KI'98*, Springer LNAI 1504, 1998.
- [Baader et al., 1999] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. IJCAI'99*, 1999.
- [Cote et al., 1993] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.
- [Henzinger et al., 1995] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science*, 1995. IEEE Computer Society Press.
- [Küsters, 2001] R. Küsters. *Non-standard Inferences in Description Logics*, Springer LNAI 2100, 2001.
- [Küsters and Molitor, 2001] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. In *Proc. KI 2001*, Springer LNAI 2174, 2001.
- [Nebel, 1991] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, 1991.
- [Spackman, 2000] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue.
- [Spackman, 2001] K. Spackman. Normal forms for description logic expressions of clinical concepts in SNOMED RT. *J. of the American Medical Informatics Association*, 2001. Symposium Supplement.
- [Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

Terminological Cycles in a Description Logic with Existential Restrictions*

Franz Baader

Theoretical Computer Science

TU Dresden

D-01062 Dresden, Germany

baader@inf.tu-dresden.de

Abstract

Cyclic definitions in description logics have until now been investigated only for description logics allowing for value restrictions. Even for the most basic language $\mathcal{FL}_0$, which allows for conjunction and value restrictions only, deciding subsumption in the presence of terminological cycles is a PSPACE-complete problem. This paper investigates subsumption in the presence of terminological cycles for the language $\mathcal{EL}$, which allows for conjunction, existential restrictions, and the top-concept. In contrast to the results for $\mathcal{FL}_0$, subsumption in $\mathcal{EL}$ remains polynomial, independent of whether we use least fixpoint semantics, greatest fixpoint semantics, or descriptive semantics.

1 Introduction

Early description logic (DL) systems allowed the use of value restrictions ($\forall r.C$), but not of existential restrictions ($\exists r.C$). Thus, one could express that all children are male using the value restriction $\forall \text{child.Male}$, but not that someone has a son using the existential restriction $\exists \text{child.Male}$. The main reason was that, when clarifying the logical status of property arcs in semantic networks and slots in frames, the decision was taken that arcs/slots should be read as value restrictions (see, e.g., [Nebel, 1990]). Once one considers more expressive DLs allowing for full negation, existential restrictions come in as the dual of value restrictions [Schmidt-Schauß and Smolka, 1991]. Thus, for historical reasons, DLs that allow for existential, but not for value restrictions, are largely unexplored. In the present paper, we investigate terminological cycles in the DL $\mathcal{EL}$, which allows for conjunction, existential restrictions, and the top-concept. In contrast to (even very inexpressive) DLs with value restrictions, subsumption in $\mathcal{EL}$ remains polynomial in the presence of terminological cycles. It should be noted that there are indeed applications where the small DL $\mathcal{EL}$ appears to be sufficient. In fact, SNOMED, the Systematized Nomenclature of Medicine [Cote *et al.*, 1993] employs $\mathcal{EL}$ [Spackman, 2001; Spackman *et al.*, 1997]. Even though SNOMED does not appear to use cyclic definitions, this may be due to a lack

of technology rather than need. In fact, the Galen medical knowledge base contains many cyclic dependencies [Rector and Horrocks, 1997]. Also, even in the case of acyclic terminologies, our polynomial subsumption algorithm improves on the usual approach that first unfolds the TBox (a potentially exponential step) and then applies the polynomial subsumption algorithm for $\mathcal{EL}$ -concept descriptions [Baader *et al.*, 1999].

The first thorough investigation of cyclic terminologies in description logics (DL) is due to Nebel [1991], who introduced three different semantics for such terminologies: least fixpoint (lfp) semantics, which considers only the models that interpret the defined concepts as small as possible; greatest fixpoint (gfp) semantics, which considers only the models that interpret the defined concepts as large as possible; and descriptive semantics, which considers all models.

In [Baader, 1990; 1996], subsumption w.r.t. cyclic terminologies in the small DL $\mathcal{FL}_0$, which allows for conjunction and value restrictions only, was characterized with the help of finite automata. This characterization provided PSPACE decision procedures for subsumption in $\mathcal{FL}_0$ with cyclic terminologies for the three types of semantics introduced by Nebel. In addition, it was shown that subsumption is PSPACE-hard. The results for cyclic $\mathcal{FL}_0$ -terminologies were extended by Küsters [1998] to $\mathcal{ALN}$, which extends $\mathcal{FL}_0$ by atomic negation and number restrictions.

The fact that the DL $\mathcal{ALC}$ (which extends $\mathcal{FL}_0$ by full negation) is a syntactic variant of the multi-modal logic K opens a way for treating cyclic terminologies and more general recursive definitions in more expressive languages like $\mathcal{ALC}$ and extensions thereof by a reduction to the modal μ -calculus [Schild, 1994; De Giacomo and Lenzerini, 1994]. In this setting, one can use a mix of the three types of semantics introduced by Nebel. However, the complexity of the subsumption problem is EXPTIME-complete.

In spite of these very general results for cyclic definitions in expressive languages, there are still good reasons to look at cyclic terminologies in less expressive (in particular sub-Boolean) description logics. One reason is, of course, that one can hope for a lower complexity of the subsumption problem. For DLs with value restrictions, this hope is not fulfilled, though. Even in the inexpressive DL $\mathcal{FL}_0$, subsumption becomes PSPACE-complete if one allows for cyclic definitions. This is still better than the EXPTIME-completeness that one

*Partially supported by the DFG under grant BA 1122/4-3.

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top-concept	$\top$	$\Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
exist. restriction	$\exists r.C$	$\{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
concept definition	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}$.

has in $\mathcal{ALC}$ with cyclic definitions, but from the practical point of view it still means that the subsumption algorithm may need exponential time.

In contrast, the subsumption problem in $\mathcal{EL}$ can be decided in polynomial time w.r.t. the three types of semantics introduced by Nebel. The main tool used to show these results is a characterization of subsumption through the existence of so-called simulation relations.

In the next section we will introduce the DL $\mathcal{EL}$ as well as cyclic terminologies and the three types of semantics for these terminologies. Then we will show in Section 3 how such terminologies can be translated into description graphs. In this section, we will also define the notion of a simulation between nodes of a description graph, and mention some useful properties of simulations. The next three sections are then devoted to the characterization of subsumption in $\mathcal{EL}$ w.r.t. gfp, lfp, and descriptive semantics, respectively.

2 Cyclic terminologies in the DL $\mathcal{EL}$

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names*. The constructors determine the expressive power of the DL. We restrict the attention to the DL $\mathcal{EL}$, whose concept descriptions are formed using the constructors top-concept ($\top$), conjunction ($C \sqcap D$), and existential restriction ($\exists r.C$). The semantics of $\mathcal{EL}$ -concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and each role $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

A *terminology* (or *TBox* for short) is a finite set of concept definitions of the form $A \equiv D$, where A is a concept name and D a concept description. In addition, we require that TBoxes do not contain *multiple definitions*, i.e., there cannot be two distinct concept descriptions D_1 and D_2 such that both $A \equiv D_1$ and $A \equiv D_2$ belongs to the TBox. Concept names occurring on the left-hand side of a definition are called *defined concepts*. All other concept names occurring in the TBox are called *primitive concepts*. Note that we allow for cyclic dependencies between the defined concepts, i.e., the definition of A may refer (directly or indirectly) to A itself. An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all definitions $A \equiv D$ in $\mathcal{T}$.

The semantics of (possibly cyclic) $\mathcal{EL}$ -TBoxes we have just defined is called *descriptive semantic* by Nebel [1991].

For some applications, it is more appropriate to interpret cyclic concept definitions with the help of an appropriate fixpoint semantics. Before defining least and greatest fixpoint semantics formally, let us illustrate their effect on an example.

Example 1 Assume that our interpretations are graphs where we have nodes (elements of the concept name Node) and edges (represented by the role edge), and we want to define the concept lnode of all nodes lying on an infinite (possibly cyclic) path of the graph. The following is a possible definition of lnode: $\text{lnode} \equiv \text{Node} \sqcap \exists \text{edge}.\text{lnode}$.

Now consider the following interpretation of the primitive concepts and roles:

$$\begin{aligned} \Delta^{\mathcal{J}} &:= \{m_0, m_1, m_2, \dots\} \cup \{n_0\}, \\ \text{Node}^{\mathcal{J}} &:= \Delta^{\mathcal{J}}, \\ \text{edge}^{\mathcal{J}} &:= \{(m_i, m_{i+1}) \mid i \geq 0\} \cup \{(n_0, n_0)\}. \end{aligned}$$

Where $M := \{m_0, m_1, m_2, \dots\}$ and $N := \{n_0\}$, there are four possible ways of extending this interpretation of the primitive concepts and roles to a model of the TBox consisting of the above concept definition: lnode can be interpreted by $M \cup N$, M , N , or $\emptyset$. All these models are admissible w.r.t. descriptive semantics, whereas the first is the gfp-model and the last is the lfp-model of the TBox. Obviously, only the gfp-model captures the intuition underlying the definition (namely, nodes lying on an infinite path) correctly.

It should be noted, however, that in other cases descriptive semantics appears to be more appropriate. For example, consider the definitions

$$\begin{aligned} \text{Tiger} &\equiv \text{Animal} \sqcap \exists \text{parent}.\text{Tiger} \quad \text{and} \\ \text{Lion} &\equiv \text{Animal} \sqcap \exists \text{parent}.\text{Lion}. \end{aligned}$$

With respect to gfp-semantics, the defined concepts Tiger and Lion must always be interpreted as the same set whereas this is not the case for descriptive semantics.¹

Before we can define lfp- and gfp-semantics formally, we must introduce some notation. Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox containing the roles N_{role} , the primitive concepts N_{prim} , and the defined concepts $N_{def} := \{A_1, \dots, A_k\}$. A *primitive interpretation* $\mathcal{J}$ for $\mathcal{T}$ is given by a domain $\Delta^{\mathcal{J}}$, an interpretation of the roles $r \in N_{role}$ by binary relations $r^{\mathcal{J}}$ on $\Delta^{\mathcal{J}}$, and an interpretation of the primitive concepts $P \in N_{prim}$ by subsets $P^{\mathcal{J}}$ of $\Delta^{\mathcal{J}}$. Obviously, a primitive interpretation differs from an interpretation in that it does not interpret the defined concepts in N_{def} . We say that the interpretation $\mathcal{I}$ is *based on* the primitive interpretation $\mathcal{J}$ iff it has the same domain as $\mathcal{J}$ and coincides with $\mathcal{J}$ on N_{role} and N_{prim} . For a fixed primitive interpretation $\mathcal{J}$, the interpretations $\mathcal{I}$ based on it are uniquely determined by the tuple $(A_1^{\mathcal{I}}, \dots, A_k^{\mathcal{I}})$ of the interpretations of the defined concepts in N_{def} . We define

$$\text{Int}(\mathcal{J}) := \{\mathcal{I} \mid \mathcal{I} \text{ is an interpretation based on } \mathcal{J}\}.$$

Interpretations based on $\mathcal{J}$ can be compared by the following ordering, which realizes a pairwise inclusion test between the

¹This example is similar to the “humans and horses” example used by Nebel [1991] to illustrate the difference between descriptive semantics and gfp-semantics in $\mathcal{ALN}$.

respective interpretations of the defined concepts: if $\mathcal{I}_1, \mathcal{I}_2 \in \text{Int}(\mathcal{J})$, then

$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \text{ iff } A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2} \text{ for all } i, 1 \leq i \leq k.$$

It is easy to see that $\preceq_{\mathcal{J}}$ induces a *complete lattice* on $\text{Int}(\mathcal{J})$, i.e., every subset of $\text{Int}(\mathcal{J})$ has a least upper bound (lub) and a greatest lower bound (glb). Using *Tarski's fixpoint theorem* [Tarski, 1955] for complete lattices, it is not hard to show [Nebel, 1991] that, for a given primitive interpretation $\mathcal{J}$, there is always a greatest and a least (w.r.t. $\preceq_{\mathcal{J}}$) model of $\mathcal{T}$ based on $\mathcal{J}$. We call these models respectively the greatest fixpoint model (gfp-model) and the least fixpoint model (lfp-model) of $\mathcal{T}$. *Greatest (least) fixpoint semantics* considers only GFP-models (LFP-models) as admissible models.

Definition 2 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and A, B be defined concepts² occurring in $\mathcal{T}$. Then,

- A is subsumed by B w.r.t. descriptive semantics ($A \sqsubseteq_{\mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.
- A is subsumed by B w.r.t. GFP-semantics ($A \sqsubseteq_{\text{gfp}, \mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all GFP-models $\mathcal{I}$ of $\mathcal{T}$.
- A is subsumed by B w.r.t. LFP-semantics ($A \sqsubseteq_{\text{lfp}, \mathcal{T}} B$) iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all LFP-models $\mathcal{I}$ of $\mathcal{T}$.

We will show in the following that all three subsumption problems are decidable in polynomial time. To do this, we represent $\mathcal{EL}$ -TBoxes as graphs.

3 Description graphs and simulations

$\mathcal{EL}$ -TBoxes as well as primitive interpretations can be represented as description graphs. Before we can translate $\mathcal{EL}$ -TBoxes into description graphs, we must normalize them. In the following, let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox, N_{def} the defined concepts of $\mathcal{T}$, N_{prim} the primitive concepts of $\mathcal{T}$, and N_{role} the roles of $\mathcal{T}$.

We say that the $\mathcal{EL}$ -TBox $\mathcal{T}$ is *normalized* iff $A \equiv D \in \mathcal{T}$ implies that D is of the form

$$P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_\ell.B_\ell,$$

for $m, \ell \geq 0$, $P_1, \dots, P_m \in N_{\text{prim}}$, $r_1, \dots, r_\ell \in N_{\text{role}}$, and $B_1, \dots, B_\ell \in N_{\text{def}}$. If $m = \ell = 0$, then $D = \top$.

Since there is a polynomial translation of general TBoxes into normalized ones [Baader, 2002], one can restrict the attention to normalized TBoxes. Thus, we will assume that all TBoxes are normalized. Normalized $\mathcal{EL}$ -TBoxes can be viewed as graphs whose nodes are the defined concepts, which are labeled by sets of primitive concepts, and whose edges are given by the existential restrictions. For the rest of this section, we fix a normalized $\mathcal{EL}$ -TBox $\mathcal{T}$ with primitive concepts N_{prim} , defined concepts N_{def} , and roles N_{role} .

Definition 3 An $\mathcal{EL}$ -description graph is a graph $\mathcal{G} = (V, E, L)$ where

- V is a set of nodes;

²Obviously, we can restrict the attention to subsumption between defined concepts since subsumption between arbitrary concept descriptions can be reduced to this problem by introducing definitions for the descriptions.

- $E \subseteq V \times N_{\text{role}} \times V$ is a set of edges labeled by role names;
- $L: V \rightarrow 2^{N_{\text{prim}}}$ is a function that labels nodes with sets of primitive concepts.

The normalized TBox $\mathcal{T}$ can be translated into the following $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{T}} = (N_{\text{def}}, E_{\mathcal{T}}, L_{\mathcal{T}})$:

- the nodes of $\mathcal{G}_{\mathcal{T}}$ are the defined concepts of $\mathcal{T}$;
- if A is a defined concept and

$$A \equiv P_1 \sqcap \dots \sqcap P_m \sqcap \exists r_1.B_1 \sqcap \dots \sqcap \exists r_\ell.B_\ell$$

its definition in $\mathcal{T}$, then

- $L_{\mathcal{T}}(A) = \{P_1, \dots, P_m\}$, and
- A is the source of the edges $(A, r_1, B_1), \dots, (A, r_\ell, B_\ell) \in E_{\mathcal{T}}$.

Any primitive interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ can be translated into the following $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{J}} = (\Delta^{\mathcal{J}}, E_{\mathcal{J}}, L_{\mathcal{J}})$:

- the nodes of $\mathcal{G}_{\mathcal{J}}$ are the elements of $\Delta^{\mathcal{J}}$;
- $E_{\mathcal{J}} := \{(x, r, y) \mid (x, y) \in r^{\mathcal{J}}\}$;
- $L_{\mathcal{J}}(x) := \{P \in N_{\text{prim}} \mid x \in P^{\mathcal{J}}\}$ for all $x \in \Delta^{\mathcal{J}}$.

The translation between $\mathcal{EL}$ -TBoxes (primitive interpretations) and $\mathcal{EL}$ -description graphs works in both directions, i.e., any $\mathcal{EL}$ -description graph can also be view as an $\mathcal{EL}$ -TBox (primitive interpretation).

Simulations are binary relations between nodes of two $\mathcal{EL}$ -description graphs that respect labels and edges in the sense defined below.

Definition 4 Let $\mathcal{G}_i = (V_i, E_i, L_i)$ ($i = 1, 2$) be two $\mathcal{EL}$ -description graphs. The binary relation $Z \subseteq V_1 \times V_2$ is a *simulation* from $\mathcal{G}_1$ to $\mathcal{G}_2$ iff

- (S1) $(v_1, v_2) \in Z$ implies $L_1(v_1) \subseteq L_2(v_2)$; and
- (S2) if $(v_1, v_2) \in Z$ and $(v_1, r, v'_1) \in E_1$, then there exists a node $v'_2 \in V_2$ such that $(v'_1, v'_2) \in Z$ and $(v_2, r, v'_2) \in E_2$.

We write $Z: \mathcal{G}_1 \rightsquigarrow \mathcal{G}_2$ to express that Z is a simulation from $\mathcal{G}_1$ to $\mathcal{G}_2$.

It is easy to see that the set of all simulations from $\mathcal{G}_1$ to $\mathcal{G}_2$ is closed under arbitrary unions. Consequently, there always exists a greatest simulation from $\mathcal{G}_1$ to $\mathcal{G}_2$. If $\mathcal{G}_1, \mathcal{G}_2$ are finite, then this greatest simulation can be computed in polynomial time [Henzinger *et al.*, 1995]. The following proposition is an easy consequence of this fact (see [Baader, 2002]).

Proposition 5 Let $\mathcal{G}_1, \mathcal{G}_2$ be two finite $\mathcal{EL}$ -description graphs, v_1 a node of $\mathcal{G}_1$ and v_2 a node of $\mathcal{G}_2$. Then we can decide in polynomial time whether there is a simulation $Z: \mathcal{G}_1 \rightsquigarrow \mathcal{G}_2$ such that $(v_1, v_2) \in Z$.

4 Subsumption w.r.t. GFP-semantics

In the following, let $\mathcal{T}$ be a normalized $\mathcal{EL}$ -TBox with primitive concepts N_{prim} , defined concepts N_{def} , and roles N_{role} . Before characterizing subsumption w.r.t. GFP-semantics, we give a characterization of when an individual of a GFP-model belongs to a defined concept in this model.

Proposition 6 Let $\mathcal{I}$ be a primitive interpretation and $\mathcal{I}$ the gfp-model of $\mathcal{T}$ based on $\mathcal{I}$. Then the following are equivalent for any $A \in N_{def}$ and $x \in \Delta^{\mathcal{I}}$:

1. $x \in A^{\mathcal{I}}$.
2. There is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{I}}$ such that $(A, x) \in Z$.

This proposition (whose proof can be found in [Baader, 2002]), can now be used to prove the following characterization of subsumption w.r.t. gfp-semantics in $\mathcal{EL}$.

Theorem 7 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and A, B defined concepts in $\mathcal{T}$. Then the following are equivalent:

1. $A \sqsubseteq_{gfp, \mathcal{T}} B$.
2. There is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{T}}$ with $(B, A) \in Z$.

Proof. (2 $\Rightarrow$ 1) Assume that the simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{T}}$ satisfies $(B, A) \in Z$. Let $\mathcal{I}$ be a primitive interpretation and $\mathcal{I}$ the gfp-model of $\mathcal{T}$ based on $\mathcal{I}$. We must show that $x \in A^{\mathcal{I}}$ implies $x \in B^{\mathcal{I}}$.

By Proposition 6, $x \in A^{\mathcal{I}}$ implies that there is a simulation $Y: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{I}}$ such that $(A, x) \in Y$. It is easy to show that the composition $X := Z \circ Y$ is a simulation from $\mathcal{G}_{\mathcal{T}}$ to $\mathcal{G}_{\mathcal{I}}$ such that $(B, x) \in X$. By Proposition 6, this implies $x \in B^{\mathcal{I}}$.

(1 $\Rightarrow$ 2) Assume that $A \sqsubseteq_{gfp, \mathcal{T}} B$. We consider the graph $\mathcal{G}_{\mathcal{T}}$, and view it as an $\mathcal{EL}$ -description graph of a primitive interpretation. Thus, let $\mathcal{I}$ be the primitive interpretation with $\mathcal{G}_{\mathcal{T}} = \mathcal{G}_{\mathcal{I}}$, and let $\mathcal{I}$ be the gfp-model of $\mathcal{T}$ based on $\mathcal{I}$.

Since the identity is a simulation $Id: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{T}} = \mathcal{G}_{\mathcal{I}}$ that satisfies $(A, A) \in Id$, Proposition 6 yields $A \in A^{\mathcal{I}}$. But then $A \sqsubseteq_{gfp, \mathcal{T}} B$ implies $A \in B^{\mathcal{I}}$, and thus Proposition 6 yields the existence of a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{I}} = \mathcal{G}_{\mathcal{T}}$ such that $(B, A) \in Z$. $\square$

The theorem together with Proposition 5 shows that subsumption w.r.t. gfp-semantics in $\mathcal{EL}$ is tractable.

Corollary 8 Subsumption w.r.t. gfp-semantics in $\mathcal{EL}$ can be decided in polynomial time.

Example 9 Consider the TBox $\mathcal{T}$ consisting of the following concept definitions:

$$B \equiv \exists r.C, C \equiv \exists r.D, D \equiv \exists r.C, A \equiv \exists r.A', A' \equiv \exists r.D.$$

The $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{T}}$ corresponding to this TBox can be found in Figure 1. Let $V_{\mathcal{T}} = \{A, A', B, C, D\}$ denote the set of nodes of this graph. Then $Z := V_{\mathcal{T}} \times V_{\mathcal{T}}$ is a simulation from $\mathcal{G}_{\mathcal{T}}$ to $\mathcal{G}_{\mathcal{T}}$. Consequently, all the defined concepts in $\mathcal{T}$ subsume each other w.r.t. gfp-semantics.

5 Subsumption w.r.t. lfp-semantics

For the sake of completeness, we also treat lfp-semantics. It should be noted, however, that the results of this section demonstrate that lfp-semantics is not interesting in $\mathcal{EL}$.

Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and $\mathcal{G}_{\mathcal{T}}$ the corresponding $\mathcal{EL}$ -description graph. Where A, B are nodes of $\mathcal{G}_{\mathcal{T}}$, we write $A \xrightarrow{*}_{\mathcal{T}} B$ to denote that there is a path in $\mathcal{G}_{\mathcal{T}}$ from A to B , and $A \xrightarrow{+}_{\mathcal{T}} B$ to denote that there is a non-empty path in $\mathcal{G}_{\mathcal{T}}$ from A to B . We define the set $Cyc_{\mathcal{T}}$ as

$$\{A \mid \text{there exists a node } B \text{ such that } A \xrightarrow{*}_{\mathcal{T}} B \xrightarrow{+}_{\mathcal{T}} B\},$$

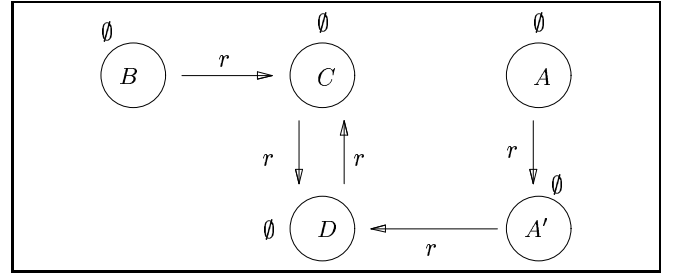


Figure 1: The $\mathcal{EL}$ -description graph $\mathcal{G}_{\mathcal{T}}$ of Example 9.

i.e., $Cyc_{\mathcal{T}}$ consists of the nodes in $\mathcal{G}_{\mathcal{T}}$ that can reach a cyclic path in $\mathcal{G}_{\mathcal{T}}$. The following proposition is an easy consequence of the definition of lfp-semantics and of $Cyc_{\mathcal{T}}$ (see [Baader, 2002]).

Proposition 10 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and A a defined concept in $\mathcal{T}$. If $A \in Cyc_{\mathcal{T}}$, then A is unsatisfiable w.r.t. lfp-semantics, i.e., $A^{\mathcal{I}} = \emptyset$ holds for all lfp-models $\mathcal{I}$ of $\mathcal{T}$.

In Example 9, all the defined concepts belong to $Cyc_{\mathcal{T}}$, and thus they are all unsatisfiable w.r.t. lfp-semantics.

Since all the defined concepts in $Cyc_{\mathcal{T}}$ are unsatisfiable, their definitions can be removed from the TBox without changing the meaning of the concepts not belonging to $Cyc_{\mathcal{T}}$. (Their definition cannot refer to an element of $Cyc_{\mathcal{T}}$.) This leaves us with an acyclic terminology, on which gfp- and lfp-semantics coincide [Nebel, 1991]. Thus, subsumption w.r.t. lfp-semantics in $\mathcal{EL}$ can be reduced to subsumption w.r.t. gfp-semantics.

Corollary 11 Subsumption w.r.t. lfp-semantics in $\mathcal{EL}$ can be decided in polynomial time.

6 Subsumption w.r.t. descriptive semantics

Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox and $\mathcal{G}_{\mathcal{T}}$ the corresponding $\mathcal{EL}$ -description graph. Since every gfp-model of $\mathcal{T}$ is a model of $\mathcal{T}$, $A \sqsubseteq_{\mathcal{T}} B$ implies $A \sqsubseteq_{gfp, \mathcal{T}} B$. Consequently, $A \sqsubseteq_{\mathcal{T}} B$ implies that there is a simulation $Z: \mathcal{G}_{\mathcal{T}} \rightsquigarrow \mathcal{G}_{\mathcal{T}}$ with $(B, A) \in Z$. In the following we will show what additional properties the simulation Z must satisfy for the implication in the other direction to hold.

To get an intuition on the difference between gfp- and descriptive semantics, let us consider Example 9. With respect to gfp-semantics, all the defined concepts of $\mathcal{T}$ are equivalent (i.e., subsume each other). With respect to descriptive semantics, A, B, D are still equivalent, C is equivalent to A' , but A' is not equivalent to B , and C and D are also not equivalent (in both cases, the concepts are not even comparable w.r.t. subsumption).

To see that C and A' are equivalent w.r.t. descriptive semantics, it is enough to note that the following identities hold in every model $\mathcal{I}$ of $\mathcal{T}$: $A'^{\mathcal{I}} = (\exists r.D)^{\mathcal{I}} = C^{\mathcal{I}}$. A similar argument shows that B and D are equivalent. In addition, equivalence of C and A' also implies equivalence of A and B . The following model of $\mathcal{T}$ is a counterexample to the other subsumption relationships:

1. $\Delta^{\mathcal{I}} := \{c, d\}$;

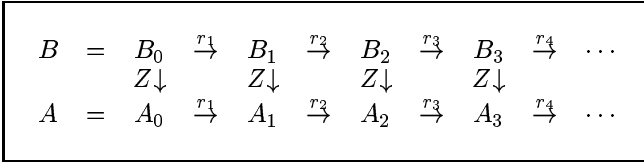


Figure 2: A (B, A) -simulation chain.

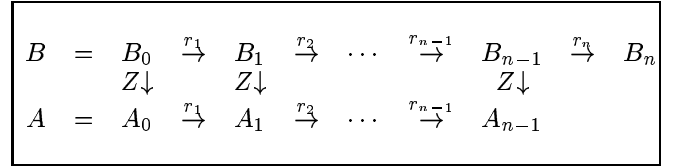


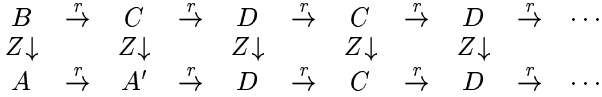
Figure 3: A partial (B, A) -simulation chain.

2. $r^I := \{(c, d), (d, c)\}$;
3. $A^I := B^I := D^I := \{d\}$, $A'^I := C^I := \{c\}$.

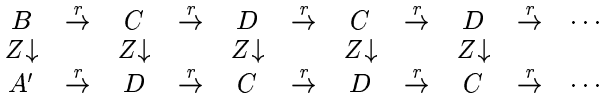
We will see below that the reason for A' and B not being equivalent is that in the infinite path in $\mathcal{G}_T$ starting with A' , one reaches D with an odd number of edges, whereas C is reached with an even number; for the path starting with B , it is just the opposite. In contrast, the infinite paths starting respectively with A and B “synchronize” after a finite number of steps. To formalize this intuition, we must introduce some notation.

Definition 12 The path $p_1: B = B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_3} B_2 \xrightarrow{r_3} B_3 \xrightarrow{r_4} \dots$ in $\mathcal{G}_T$ is Z -simulated by the path $p_2: A = A_0 \xrightarrow{r_1} A_1 \xrightarrow{r_3} A_2 \xrightarrow{r_3} A_3 \xrightarrow{r_4} \dots$ in $\mathcal{G}_T$ iff $(B_i, A_i) \in Z$ for all $i \geq 0$. In this case we say that the pair (p_1, p_2) is a (B, A) -simulation chain w.r.t. Z (see Figure 2).

Consider the TBox $\mathcal{T}$ and the simulation Z introduced in Example 9. Then



is a (B, A) -simulation chain w.r.t. Z , and



is a (B, A') -simulation chain w.r.t. Z . Note that the first chain synchronizes after a finite number of steps in the sense that there is a Z -link between the same defined concept. In contrast, the second chain does not synchronize in this sense.

If $(B, A) \in Z$, then (S2) of Definition 4 implies that, for every infinite path p_1 starting with $B_0 := B$, there is an infinite path p_2 starting with $A_0 := A$ such that p_1 is Z -simulated by p_2 . In the following we construct such a simulating path step by step. The main point is, however, that the decision which concept A_n to take in step n should depend only on the partial (B, A) -simulation chain already constructed, and not on the parts of the path p_1 not yet considered.

Definition 13 A *partial (B, A) -simulation chain* is of the form depicted in Figure 3. A *selection function* S for A, B and Z assigns to each partial (B, A) -simulation chain of this form a defined concept A_n such that (A_{n-1}, r_n, A_n) is an edge in $\mathcal{G}_T$ and $(B_n, A_n) \in Z$.

Given a path $B = B_0 \xrightarrow{r_1} B_1 \xrightarrow{r_3} B_2 \xrightarrow{r_3} B_3 \xrightarrow{r_4} \dots$ and a defined concept A such that $(B, A) \in Z$, one can use a selection function S for A, B and Z to construct a Z -simulating path. In this case we say that the resulting (B, A) -simulation chain is S -selected.

Definition 14 The simulation $Z: \mathcal{G}_T \xrightarrow{\sim} \mathcal{G}_T$ is called (B, A) -synchronized iff there exists a selection function S for A, B and Z such that the following holds: for every infinite S -selected (B, A) -simulation chain of the form depicted in Figure 2 there exists an $i \geq 0$ such that $A_i = B_i$.

We are now ready to state our characterization of subsumption w.r.t. descriptive semantics (see [Baader, 2002] for the proof).

Theorem 15 Let $\mathcal{T}$ be an $\mathcal{EL}$ -TBox, and A, B defined concepts in $\mathcal{T}$. Then the following are equivalent:

1. $A \sqsubseteq_{\mathcal{T}} B$.
2. There is a (B, A) -synchronized simulation $Z: \mathcal{G}_T \xrightarrow{\sim} \mathcal{G}_T$ such that $(B, A) \in Z$.

It remains to be shown that property (2) of the theorem can be decided in polynomial time. To this purpose, we construct a simulation Y such that (2) of Theorem 15 is equivalent to $(B, A) \in Y$ (see [Baader, 2002] for the proof that this is indeed the case):

We define $Y := \bigcup_{n \geq 0} Y_n$, where the relations Y_n are defined by induction on n : Y_0 is the identity on the nodes of $\mathcal{G}_T$. If Y_n is already defined, then

$Y_{n+1} := Y_n \cup \{(C, C') \mid (C, C') \text{ satisfies (1), (2), (3) below}\}.$

1. $L_{\mathcal{T}}(C) \subseteq L_{\mathcal{T}}(C')$,
2. $(C, r_1, C_1), \dots, (C, r_\ell, C_\ell)$ are all the edges in $\mathcal{G}_T$ with source C ,
3. there are edges $(C', r_1, C'_1), \dots, (C', r_\ell, C'_\ell)$ in $\mathcal{G}_T$ such that $(C_1, C'_1) \in Y_n, \dots, (C_\ell, C'_\ell) \in Y_n$.

The relation Y can obviously be computed in time polynomial in the size of $\mathcal{G}_T$. By using the techniques employed to decide Horn-SAT in linear time [Dowling and Gallier, 1984], it is not hard to show that Y can actually be computed in time quadratic in the size of $\mathcal{G}_T$.

Corollary 16 Subsumption w.r.t. descriptive semantics in $\mathcal{EL}$ can be decided in polynomial time.

An alternative way for showing the polynomiality result would be to reduce the existence of a (B, A) -synchronized simulation Z satisfying $(B, A) \in Z$ to the strategy problem for a certain two-player game with a positional winning condition. The existence of a winning strategy is in this case a polynomial time problem [Grädel, 2002].

7 Future and related work

We have seen that subsumption in $\mathcal{EL}$ with cyclic terminologies is polynomial for the three types of semantics introduced

by Nebel [1991]. In some applications, it would be interesting to have a mix of all three semantics, and it remains to be seen whether the polynomiality results also hold in such a setting (which would correspond to a restriction of the modal μ -calculus [Kozen, 1983]).

Sub-Boolean DLs (like $\mathcal{EL}$) have attracted renewed attention in the context of so-called non-standard inferences [Küsters, 2001] like computing the least common subsumer and the most specific concept. In [Baader, 2003] we have shown that the characterization of subsumption in $\mathcal{EL}$ w.r.t. gfp-semantics also yields an approach for computing the least common subsumer in $\mathcal{EL}$ w.r.t. gfp-semantics. In addition, we have extended the characterization of subsumption in $\mathcal{EL}$ w.r.t. gfp-semantics to the instance problem, and have shown how this can be used to compute the most specific concept.

Simulations and bisimulations play an important rôle in modal logics (and thus also in description logics). However, until now they have mostly been considered for modal logics that are closed under all the Boolean operators, and they have usually not been employed for reasoning in the logic. A notable exception are [Kurtonina and de Rijke, 1997; 1999], where bisimulation characterizations are given for sub-Boolean Modal Logics and DLs. However, these characterizations are used to give a formal account of the expressive power of these logics. They are not employed for reasoning purposes.

In [Baader *et al.*, 1999], subsumption between $\mathcal{EL}$ -concept descriptions was characterized through the existence of homomorphisms between the description trees (basically the syntax trees) associated with the descriptions. If one looks at the polynomial time algorithm for deciding the existence of such a homomorphism, then it is easy to see that it actually computes the greatest simulation between the trees. For trees, the existence of a homomorphism mapping the root to the root coincides with the existence of a simulation containing the tuple of the roots. For graphs, a similar connection does not hold. In fact, for graphs the existence of a homomorphism is an NP-complete problem. For simple conceptual graphs (or equivalently, conjunctive queries) the implication (containment) problem can be characterized via the existence of certain homomorphisms between graphs [Chein *et al.*, 1998], and is thus NP-complete.

References

- [Baader, 2003] F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proc. IJCAI'03*, 2003.
- [Baader, 2002] F. Baader. Terminological cycles in a description logic with existential restrictions. LTCS-Report 02-02, TU Dresden, 2002. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [Baader *et al.*, 1999] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. IJCAI'99*, 1999.
- [Baader, 1990] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. AAAI'90*, 1990.
- [Baader, 1996] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Mathematics and Artificial Intelligence*, 18, 1996.
- [Chein *et al.*, 1998] M. Chein, M.-L. Mugnier, and G. Simonet. Nested graphs: A graph-based knowledge representation model with FOL semantics. In *Proc. KR'98*, 1998.
- [Cote *et al.*, 1993] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.
- [De Giacomo and Lenzerini, 1994] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ -calculus. In *Proc. ECAI'94*, 1994.
- [Dowling and Gallier, 1984] W. F. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Programming*, 1(3), 1984.
- [Grädel, 2002] E. Grädel. Model checking games. In *Proc. WOL-LIC 02*, volume 67 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.
- [Henzinger *et al.*, 1995] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science*, 1995. IEEE Computer Society Press.
- [Kozen, 1983] D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27, 1983.
- [Kurtonina and de Rijke, 1997] N. Kurtonina and M. de Rijke. Simulating without negation. *J. Logic and Computation*, 7, 1997.
- [Kurtonina and de Rijke, 1999] N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107(2), 1999.
- [Küsters, 1998] R. Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. In *Proc. KR'98*, 1998.
- [Küsters, 2001] R. Küsters. *Non-standard Inferences in Description Logics*, Springer LNAI 2100, 2001.
- [Nebel, 1990] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Springer LNAI 422, 1990.
- [Nebel, 1991] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufmann, Los Altos, 1991.
- [Rector and Horrocks, 1997] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the WS on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, 1997.
- [Schild, 1994] K. Schild. Terminological cycles and the propositional μ -calculus. In *Proc. KR'94*, 1994.
- [Schmidt-Schauß and Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
- [Spackman *et al.*, 1997] K. Spackman, K. Campbell, and R. Cote. SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, 1997. Fall Symposium Supplement.
- [Spackman, 2001] K. Spackman. Normal forms for description logic expressions of clinical concepts in SNOMED RT. *J. of the American Medical Informatics Association*, 2001. Symposium Supplement.
- [Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Mathematics*, 5:285–309, 1955.

On the Undecidability of Description and Dynamic Logics with Recursion and Counting

Piero A. Bonatti

Università di Napoli “Federico II”

bonatti@na.infn.it

Abstract

The evolution of Description Logics (DLs) and Propositional Dynamic Logics produced a hierarchy of decidable logics with multiple maximal elements. It would be desirable to combine different maximal logics into one super-logic, but then inference may turn out to be undecidable. Then it is important to characterize the decidability threshold for these logics. In this perspective, an interesting open question pointed out by Sattler and Vardi [Sattler and Vardi, 1999] is whether inference in a hybrid μ -calculus with restricted forms of graded modalities is decidable, and which complexity class it belongs to. In this paper we prove that this calculus and the corresponding DL μALCCIO_f are undecidable. Second, we prove undecidability results for logics that support both a transitive closure operator over roles and number restrictions.

Keywords: Description logics, hybrid μ -calculus, regular roles, graded modalities, number restrictions.

1 Introduction

Description logics are popular knowledge representation languages, with important applications to the semantic web, software engineering and heterogeneous databases. Description logics (DLs) are strictly related to propositional dynamic logics (PDLs), that play an important role in software and protocol verification based on automated reasoning techniques. The analogies between the two frameworks are so tight that DLs and PDLs can be regarded as syntactic variants of the same family of logics.

The simplest DLs and PDLs can be easily embedded into a fragment of L^2 , that is, first-order logic with two variables. Application requirements led researchers to extend these basic logics with more expressive constructs, such as *fixpoints*, *nominals* (that represent individuals in DLs), transitive closure operators similar to Kleene’s star, and equivalents of generalized quantification called *number restrictions* (or *counting*) in DLs and *graded modalities* in PDLs. At the same time, applications require these logics to be decidable and have acceptable computational complexity.

The evolution of DLs and PDLs produced—and keeps on extending—a hierarchy of decidable logics with multiple maximal elements. Currently, two of the maximal decidable DLs are μALCCIO (featuring fixpoints and nominals [Sattler and Vardi, 1999; Bonatti, 2002]) and μALCCIQ (featuring fixpoints and number restrictions). The corresponding PDLs are the *hybrid μ -calculus* and the *μ -calculus with graded modalities*, respectively [Sattler and Vardi, 1999; Kupferman *et al.*, 2002].

Of course, it would be desirable to combine the features of different maximal logics into one super-logic. A combination of μALCCIO and μALCCIQ would help—for example—in describing the functional behavior of e-Services (cf. [Bonatti, 2002] and related comments on $\text{SDL}(\mathcal{X})$ in Section 5). However, in the super-logic inference may turn out to be too complex, and in particular undecidable.

A related, interesting open question pointed out by Sattler and Vardi [Sattler and Vardi, 1999] is whether inference in the union of the hybrid μ -calculus and the μ -calculus with graded modalities is decidable, and which complexity class it belongs to. More precisely, Sattler and Vardi mention a slightly simpler logic: a hybrid μ -calculus with *deterministic programs*. Deterministic programs are a special case of graded modality, whose counterpart in DLs are *features*, i.e., functional roles. The DL corresponding to the hybrid μ -calculus with deterministic programs is called μALCCIO_f .

The main contribution of this paper is a negative answer to the above open question. We prove that the hybrid μ -calculus with deterministic programs and the corresponding DL μALCCIO_f are undecidable. For this purpose, we use a novel approach based on nested fixpoints.

The second contribution is an undecidability result for logics that support number restrictions together with regular role expressions. These results show that transitive role closure can be more expressive than fixpoints in some contexts.

In the next section we recall the basic notions about DLs and the μ -calculus. Section 3 is devoted to the undecidability proof for μALCCIO_f and the hybrid μ -calculus with deterministic programs. Section 4 briefly illustrates the undecidability result for the combination of transitive closure and number restrictions. Finally, Section 5 concludes the paper with a brief discussion of the results and some directions for further research. Some proofs will be omitted because of space limitations.

2 Preliminaries

The vocabulary of the description logics we deal with in this paper is specified by the following disjoint sets of symbols: a set of atomic concepts **At**, a set of nominals **Nom**, a set of concept variables **Var**, and a set of atomic roles **AR**.

The set of *roles* is the smallest superset of **AR** such that if R, R' are roles then $R^-, R \sqcup R'$, and R^* are roles.

Let R be a role, $X \in \text{Var}$ and $n \in \mathbb{N}$. The set of *concepts* is the smallest superset of $\text{At} \cup \text{Nom} \cup \text{Var}$ such that if C, C', D are concepts, then $\neg C, C \sqcap D, \exists^{\leq n} R.C$, and $\mu X.C'$ are concepts, provided that all the free occurrences of X in C' lie within the scope of an even number of operators $\neg$ and $\exists^{\leq n}$.

Semantics is based on interpretations of the form $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a set of *individuals* and $\cdot^{\mathcal{I}}$ is an interpretation function mapping each $A \in \text{At} \cup \text{Nom}$ onto some $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $R \in \text{AR}$ onto some $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Furthermore, nominals must be mapped onto singletons. A *valuation on $\mathcal{I}$* is a function $\rho : \text{Var} \rightarrow \wp(\Delta^{\mathcal{I}})$. As usual, $\rho[X/S]$ denotes the valuation such that $\rho[X/S](X) = S$ and for all $Y \neq X$, $\rho[X/S](Y) = \rho(Y)$. The meaning of inverse roles is

$$(R^-)^{\mathcal{I}} = \{ \langle y, x \rangle \mid \langle x, y \rangle \in R^{\mathcal{I}} \},$$

while $(R \sqcup R')^{\mathcal{I}} = R^{\mathcal{I}} \cup R'^{\mathcal{I}}$, and $(R^*)^{\mathcal{I}}$ denotes the reflexive transitive closure of $R^{\mathcal{I}}$.

The meaning of compound concepts is determined by pairs $(\mathcal{I}, \rho)$. By $\# S$ we denote the cardinality of a set S .

$$\begin{aligned} (\neg C)_{\rho}^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C_{\rho}^{\mathcal{I}} & A_{\rho}^{\mathcal{I}} &= A^{\mathcal{I}} \quad (A \in \text{At}) \\ (C \sqcap D)_{\rho}^{\mathcal{I}} &= C_{\rho}^{\mathcal{I}} \cap D_{\rho}^{\mathcal{I}} & X_{\rho}^{\mathcal{I}} &= \rho(X) \quad (X \in \text{Var}) \\ (\exists^{\leq n} R.C)_{\rho}^{\mathcal{I}} &= \{ x \mid \# \{ y \mid \langle x, y \rangle \in R^{\mathcal{I}} \wedge y \in C_{\rho}^{\mathcal{I}} \} \leq n \} \\ (\mu X.C)_{\rho}^{\mathcal{I}} &= \bigcap \left\{ S \subseteq \Delta^{\mathcal{I}} \mid C_{\rho[X/S]}^{\mathcal{I}} \subseteq S \right\}. \end{aligned}$$

Subscript ρ will be sometimes omitted when it applies to a closed concept (i.e., such that all variables are bound by μ).

Other standard constructs can be derived from the above concepts. We use the symbol $\triangleq$ to define abbreviations.

$$\begin{aligned} C \sqcup D &\triangleq \neg(\neg C \sqcap \neg D) & \top &\triangleq A \sqcup \neg A \quad (A \in \text{At}) \\ \forall R.C &\triangleq \exists^{\leq 0} R.\neg C & \exists R.C &\triangleq \neg \forall R.\neg C \\ \exists^{\geq n} R.C &\triangleq \neg \exists^{\leq n-1} R.C & \nu X.C &\triangleq \neg \mu X.\neg C[X/\neg X] \end{aligned}$$

Here $C[X/\neg X]$ is the concept obtained from C by replacing all free occurrences of X with $\neg X$.

The syntactic restrictions on concept variables make every concept C monotonic with respect to its free variables. Then $\mu X.C(X)$ and $\nu X.C(X)$ denote exactly the least and the greatest fixpoints of $C(X)$, that can be characterized with the standard iterative constructions.

An *assertion* has the form $C \sqsubseteq D$ where C and D are closed concepts. It is satisfied by $\mathcal{I}$ (equivalently, $\mathcal{I}$ is a model of the assertion) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A *TBox* is a finite set of assertions. A TBox is satisfiable iff it has a model, that is, an $\mathcal{I}$ that satisfies all the assertions in the TBox. Symmetric pairs of assertions such as $C \sqsubseteq D$ and $D \sqsubseteq C$ will be abbreviated by $C \equiv D$. A TBox $\mathcal{T}$ entails $C \sqsubseteq D$ if every model of $\mathcal{T}$ satisfies $C \sqsubseteq D$.

The description logic $\mathcal{ALC}$ is a fragment of the logic described so far, supporting only atomic roles, $\neg$, $\sqcap$ and $\exists R.C$

(plus all the constructs definable from these). In $\mathcal{ALC}$, $\text{Nom} = \emptyset$.

By convention, the name of a description logic contains $\mathcal{ALC}$ if the logic extends $\mathcal{ALC}$. Moreover, the name contains an $\mathcal{I}$ if inverse roles (R^-) are supported, an $\mathcal{O}$ if $\text{Nom} \neq \emptyset$, a $\mathcal{Q}$ if number restrictions ($\exists^{\leq n} R.C$) are supported, and a μ if fixpoints are supported. For example, $\mathcal{ALC}\mathcal{I}\mathcal{O}$ denotes the extension of $\mathcal{ALC}$ with inverse roles and nominals. A subscript f indicates that roles may be declared to be functions. Note that such assertions are a special case of number restriction, as they can be expressed with axioms of the form $\top \sqsubseteq \exists^{\leq 1} R.\top$. If the logic supports role operators besides inversion, we list those operators as superscripts. For example $\mathcal{ALC}\mathcal{I}\mathcal{O}^{\sqcup,*}$ denotes the extension of $\mathcal{ALC}\mathcal{I}$ with role union and reflexive transitive closure.

Description logics can be regarded as variants of the propositional μ -calculus. Individuals correspond to possible worlds and roles correspond to accessibility relations. Atomic concepts play the role of propositional symbols. In particular, $\mu\mathcal{ALC}\mathcal{I}\mathcal{O}_f$ can be embedded into the hybrid μ -calculus with deterministic programs and graded modalities ($\langle n, P \rangle F$ and $[n, P]F$) via the following satisfiability-preserving translation. For all propositions p , and for all $n > 0$,

$$\begin{aligned} \varepsilon(p) &= p & \varepsilon(F \sqcap G) &= \varepsilon(F) \wedge \varepsilon(G) \\ \varepsilon(\neg F) &= \neg \varepsilon(F) & \varepsilon(\exists^{\geq n} P.F) &= \langle n-1, P \rangle \varepsilon(F) \\ \varepsilon(\mu X.F) &= \mu X.\varepsilon(F) & \varepsilon(C \sqsubseteq D) &= [o] \varepsilon(\neg C \sqcup D). \end{aligned}$$

Moreover, functional roles are mapped onto deterministic programs (whose accessibility relation is the graph of a function), and nominals are mapped onto their equivalents (called nominals, too), that in PDL terms are propositional symbols that are true in exactly one world. Program o denotes the universal program whose accessibility relation consists of all pairs of possible worlds. The reader is referred to [Sattler and Vardi, 1999; Kupferman *et al.*, 2002; De Giacomo, 1995] for further details.

3 Undecidability of $\mu\mathcal{ALC}\mathcal{I}\mathcal{O}_f$

This section is devoted to the proof of the following theorem.

Theorem 3.1 *In $\mu\mathcal{ALC}\mathcal{I}\mathcal{O}_f$, concept satisfiability, TBox satisfiability and entailment are all undecidable.*

We find it convenient to prove this theorem by first reducing domino problems to TBox satisfiability, and then extending this result to the other decision problems.

Recall that domino problems consist in placing tiles on an infinite grid, satisfying a given set of constraints on adjacent tiles. Formally, a domino problem is a structure $\mathcal{D} = \langle T, H, V \rangle$, where T is a set of *tile types* and $H, V \subseteq T^2$ specify which tiles can be adjacent horizontally and vertically, respectively. A *solution to $\mathcal{D}$* is a *tiling*, that is, a function $\tau : \mathbb{N}^2 \rightarrow T$, such that

1. if $\tau(x, y) = t$ and $\tau(x+1, y) = t'$ then $(t, t') \in H$, and
2. if $\tau(x, y) = t$ and $\tau(x, y+1) = t'$ then $(t, t') \in V$.

The existence of a solution for a given domino problem is known to be undecidable (cf. [Grädel *et al.*, 1999]).

Domino problems are reduced to reasoning problems by characterizing (i) the grid and (ii) correct tilings. Formally, the grid is a structure $\mathcal{G} = \langle \mathbb{N}^2, h^{\mathcal{G}}, v^{\mathcal{G}} \rangle$, where $h^{\mathcal{G}}(x, y) = (x + 1, y)$, and $v^{\mathcal{G}}(x, y) = (x, y + 1)$, for all $x, y \in \mathbb{N}$.

In description logics, $h^{\mathcal{G}}$ and $v^{\mathcal{G}}$ can be denoted by two roles, h and v . If the two roles characterize the grid correctly (see Figure 1(a)), then characterizing the solutions of a domino problem is easy, even within simple (and decidable) description logics such as $\mathcal{ALC}$, by means of the following assertion:

$$\top \sqsubseteq \left(\bigcup_{t \in T} C_t \right) \sqcap \left(\bigcap_{t \in T} \bigcap_{v' \in T \setminus \{t\}} \neg(C_t \sqcap C_{v'}) \right) \sqcap \left(\bigcap_{t \in T} \left[\neg C_t \sqcup \left(\exists h. \bigcup_{(t, t') \in H} C_{t'} \sqcap \exists v. \bigcup_{(t, t') \in V} C_{t'} \right) \right] \right).$$

Here for each tile type t , a distinct concept name C_t is introduced. Assertion (1) basically states that each individual is a tile (first term), that distinct tile types contain different tiles (rest of the first line), and that the tiling preserves the constraints specified by H and V (second line).

The real problem is characterizing the grid, because there is no direct way to force h and v to commute. Here we shall provide a projective characterization of $\mathcal{G}$, that is, we shall capture exactly the class of interpretations isomorphic to the expansion $\mathcal{G}^* = \langle \mathbb{N}^2, h^{\mathcal{G}}, v^{\mathcal{G}}, O^{\mathcal{G}^*} \rangle$ of $\mathcal{G}$, where O is a unary relation and $O^{\mathcal{G}^*} = \{(0, 0)\}$.

We proceed in three steps. First, h and v must be forced to be injective functions. For this purpose, we declare all roles and their converse to be functional, which is equivalent to adopting the following assertion, for all roles R .

$$\top \sqsubseteq (\exists \leq^1 R. \top) \sqcap (\exists \leq^1 R^-. \top). \quad (2)$$

Second, all nodes are classified with respect to their incoming and outgoing edges. Note that the domain and the range of a role R can be defined as follows.

$$\begin{aligned} \text{dom}(R) &\triangleq \exists R. \top \\ \text{range}(R) &\triangleq \exists R^-. \top \end{aligned}$$

Now we can define the vertical and the horizontal borders of the grid (B_v and B_h , respectively), and the internal nodes (C). O is a nominal that represents the origin of the grid.

$$\begin{aligned} B_v &\triangleq O \sqcup (\text{dom}(h) \sqcap \text{dom}(v) \sqcap \neg \text{range}(h) \sqcap \text{range}(v)) \\ B_h &\triangleq O \sqcup (\text{dom}(h) \sqcap \text{dom}(v) \sqcap \text{range}(h) \sqcap \neg \text{range}(v)) \\ C &\triangleq \text{dom}(h) \sqcap \text{dom}(v) \sqcap \text{range}(h) \sqcap \text{range}(v) \end{aligned}$$

The following assertions state the properties of O and force the above concepts to cover all the domain.

$$\begin{aligned} O &\sqsubseteq \text{dom}(h) \sqcap \text{dom}(v) \sqcap \neg \text{range}(h) \sqcap \neg \text{range}(v) \quad (3) \\ \top &\sqsubseteq B_v \sqcup B_h \sqcup C \quad (4) \end{aligned}$$

Third, we characterize the global structure of the grid. The next assertions ensure that the vertical border B_v and the horizontal border B_h have the desired structure (i.e., they should be isomorphic to $\mathbb{N}$).

$$\begin{aligned} B_v &\equiv \mu X [O \sqcup \exists v^-. X] \quad (5) \\ B_h &\equiv \mu X [O \sqcup \exists h^-. X]. \quad (6) \end{aligned}$$

Note that B_v and B_h contain no cycles, because of (2) and (3), so the two fixpoints induce infinite linear sequences of nodes.

Finally, we introduce an assertion that forces h and v to commute everywhere.

$$\begin{aligned} \top &\sqsubseteq \mu X. \xi(X), \text{ where} \quad (7) \\ \xi(X) &\triangleq O \sqcup \varphi_0(X) \sqcup \varphi_1(X) \\ \varphi_0(X) &\triangleq (\exists h. \exists v^-. X) \sqcap (\exists v^-. \exists h. X) \\ \varphi_1(X) &\triangleq B_h \sqcap \exists h^-. \mu Y. \varphi_2(X, Y) \\ \varphi_2(X, Y) &\triangleq (B_v \sqcup \exists v. \exists h^-. Y) \sqcap X \end{aligned}$$

Informally speaking, the constructive characterization of the fixpoint in (7) corresponds to a visit of the grid along diagonals directed north-west (Figure 1(b)). At each iteration, a new node x_0 is considered. Subformula $\varphi_0(X)$ ensures that x_0 is connected to the last visited element x_1 in such a way that h and v commute. Actually, h and v are not explicitly required to commute. They actually do (equation (9)) because each visited node $\bar{x}$ but the last one (i.e., x_1) must be connected by $v \circ h^-$ and $h^- \circ v$ to another visited node (Proposition 3.3.e), therefore, by (2), there exist no further links $v \circ h^-$ and $h^- \circ v$ connecting $\bar{x}$ to (the not yet visited node) x_0 . It follows that only x_1 can be connected to x_0 as specified by $\varphi_0(X)$. This makes h and v commute and ensures that x_0 is unique. Every time the vertical border B_v is reached, subformula $\varphi_1(X)$ adds the first element of the next diagonal. For this purpose, subformula $\mu Y. \varphi_2(X, Y)$ looks for diagonals entirely contained in X (the set of nodes visited so far) so that a new diagonal is not entered before the previous one has been completely visited. Figure 1(c) illustrates this phase. Let X_i be the current set of visited nodes. The black circles are the elements of $\varphi_1(X_i)$. In this example, $\mu Y. \varphi_2(X_i, Y)$ equals X_i , because the latter contains precisely the first 3 diagonals. In the following we formalize all the above intuitions.

Let $\mathcal{T}$ consist of the assertions (2), the local constraints (3) and (4), and the fixpoint assertions (5), (6) and (7). In the rest of this section, let $\mathcal{I}$ be a model of $\mathcal{T}$.

First we introduce some notation related to the domain elements of $\mathcal{I}$. Let $o \in O^{\mathcal{I}}$. For all $d, j \in \mathbb{N}$, let

$$e_{d,j} = (h^{-\mathcal{I}} \circ v^{\mathcal{I}})^j \circ (h^{\mathcal{I}})^d(o).$$

Note that $e_{d,j}$ is not necessarily defined, because $h^{-\mathcal{I}}$ is partial. Informally speaking, d counts diagonals and j is the displacement within a diagonal (cf. Figure 1(b)). Indexes are ordered lexicographically. Define

$$(d_1, j_1) \preceq (d_2, j_2) \text{ iff } d_1 < d_2 \text{ or } (d_1 = d_2 \text{ and } j_1 \leq j_2).$$

For all sets $S \subseteq \Delta^{\mathcal{I}}$, let $\max(S)$ be the element $e_{d,j} \in S$ with $\preceq$ -maximal index (if any). The restriction of $\preceq$ to $\mathbb{N}_{\geq}^2 = \{(x, y) \in \mathbb{N}^2 \mid x \geq y\}$ admits a successor relation, namely:

$$\text{succ}_{\preceq}(d, j) = \begin{cases} (d, j+1) & \text{if } j < d \\ (d+1, 0) & \text{if } j = d \end{cases}.$$

With a slight abuse of notation, $\text{succ}_{\preceq}(e_{d,j})$ will denote $e_{\text{succ}_{\preceq}(d,j)}$.

Next we need a constructive characterization of the fixpoint in assertion (7). To improve readability, in the following we

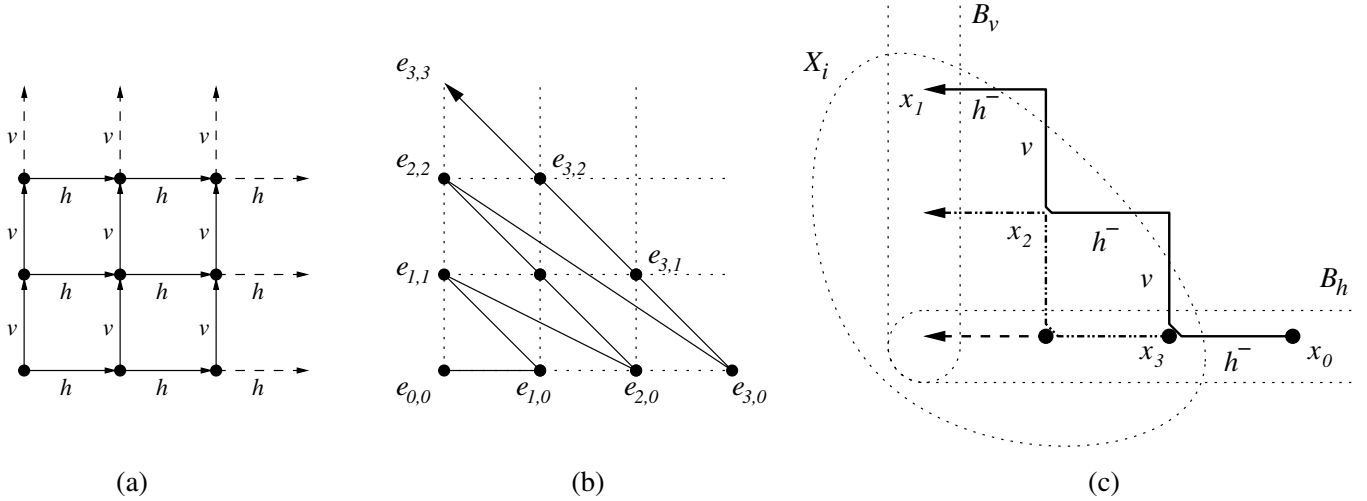


Figure 1: Modeling and visiting the grid $\mathcal{G}$

abbreviate $\xi(X)_{\rho[X/S]}^{\mathcal{I}}$ to $\xi^{\mathcal{I}}(S)$, where $S \subseteq \Delta^{\mathcal{I}}$. Formulae φ_i ($i = 0, 1, 2$) will be treated in a similar way. Then the fixpoint can be characterized as follows.

$$\begin{aligned} \mu X. \xi(X) &= \bigcup_{i < \omega} X_i, \text{ where} \\ X_0 &= \xi^{\mathcal{I}}(\emptyset) \\ X_i &= \xi^{\mathcal{I}}(X_{i-1}) \quad (i > 0) \end{aligned} \quad (8)$$

To ensure that the fixpoint is actually reached within ω steps, as implicitly claimed by the above construction, it suffices to show that $\xi_i^{\mathcal{I}}$ is continuous.

Proposition 3.2 $\xi_i^{\mathcal{I}}$ is continuous, that is, for all non-decreasing sequences $\{X_i\}_{i \leq \alpha}$, where α is an ordinal, $\xi^{\mathcal{I}}(\bigcup_{i < \alpha} X_i) = \bigcup_{i < \alpha} \xi^{\mathcal{I}}(X_i)$.

The following technical result formalizes the intuitive explanation of the nested fixpoint construction.

Proposition 3.3 For all integers $i \geq 0$ such that $X_i \subset X_{i+1}$, there exists $e_{d,j} \in X_{i+1}$ such that:

- a. $X_{i+1} = X_i \cup \{e_{d,j}\}$.
- b. $e_{d,j} = \text{succ}_{\leq}(\max(X_i)) = \max(X_{i+1})$.
- c. If $j > 0$ then $v^{-\mathcal{I}}(e_{d,j}) = e_{d-1,j-1}$.
- d. If $d > j$ then $h^{-\mathcal{I}}(e_{d,j}) = e_{d-1,j}$.
- e. For all $x \in X_{i+1} \setminus \{e_{d,j}\}$, if $h^{-\mathcal{I}}(x)$ is defined then $v^{\mathcal{I}}(h^{-\mathcal{I}}(x)) \in X_{i+1}$ and $h^{-\mathcal{I}}(v^{\mathcal{I}}(x)) \in X_{i+1}$.
- f. For all $e_{k,l} \in X_{i+1}$, if $e_{k,l} \in B_v^{\mathcal{I}}$ then $k = l$.

Proof. We prove a–f simultaneously, by induction on i . For $i = 0$, $X_i = O^{\mathcal{I}} = \{o\} = \{e_{0,0}\}$ and $X_{i+1} = \varphi_1^{\mathcal{I}}(X_0) = X_i \cup \{e_{1,0}\} \subset B_h^{\mathcal{I}}$. Then a–f are trivially satisfied with $(d, j) = (1, 0)$.

Now assume $i > 0$. Let $x_0 \in X_{i+1} \setminus X_i$. Then there are two possibilities: either $x_0 \in \varphi_0^{\mathcal{I}}(X_i)$ or $x_0 \in \varphi_1^{\mathcal{I}}(X_i)$.

Note that these two cases are mutually exclusive, because the elements of $\varphi_0^{\mathcal{I}}(X_i)$ are in the range of $v^{\mathcal{I}}$, while the elements of $\varphi_1^{\mathcal{I}}(X_i)$ are not, because they belong to $B_h^{\mathcal{I}}$.

Case 1: $x_0 \in \varphi_0^{\mathcal{I}}(X_i)$. By definition of φ_0 , there must be $x_1, x_2 \in X_i$ such that $h^{-\mathcal{I}}(v^{-\mathcal{I}}(x_0)) = x_1$ and $v^{-\mathcal{I}}(h^{\mathcal{I}}(x_0)) = x_2$. Since $h^{-\mathcal{I}}$ and $v^{-\mathcal{I}}$ are injective functions (by the assertions (2)), $v^{\mathcal{I}}(h^{-\mathcal{I}}(x_1)) = h^{-\mathcal{I}}(v^{\mathcal{I}}(x_2)) = x_0 \notin X_i$, and hence, by induction hypothesis *e* (I.H.e for short), $x_1 = x_2 = \max(X_i)$. Now, by choosing d and j such that $\max(X_i) = e_{d,j-1}$, we have $x_0 = h^{-\mathcal{I}}(v^{\mathcal{I}}(x_2)) = h^{-\mathcal{I}}(v^{\mathcal{I}}(e_{d,j-1})) = e_{d,j}$. The above equations can be rewritten as

$$\begin{aligned} v^{\mathcal{I}}(h^{-\mathcal{I}}(\max(X_i))) &= h^{-\mathcal{I}}(v^{\mathcal{I}}(\max(X_i))) = \\ &= x_0 = e_{d,j}. \end{aligned} \quad (9)$$

Since (9) holds for arbitrary $x_0 \in X_{i+1} \setminus X_i$, we have that x_0 is the unique member of $X_{i+1} \setminus X_i$, and hence point *a* holds.

Now, since $\max(X_i) = e_{d,j-1}$ and $\text{succ}_{\leq}(d, j-1) = (d, j)$, we immediately get point *b*.

To prove *c*, note that

$$\begin{aligned} v^{-\mathcal{I}}(e_{d,j}) &\stackrel{(9)}{=} v^{-\mathcal{I}}(v^{\mathcal{I}}(h^{-\mathcal{I}}(\max(X_i)))) = \\ &= h^{-\mathcal{I}}(e_{d,j-1}) \stackrel{\text{I.H.d}}{=} e_{d-1,j-1}. \end{aligned} \quad (10)$$

To prove *d*, note that

$$\begin{aligned} h^{-\mathcal{I}}(e_{d,j}) &= h^{-\mathcal{I}}(v^{\mathcal{I}}(v^{-\mathcal{I}}(e_{d,j}))) \stackrel{(10)}{=} \\ &= h^{-\mathcal{I}}(v^{\mathcal{I}}(e_{d-1,j-1})) \stackrel{\text{by def.}}{=} e_{d-1,j}. \end{aligned}$$

Point *e* follows from the corresponding I.H.e and (9).

We are only left to show that *f* holds. The induction hypothesis I.H.f covers all cases but $(k, l) = (d, l)$. If $e_{d,l} \in B_v^{\mathcal{I}}$, then $v^{-\mathcal{I}}(e_{d,j}) \in B_v^{\mathcal{I}}$, by assertion (5). Moreover, $v^{-\mathcal{I}}(e_{d,j}) = e_{d-1,j-1}$, by (10). Then $e_{d-1,j-1} \in B_v^{\mathcal{I}}$, and hence, by I.H.f, $d-1 = j-1$. Point *f* follows immediately.

This concludes the proof for Case 1.

Case 2: $x_0 \in \varphi_1^{\mathcal{I}}(X_i)$. By definition of φ_1 , the following facts hold: (i) $x_0 \in B_h^{\mathcal{I}}$, (ii) there exist $\{x_1, \dots, x_n\} \in X_i$ ($n > 0$) such that

$$x_1 \in B_v^{\mathcal{I}} \quad (11)$$

$$x_n = h^{-\mathcal{I}}(x_0) \quad (12)$$

$$x_k = h^{-\mathcal{I}}(v^{\mathcal{I}}(x_{k+1})) \quad (1 \leq k \leq n-1). \quad (13)$$

(cf. Figure 1(c)). By I.H.f and (11), $x_1 = e_{d-1,d-1}$ for some $d \in \mathbb{N}$. Then, by (13), $\{x_1, \dots, x_n\}$ must equal $\{e_{d-1,d-1}, e_{d-1,d-2}, \dots, e_{d-1,0}\}$.

Now, if $e_{d-1,d-1} \neq \max(X_i)$, then the set $\{e_{d-1,d-1}, e_{d-1,d-2}, \dots, e_{d-1,0}\}$ would be entirely contained in X_{i-1} and hence $x_0 \in \varphi_1^{\mathcal{I}}(X_{i-1}) \subseteq X_i$, a contradiction. Consequently,

$$e_{d-1,d-1} = \max(X_i). \quad (14)$$

Note also that

$$x_0 \stackrel{(12)}{=} h^{\mathcal{I}}(x_n) = h^{\mathcal{I}}(e_{d-1,0}) = e_{d,0}. \quad (15)$$

Now we are ready to prove a–f. Let $j = 0$.

Since $x_0 = e_{d,0}$ for an arbitrary $x_0 \in X_{i+1} \setminus X_i$, point a holds.

Moreover, $\text{succ}_{\leq}(d-1, d-1) = (d, 0)$ so point b is entailed by (14).

Point c holds vacuously as $j = 0$.

Point d holds, as $h^{-\mathcal{I}}(e_{d,0}) = e_{d-1,0}$ by definition of $e_{d,0}$.

Point e is covered by I.H.e with the exception of $\max(X_i) = e_{d-1,d-1}$, that satisfies e vacuously because it is a member of $B_v^{\mathcal{I}}$ and hence $h^{-\mathcal{I}}(e_{d-1,d-1})$ is undefined.

Finally, to prove point f, note that $e_{d,0} \notin B_v^{\mathcal{I}}$ (because $e_{d,0} \in B_h^{\mathcal{I}} \setminus O^{\mathcal{I}}$), so f is satisfied vacuously when $(k, l) = (d, 0)$. All the other cases are implied by I.H.f. This completes the proof. ■

We are now ready to prove the two propositions that confirm that $\mathcal{T}$ characterizes the grid.

Proposition 3.4 *Every model of $\mathcal{T}$ is isomorphic to $\mathcal{G}^*$.*

Proof. Suppose $\mathcal{I}$ is a model of $\mathcal{T}$. Then (5) and (6) cause $\Delta^{\mathcal{I}}$ to be infinite. On the contrary, each X_i ($i < \omega$) is finite, by Proposition 3.3.a. It follows by assertion (7) that no X_i can be a fixpoint of $\xi^{\mathcal{I}}$ for $i < \omega$, and hence the hypothesis of Proposition 3.3—that is, $X_i \subset X_{i+1}$ —is satisfied for all $i < \omega$.

As a first consequence of Propositions 3.3.a, b, we have that for all $(d, j) \in \mathbb{N}_{\geq}^2$, $e_{d,j}$ is defined and $\Delta^{\mathcal{I}} = \{e_{d,j} \mid (d, j) \in \mathbb{N}_{\geq}^2\}$. Since $X_i \subset X_{i+1}$, Propositions 3.3.a, b imply also that $e_{d,j}$ and $e_{k,l}$ are always distinct when $(d, j) \neq (k, l)$.

Then the function $f : \Delta^{\mathcal{I}} \rightarrow \mathbb{N}^2$ specified by

$$f(e_{d,j}) = (d - j, j)$$

is well defined and total. Moreover, the total function $g : \mathbb{N}^2 \rightarrow \Delta^{\mathcal{I}}$ defined by $g(i, j) = e_{i+j,j}$ satisfies $g \circ f = f \circ g = \text{id}$, so f is a bijection.

We are left to show that f is a morphism. By Proposition 3.3.c, d, we have that for all $(d, j) \in \mathbb{N}_{\geq}^2$, $h^{\mathcal{I}}(e_{d,j}) = e_{d+1,j}$ and $v^{\mathcal{I}}(e_{d,j}) = e_{d+1,j+1}$. Then

$$h^{\mathcal{G}}(f(e_{d,j})) = (d - j + 1, j) = f(h^{\mathcal{I}}(e_{d,j})), \text{ and}$$

$$v^{\mathcal{G}}(f(e_{d,j})) = (d - j, j + 1) = f(v^{\mathcal{I}}(e_{d,j})).$$

Moreover, $f(O^{\mathcal{I}}) = \{f(e_{0,0})\} = \{(0, 0)\} = O^{\mathcal{G}^*}$. We conclude that f is an isomorphism. ■

Proposition 3.5 *$\mathcal{G}^*$ is a model of $\mathcal{T}$.*

Since the $\mu\text{ALC}\mathcal{I}\mathcal{O}_f$ TBox $\mathcal{T}$ is a projective characterization of $\mathcal{G}$ (by Propositions 3.4 and 3.5) and the $\mu\text{ALC}\mathcal{I}\mathcal{O}_f$ assertion (1) is satisfied only by correct tilings, we derive the following lemma.

Lemma 3.6 *Satisfiability of $\mu\text{ALC}\mathcal{I}\mathcal{O}_f$ TBoxes is undecidable.*

We are left to extend this lemma to concept satisfiability and entailment. This is done through the following reductions.

Lemma 3.7 *In all extensions of $\mu\text{ALC}\mathcal{I}$,*

- TBox satisfiability can be reduced to concept satisfiability in polynomial time.*
- Concept unsatisfiability can be reduced to entailment in polynomial time.*

By Lemma 3.6 and Lemma 3.7, we conclude that the main result of this section, Theorem 3.1, holds.

Finally, with Theorem 3.1 and the standard embedding of description logics into propositional dynamic logics, we immediately obtain the following result.

Corollary 3.8 *Formula satisfiability in the hybrid μ -calculus with deterministic programs is undecidable.*

4 Regular roles and counting

Description logics with regular role expressions introduce a form of recursion (Kleene's star, or reflexive transitive closure) different from fixpoints. It is interesting to investigate the interplay of this form of recursion and counting.

Consider $\text{ALC}\mathcal{I}\mathcal{Q}^{\sqcup,*}$ (the extension of $\text{ALC}\mathcal{I}\mathcal{Q}$ with role union and reflexive transitive closure). We prove its undecidability by characterizing domino problems with unbounded grids (i.e., grids without borders) in $\text{ALC}\mathcal{I}\mathcal{Q}^{\sqcup,*}$. The grid is modelled by splitting the injective functional role h (resp. v) into the disjoint union of two roles h_0 and h_1 (resp. v_0 and v_1), alternated as shown in Figure 2(a). Nodes are then partitioned into four classes $\eta_{i,j}$, $0 \leq i, j \leq 1$, according to their incoming and outgoing edges ($(\exists R)$ abbreviates $\exists R.\top$):

$$\begin{aligned} \eta_{i,j} \triangleq & (\exists h_i) \cap (\exists v_j) \cap (\exists h_{1-i}^-) \cap (\exists v_{1-j}^-) \cap \\ & \neg(\exists h_{1-i}) \cap \neg(\exists v_{1-j}) \cap \neg(\exists h_i^-) \cap \neg(\exists v_j^-). \end{aligned}$$

All nodes are forced to belong to one of these four classes by an assertion $\top \sqsubseteq \eta_{0,0} \sqcup \eta_{0,1} \sqcup \eta_{1,0} \sqcup \eta_{1,1}$.

Furthermore, the four assertions

$$\eta_{i,j} \sqsubseteq (\forall h_i.\eta_{1-i,j}) \cap (\forall v_j.\eta_{i,1-j}) \quad (16)$$

(where $0 \leq i, j \leq 1$) force the desired alternation of h_0 , h_1 , v_0 and v_1 (cf. Figure 2(a)).

Now the compound role $(h_i \sqcup v_j)^*$ applied to a node c_0 can reach at most five different nodes c_0 – c_4 (cf. Figure 2(b)), because the alternation of edges with index 0 and 1 (more precisely, assertions (16)) guarantees that c_1 , c_3 , c_4 have no

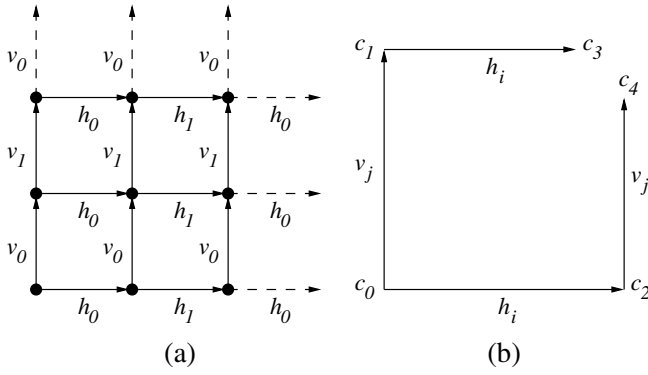


Figure 2: Modeling an even grid $\mathcal{G}$

outcoming v_j edges, and that c_2, c_3, c_4 have no outcoming h_i edges.

Then the following assertions (where $0 \leq i, j \leq 1$) imply that for each node in $\eta_{i,j}$, h_i and v_j commute.

$$\eta_{i,j} \sqsubseteq \exists^{\leq 4}(h_i \sqcup v_j)^*. \quad (17)$$

In fact—with reference to Figure 2(b)—for all c_0 in $\eta_{i,j}$, assertion (16) forces nodes c_0 – c_4 to belong to different (mutually disjoint) concepts $\eta_{x,y}$, with the exception of c_3 and c_4 , both of which belong to $\eta_{1-i,1-j}$. Now if $c_3 \neq c_4$ then (17) would be violated (there would be 5 nodes reachable with $(h_i \sqcup v_j)^*$), so h_i and v_j commute (i.e., h_0, h_1, v_0, v_1 characterize the unbounded grid).

We are only left to model correct tilings. It suffices to adapt assertion (1) by replacing h and v with $h_0 \sqcup h_1$ and $v_0 \sqcup v_1$, respectively. From the above discussion we derive the following theorem.

Theorem 4.1 $\mathcal{ALCTQ}^{\sqcup,*}$ is undecidable.

On the contrary, $\mu\mathcal{ALCTQ}^{\sqcup}$ is decidable. In fact, every expression in $\mu\mathcal{ALCTQ}^{\sqcup}$ is equivalent to an expression in the decidable logic $\mu\mathcal{ALCTQ}$ thanks to the equivalence

$$\exists^{\leq n}(R \sqcup R').C \equiv \bigsqcup_{0 \leq k \leq n} (\exists^{\leq k} R.C \sqcap \exists^{\leq n-k} R'.C).$$

Then Theorem 4.1 suggests that transitive closure is more powerful than fixpoints in this context (the extension of $\mathcal{ALCTQ}^{\sqcup}$ with fixpoints is decidable, while the extension with $*$ is not).

5 Discussion and conclusions

Description logics evolved into a hierarchy of decidable logics with multiple maximal elements. Some support fixpoints, inverse roles, and either nominals or number restrictions (but not both) [Sattler and Vardi, 1999; Kupferman *et al.*, 2002]. Others support rich sets of role operators, including union and transitive closure.

The results of this paper show that the above features cannot be easily combined into one decidable logic. In particular, no decidable extension of $\mathcal{ALCT}$ can simultaneously support fixpoints, nominals and number restrictions, even in the very special case where number restrictions are confined into the

functionality assertions (2).¹ As a corollary, the hybrid μ -calculus with deterministic programs is proved to be undecidable. Moreover, Theorem 4.1 shows that role union and transitive closure cannot occur together within number restrictions (it provides also evidence that $*$ is more expressive than fixpoints).

These results have immediate implications on $\mathcal{DLR}$ [Calvanese *et al.*, 1999], a rich DL with n -ary relations. Recall that $\mu\mathcal{ALCTQ}$ can be embedded into $\mathcal{DLR}_\mu$ [Calvanese *et al.*, 1999]. Similarly, $\mathcal{ALCTQ}$ can be embedded into $\mathcal{DLR}$ (the fragment of $\mathcal{DLR}_\mu$ without fixpoints). Then Theorem 3.1 and Theorem 4.1 imply that decidability is preserved neither by extending $\mathcal{DLR}_\mu$ with nominals,² nor by extending $\mathcal{DLR}$ with role operators $\sqcup$ and $*$.

An interesting question arising from these results concerns the family of *service description logics* $\mathcal{SDL}(\mathcal{X})$ [Bonatti, 2002]. These logics are analogous to $\mathcal{DLR}_\mu$, in the sense that they model mappings (that can be regarded as n -ary relations). $\mathcal{SDL}(\mathcal{X})$ differs from $\mathcal{DLR}_\mu$ because the former features set abstraction and composition, while $\mathcal{DLR}_\mu$ supports number restrictions. Service descriptions in $\mathcal{SDL}(\mathcal{X})$ are supposed to extend an underlying ontology written in a standard description logic $\mathcal{X}$ (modelling concepts and roles only). The main reasoning tasks for $\mathcal{SDL}(\mathcal{X})$ are proved to be decidable by embedding $\mathcal{SDL}(\mathcal{X})$ into decidable extensions of both $\mu\mathcal{ALCTO}$ and $\mathcal{X}$. By the undecidability of $\mu\mathcal{ALCTO}_f$, it follows that this technique cannot be applied when $\mathcal{X}$ supports number restrictions, or simply functional roles. Then the (un)decidability of $\mathcal{SDL}(\mathcal{X})$, when $\mathcal{X}$ supports number restrictions of some sort, remains an interesting open issue.

References

- [Bonatti, 2002] P.A. Bonatti. Towards Service Description Logics. In *Proc. of JELIA'02*, LNAI 2424, pages 74–85, 2002. Springer.
- [Calvanese *et al.*, 1999] D. Calvanese, G. De Giacomo, M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of IJCAI'99*, pages 84–89, 1999. Int'l Joint Committee on Artificial Int.
- [De Giacomo, 1995] G. De Giacomo. *Decidability of class-based knowledge representation formalisms*. PhD Thesis, Università di Roma “La Sapienza”, 1995.
- [Grädel *et al.*, 1999] E. Grädel, M. Otto, E. Rosen. Undecidability results on two-variable logics. *Archive for Mathematical Logic*, 38:213–354, 1999.
- [Kupferman *et al.*, 2002] O. Kupferman, U. Sattler, M.Y. Vardi. The Complexity of the Graded μ -Calculus. In *Proc. of CADE 2002*, pages 423–437, 2002. Springer.
- [Sattler and Vardi, 1999] U. Sattler, M.Y. Vardi. The hybrid μ -calculus. In *Proc. of IJCAR'01*, LNCS 2392, pages 27–30, 2001. Springer.

¹We conjecture this result still holds when functionality assertions are restricted to atomic roles only.

²This application of our results constitutes an alternative proof of a known result (De Giacomo, personal communication).

Abductive Matchmaking using Description Logics

Tommaso Di Noia⁽¹⁾, Eugenio Di Sciascio⁽¹⁾, Francesco M. Donini⁽²⁾, Marina Mongiello⁽¹⁾

⁽¹⁾Politecnico di Bari, Via Re David, 200, 70125 Bari, Italy

⁽²⁾Università della Tuscia, Via S. Carlo 32, 01100 Viterbo, Italy
{t.dinoia, disciascio, mongiello}@poliba.it; donini@unitus.it

Abstract

Motivated by the matchmaking problem in electronic marketplaces, we study abduction in Description Logics. We devise suitable definitions of the problem, and show how they can model commonsense reasoning usually employed in analyzing classified announcements having a standardized terminology. We then describe a system partially implementing these ideas, and present a simple experiment, which shows the correspondence between the system behavior with human users judgement.

1 Motivation

We describe several scenarios in which matchmaking [Di Noia *et al.*, 2003; Sycara *et al.*, 2002; Di Sciascio *et al.*, 2001; Trastour *et al.*, 2002] is necessary, starting with a human situation and moving to more automated ones.

Human matchmaking. Imagine you are looking for an apartment in London, UK. On a local newspaper, you published a classified ad, which we call D (for Demand), asking the following:

Apt. required: Soho, 2 rooms, smoker, dog, garden, max 600/month. Phone xxx-yyyyy

Since no one phoned you, you open a copy of the same newspaper, and start filtering a long list of classified ads offering apartments to rent. You might encounter the following ad, that we call S_1 (for Supply):

Apt. centre (Piccadilly), car place, fireplace, 2 rooms. Phone zzz-wwwww

Then you wonder if this offer fits your needs. The match between Soho and Piccadilly depends on some domain knowledge — actually, they are consistent, although Piccadilly does not imply Soho. Then S_1 does not mention that dogs and smokers are allowed, but this does not necessarily mean that they are not (open-world assumption). It does not mention a garden too, but there is a car place, which could be turned into a small garden if it is open air and private. Should you phone to inquire S_1 first, or phone first the following S_2 :

AAA Apt. in London. Phone sss-ttttt

Here it is not just a problem of hypothetical reasoning, it is also a problem of giving some total order to offers that are compatible with yours, presumably trying to maximize the probability of finding a good offer in few phone calls — but wait: if these offers would match, why *they* didn't phone *you*?

Software agents matchmaking. A similar scenario, but for the fact that agents may actively look for a matching counteroffer, instead of simply posting their offer. So they may actually call you...

Matchmaking in Electronic marketplaces. The same scenario, but with some sort of standardized language to describe supplies and demands, and maybe some centralized facilitator for matchmaking, *e.g.*, containing the necessary domain knowledge.

Service Discovery. Either for software agents or in e-marketplaces, the offers may describe a general service in some standardized language, such as the simple 128-bit string of Bluetooth, or the more structured DAML-S.

The third and fourth scenarios are more amenable to automated matchmaking, since they imply a standardized language and ontology. However, observe that also in the first and second scenarios, when supplies and demands are expressed in heterogeneous forms, matchmaking is not a mere schema/data integration problem. In fact, integration techniques [Madhavan *et al.*, 2001] may be employed to make heterogeneous supplies and demands comparable; but once they are reformulated in a comparable way, one is still left with true matchmaking problems: *i)* given a proposal, are there compatible counteroffers? *ii)* if there are many compatible counteroffers, which are the most promising ones (and why)?

2 Principles for matchmaking

We now discuss in detail various technical options for performing matchmaking, first trying to highlight limitations of non-logical approaches, and then discussing the general Knowledge Representation principles that a logical approach may yield.

First of all, we note that if supplies and demands are simple names or strings, the only possible match would be identity, resulting in an all-or-nothing approach to matchmaking. This is the approach taken by Bluetooth, whose Service Discovery Protocol tries to match Universally Unique Identifiers of

services and requests. Although effective for fixed technical domains, such an approach misses the fact that proposals usually have some sort of structure in them. Such a structure could be exploited in order to evaluate “interesting” inexact matches.

Vector-based techniques taken by classical Information Retrieval can be used, too, thus reverting matchmaking to similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker [Kuokka and Harada, 1996] or in LARKS [Sycara *et al.*, 2002]. Obviously lack of document structure in descriptions would make matching only probabilistic and strange situations may ensue, *e.g.*, the demand “Apt. with two Rooms in Soho pets allowed no smokers” would match the supply “Apt. with two Rooms in Soho, no pets, smokers allowed”.

A further possibility is to use sets of words to describe the structure of supplies and demands. We consider these description as a particular case of logical description, and move on to Description Logics (DLs). Hence, from now on we suppose that supplies and demands are expressed in a DL $\mathcal{L}$, equipped with a model-theoretic semantics. We suppose also that a common ontology for supplies and demands is established, as a TBox $\mathcal{T}$ in $\mathcal{L}$. Now a match between a supply S and a demand D could be evaluated according to $\mathcal{T}$. This framework ensures the first property that we would like to hold for matchmaking, namely, an open-world assumption.

Property 1 (Open-world descriptions) *The absence of a characteristic in the description of a proposal should not be interpreted as a constraint of absence. Instead, it should be considered as a characteristic that could be either refined later, or left open if it is irrelevant for the issuer of the proposal.*

Moreover, if all constraints of a demand D were fulfilled by a supply S , but not vice versa, then S should be among the top ranked supplies in the list of potential partners of the demander, while D should not appear at the top in the list of potential partners of the supplier.

Property 2 (Non-symmetric evaluation) *A matchmaking system may give different evaluations to the match between a supply S and a demand D , depending on whether it is trying to match S with D , or D with S — i.e., depending on who is going to use this evaluation.*

There are three relations between concepts expressing supplies and demands, that we consider meaningful in matchmaking: implication, consistency, and inconsistency.

In the first case, $\mathcal{T} \models (D \sqsubseteq S)$, i.e., constraints imposed by D imply those of S , and vice versa if $\mathcal{T} \models (S \sqsubseteq D)$. This relation extends the previous set-based inclusion to general concepts. If both $\mathcal{T} \models (D \sqsubseteq S)$ and $\mathcal{T} \models (S \sqsubseteq D)$, then D and S should be considered equivalent in $\mathcal{T}$. This relation extends exact matching, making syntax differences irrelevant. In case of consistency, $D \sqcap S$ is satisfiable in $\mathcal{T}$. Then, there is a *potential* match, in the sense that the constraints of neither proposal exclude the other. This relation has been highlighted also by other researchers [Trastour *et al.*, 2002]. However, that proposal lacks a *ranking* between different potential matches. In the third case, $D \sqcap S$ is unsatisfiable in $\mathcal{T}$. Although also in this case a matching could be

defined [Di Noia *et al.*, 2003], we do not treat this case here for lack of space. Hence, from now on we concentrate on potential match only. We now highlight some principles that — we believe — every ranking function should have in logical matchmaking.

First of all, if a logic is used to give some meaning to descriptions of supplies and demands, then proposals with the same meaning should have the same ranking, independently of their syntactic descriptions.

Definition 1 (Syntax independence in ranking) *A ranking of matches is syntax independent if for every pair of supplies S_1 and S_2 , demand D , and ontology $\mathcal{T}$, when S_1 is logically equivalent to S_2 then S_1 and S_2 have the same ranking for D , and the same holds also for every pair of logically equivalent demands D_1, D_2 with respect to every supply S .*

For example, an apartment S_1 , described as available for the summer quarter, should have the same rank — with respect to a request — as another S_2 , identical but for the fact that it is described to be available for june-july-august. Clearly, when the logic admits a normal form of expressions — as CNF or DNF for propositional logic, or the normal form of concepts for the DL of CLASSIC [Borgida *et al.*, 1989] — using such a normal form ensures by itself syntax independence.

We now consider the relation between ranking and implications. We go back to the descriptions with sets of words, since they are easy to read through. Let D be a demand and S_1, S_2 be two supplies defined as follows:

$$\begin{aligned} D &= \{\text{apartment, soho, twoRooms, petsAllowed}\} \\ S_1 &= \{\text{apartment, soho, boiler, quiet}\} \\ S_2 &= \{\text{apartment, soho, boiler, quiet, lastFloor}\} \end{aligned}$$

In this case, the characteristics that S_2 adds to S_1 are irrelevant for D . Hence, whatever the rank for S_1 , the one for S_2 should be the same. If instead we let

$$S_3 = \{\text{apartment, soho, boiler, quiet, petsAllowed}\}$$

then S_3 should be ranked better than S_1 since it adds a characteristic required by D . We generalize this example to concepts, and state the following definition.

Definition 2 (Monotonicity of ranking over subsumption) *A ranking of potential matches is monotonic over subsumption whenever for every demand D , for every pair of supplies S_1 and S_2 , and ontology $\mathcal{T}$, if S_1 and S_2 are both potential matches for D , and $\mathcal{T} \models (S_2 \sqsubseteq S_1)$, then S_2 should be ranked either the same, or better than S_1 , and the same should hold for every pair of demands D_1, D_2 with respect to a supply S .*

Intuitively, the above definition could be read of as *A ranking is monotonic over subsumption if: the more specific, the better*. Observe that we use the word “better” instead of using any symbol $\leq, \geq$. This is because some rankings may assume that “better” means “increasing” (towards either infinity, or 1) while others may assume “decreasing” (towards 0). Observe also that the above definition refers only to potential matches, i.e., consistent matches — otherwise, $\perp$ would be the best

match (being the most specific concept), which is obviously not the case.

We remark that the properties and definitions we stated in this section are independent of the particular DL employed, or even the particular *logic* chosen. For instance, the same properties could be stated if propositional logic was used to describe supplies, demands and the ontology. In this respect, we believe that this section keeps its significance also if one chooses more expressive DLs such as $\mathcal{SHOQ}(\mathcal{D})$ [Horrocks and Sattler, 2001] or DAML [Paolucci *et al.*, 2002].

3 Abduction in Description Logics

We tend to follow the notation of [Eiter and Gottlob, 1995] for propositional abduction whenever possible, and adapt it to our setting.

Definition 3 Let $\mathcal{L}$ be a DL, C, D , be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. A Concept Abduction Problem (CAP), denoted as $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$, is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \not\models C \sqcap H \equiv \perp$, and $\mathcal{T} \models C \sqcap H \sqsubseteq D$.

We use $\mathcal{P}$ as a symbol for a CAP, and we denote with $SOL(\mathcal{P})$ the set of all solutions to a CAP $\mathcal{P}$. Observe that if $C \sqcap D$ is unsatisfiable in $\mathcal{T}$, then trivially $SOL(\mathcal{P}) = \emptyset$, i.e., there are no solution to $\mathcal{P}$ if C and D are not a potential match. Hence from now on we concentrate on the case $\mathcal{T} \models C \sqcap D \equiv \perp$.

As propositional abduction extends implication, a CAP extends concept subsumption. But differently from propositional abduction, we do not make any distinction between manifestations and hypotheses, which is usual when abduction is used for diagnosis. However, diagnosis is not our domain of application for abduction, and when making hypotheses about properties of goods in e-marketplaces, there is no point in making such a distinction. This uniformity implies that if $C \sqcap D$ is satisfiable in $\mathcal{T}$, then $D \in SOL(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$, i.e., there is always the trivial solution D to a non-trivial CAP $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$. Interpreted in our e-marketplace application domain, it means that if I hypothesize for the counteroffer exactly all my specifications, the counteroffer trivially meets my specifications. Since one wants to model Occam's razor, some minimality in the hypotheses must be defined. In the following definition, we denote with $\sqsubseteq_{\mathcal{T}}$ the subsumption relation between concepts w.r.t. a TBox $\mathcal{T}$.

Definition 4 Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP. The set $SOL_{\sqsubseteq}(\mathcal{P})$ is the subset of $SOL(\mathcal{P})$ whose concepts are maximal under $\sqsubseteq_{\mathcal{T}}$. The set $SOL_{\leq}(\mathcal{P})$ is the subset of $SOL(\mathcal{P})$ whose concepts have minimum length.

We note that being maximal under $\sqsubseteq_{\mathcal{T}}$ is still a minimality criterion, since it means that no unnecessary hypothesis is assumed. It can be proved that the two measures are incomparable.

Proposition 1 There exists a CAP $\mathcal{P}$ such that the two sets $SOL_{\sqsubseteq}(\mathcal{P})$ and $SOL_{\leq}(\mathcal{P})$ are incomparable.

Proof. It is sufficient to consider $D = A_1 \sqcap A_2 \sqcap A_3$, $C = A_1$, and $\mathcal{T} = \{B \sqsubseteq A_2 \sqcap A_3\}$. The logic is even propositional. Then $A_2 \sqcap A_3 \in SOL_{\sqsubseteq}(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$,

$B \in SOL_{\leq}(\langle \mathcal{L}, C, D, \mathcal{T} \rangle)$, and neither solution is in the other set. $\square$

The proof highlights that, although $\leq$ -minimality could be preferable for conciseness, it is heavily dependent on $\mathcal{T}$. For every concept $H \in SOL(\mathcal{P})$, it is sufficient to add the axiom $A = H$ to get a $\leq$ -minimal solution A .

A third minimality criterion is possible for DLs that admit a normal form as a conjunction of concepts, that is, every concept C in $\mathcal{L}$ can be rewritten as an equivalent concept $C_1 \sqcap \dots \sqcap C_n$. This is the case for $\mathcal{L} = \mathcal{ALN}$, and for the DL of the CLASSIC KR system. We call such a normal form CNF, in analogy with propositional logic.

Definition 5 Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP in which $\mathcal{L}$ admits a CNF. The set $SOL_{\sqcap}(\mathcal{P})$ is the subset of $SOL(\mathcal{P})$ whose concepts are minimal conjunctions, i.e., if $C \in SOL_{\sqcap}(\mathcal{P})$ then no sub-conjunction of C is in $SOL(\mathcal{P})$. We call such concepts irreducible solutions of $\mathcal{P}$.

It turns out that $\sqcap$ -minimality subsumes both $\sqsubseteq_{\mathcal{T}}$ -minimality and $\leq$ -minimality. This is not a surprise, since $\sqcap$ -minimality is a form of $\sqsubseteq_{\emptyset}$ -minimality, i.e., maximality for subsumption w.r.t. an empty TBox.

Proposition 2 For every CAP $\mathcal{P}$ in which $\mathcal{L}$ admits a CNF, both $SOL_{\sqsubseteq}(\mathcal{P})$ and $SOL_{\leq}(\mathcal{P})$ are included in $SOL_{\sqcap}(\mathcal{P})$.

Proof. If a concept C is not $\sqcap$ -minimal, then it is not $\leq$ -minimal, and the same for $\sqsubseteq_{\mathcal{T}}$. $\square$

3.1 Computational Complexity

Since Concept Abduction extends Concept Subsumption w.r.t. a TBox, complexity lower bounds of the latter problem carry over to decision problems related to a CAP.

Proposition 3 Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP. If Concept Subsumption w.r.t. a TBox in $\mathcal{L}$ is a problem $\mathcal{C}$ -hard for a complexity class $\mathcal{C}$, then deciding whether a concept belongs to $SOL(\mathcal{P})$ is both $\mathcal{C}$ -hard and co- $\mathcal{C}$ -hard.

Proof. Hardness for $\mathcal{C}$ comes from the fact that C is subsumed by D in $\mathcal{T}$ iff $\top \in SOL(\mathcal{P})$. $\square$

Hence, if $\mathcal{L}$ contains the DL $\mathcal{AL}$, then deciding whether a concept belongs to $SOL(\mathcal{P})$ is EXPTIME-hard [Donini, 2003] for a general TBox $\mathcal{T}$, but it is PSPACE-hard if $\mathcal{T}$ contains only acyclic concept axioms [Calvanese, 1996].

Regarding upper bounds, a simple result can be derived from the fact that D is always a solution of the CAP $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$ — although not always a minimal one. First of all, a total length-lexicographic order $\prec$ can be defined over concepts as follows: given two concepts $C, D \in \mathcal{L}$, let $C \prec D$ if either $|C| < |D|$, or both $|C| = |D|$ and C is lexicographically before D . Based on this total order, an algorithm for finding a $\leq$ -minimal solution of a CAP, using polynomial space relatively to an oracle for subsumption in $\mathcal{L}$, is the following:

input: a CAP $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$, with $\mathcal{T} \models C \sqcap D \equiv \perp$
output: a concept in $SOL_{\leq}(\mathcal{P})$
variables: concept $x \in \mathcal{L}$
begin
 $x := \top$;

```

while  $|x| < |D|$  do
  if  $(\mathcal{T} \models C \sqcap x \sqsubseteq D \text{ and } \mathcal{T} \not\models C \sqcap x \sqsubseteq \perp)$ 
  then return  $x$ ;
   $x :=$  next concept following  $x$  in  $\prec$ 
endwhile;
return  $D$ 
end.

```

The above algorithm uses polynomial space (considering the two calls to subsumption as an oracle) since it just tries all concepts with less symbols than D , and returns D if it does not find a shorter solution. Therefore, it provides an upper bound on the complexity of CAP, depending on the complexity class to which subsumption in $\mathcal{L}$ belongs to. Although this result does not directly lead to a practical algorithm, it puts an upper bound on the complexity of the problem, hence on the complexity of every optimal algorithm.

Theorem 1 *Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP. If subsumption in $\mathcal{L}$ belongs to a complexity class $\mathcal{C}$ that is included in PSPACE, then finding a concept in $SOL_{\leq}(\mathcal{P})$ is a problem in PSPACE. Otherwise if PSPACE is included in $\mathcal{C}$, then finding a concept in $SOL_{\leq}(\mathcal{P})$ is a problem in $\mathcal{C}$.*

Given that the problem of finding a solution cannot be simpler than the corresponding decision problem, we can conclude with some general results about $\leq$ -minimal abduction.

Theorem 2 *Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP, with $\mathcal{L}$ a DL whose expressiveness is between $\mathcal{AL}$ and the DL containing concept constructors $\sqcap, \sqcup, \neg, \exists R.C, \forall R.C$ and role constructors $\sqcup$, role chain, transitive-reflexive roles, role identity, role inverse. Then finding a concept in $SOL_{\leq}(\mathcal{P})$ is a problem EXPTIME-complete when $\mathcal{T}$ is a general TBox.*

Proof. Hardness results for $\mathcal{AL}$ are in [Donini, 2003]. Membership result for the most expressive logic comes from converse-Propositional Dynamic Logic [Vardi and Wolper, 1986]. $\square$

Hence, for a general TBox the best known algorithms require exponential time and also exponential space (unless one proves $PSPACE = EXPTIME$).

When the TBox is acyclic, complexity results for subsumption imply that finding a concept in $SOL_{\leq}(\mathcal{P})$ is a problem PSPACE-complete for DLs whose expressiveness is between $\mathcal{ALE}$ [Calvanese, 1996] and $\mathcal{ALC}$ [Baader and Hollunder, 1991]. Even for the simplest logic $\mathcal{AL}$, the problem is co-NP-hard [Calvanese, 1996].

3.2 Irreducible solutions in $\mathcal{ALN}$

In this section, we assume that $\mathcal{T}$ of a CAP $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ is always acyclic. Finding an irreducible solution is easier than finding a $\leq$ -minimal or a $\sqsubseteq$ -minimal solution, since a greedy approach can be used to minimize the set of conjuncts in the solution: starting from $C \sqcap D$, delete one redundant conjunct at a time from D . However, instead of starting from $C \sqcap D$, we adapt a structural subsumption algorithm [Borgida and Patel-Schneider, 1994] that collects all concepts that should be conjoined to C to be subsumed by D . The algorithm operates on concepts in normal form; roughly speaking, this form is obtained by expanding concept names with

their definition in $\mathcal{T}$, and then applying the following rewriting rules as much as possible:

$$\begin{aligned}
 \forall R.C \sqcap \forall R.D &\rightarrow \forall R.C \sqcap D \\
 C \sqcap \perp &\rightarrow \perp \\
 (\geq n R) \sqcap (\leq m R) &\rightarrow \perp \text{ if } n > m \\
 \forall R.\perp &\rightarrow \forall R.\perp \sqcap (\leq 0 R)
 \end{aligned}$$

In the following algorithm, we denote the fact that a concept A appears as a conjunct of a concept C with $A \in C$.

Algorithm *findIrred*($\mathcal{P}$);
input: a CAP $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$, with $\mathcal{L} = \mathcal{ALN}$, acyclic $\mathcal{T}$, and $\mathcal{T} \not\models C \sqcap D \equiv \perp$
output: concept $H \in SOL_{\sqcap}(\mathcal{P})$
 (where $H = \top$ means that C is already subsumed by D)
variables: concept H
begin
 $H := \top$;
 for every concept name y in D
 if y is not in C
 then $H := H \sqcap y$;
 for every concept $(\geq n R)$ in D
 such that there is no concept $(\geq m R)$ in C with $m \geq n$
 $H := H \sqcap (\geq n R)$;
 for every concept $(\leq n R)$ in D
 such that there is no concept $(\leq m R)$ in C with $m \leq n$
 $H := H \sqcap (\leq n R)$;
 for every concept $\forall R.E$ in D
 if there exists $\forall R.F$ in C
 then $H := H \sqcap \text{findIrred}(\langle \mathcal{L}, F, E, \mathcal{T} \rangle)$;
 else $H := H \sqcap \text{findIrred}(\langle \mathcal{L}, \top, E, \mathcal{T} \rangle)$;
 /* now $H \in SOL(\mathcal{P})$, but it might be reducible */
 for every concept H_i in H
 if H without $H_i \in SOL(\mathcal{P})$
 then delete H_i from H ; **return** H ;
end.

It can be proved that the concept H returned by *findIrred*() is indeed an irreducible solution of $\mathcal{P}$. As for complexity, the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up, as demonstrated by Nebel in [1990]. And anyway, a polynomial algorithm cannot be expected since subsumption in $\mathcal{AL}$ with an acyclic $\mathcal{T}$ is co-NP-hard [Calvanese, 1996]. However, in the cited paper Nebel argues that the expansion is exponential in the depth of the hierarchy $\mathcal{T}$; if the depth of $\mathcal{T}$ is $O(\log |\mathcal{T}|)$, then the expansion is polynomial, and so is the above algorithm.

Theorem 3 *Let $\mathcal{P} = \langle \mathcal{L}, C, D, \mathcal{T} \rangle$ be a CAP, with $\mathcal{L} = \mathcal{AL}$, and $\mathcal{T}$ an acyclic TBox whose depth is always bounded by $O(\log |\mathcal{T}|)$. Then finding an irreducible solution to $\mathcal{P}$ is a problem solvable in polynomial time.*

In order to rank the proposals in a marketplace according to how “near” they are to a given proposal D , we take the number of concept names in the irreducible solution returned by the above algorithm. Although this makes the rank depend on the algorithm, it can be easily proved that this definition is in accordance with the principles highlighted in Section 2. Moreover, the fact that a concept H is actually computed

makes it easy to devise an explanation facility, in case a user wants to know why a given proposal has been ranked before another. This transparency is crucial for our electronic business scenario, where we must give the user reasons to trust the system. In the next section, we show the results of an experiment comparing the rank obtained by our deployed system and the rank provided by some users in the same scenario.

4 Experiments

Using the highlighted properties as a formal specification we designed and implemented a prototype facilitator. The system embeds an adapted NeoClassic reasoner, the C++ implementation of the original Classic. The rationale of the choice of the Classic system, apart from the obviously useful availability of concrete datatypes and the possibility to extend its functionalities through test functions, is its polynomial time inference. The system receives a description of demand/supply. Then, the reasoner checks the description for consistency; if it fails, based on the reasoner output, the system provides an error message stating the error occurred. Otherwise the proper matchmaking process takes place, and the system returns a ranked set of matches. When no exact match is available, The user may investigate potential matches. In this process he/she is helped by the system ranking. We believe that degree of conformance of an automated matchmaking facilitator to users' perception is of extreme importance, especially in a setting as our own, which ranks and categorizes matches. To evaluate the ability of the system to meet users' expectations we set up a little experimental framework.

We selected all apartments-rental advertisements from the dedicated section of a local newspaper on Oct. 6th of last year. We subdivided them in two sets, *demands* (23 advertisements) and *supplies* (39 advertisements). It is noteworthy that, at least for the single-day pick we made, we were not able to detect any exact match. We submitted to twenty volunteers of various sex and age, a questionnaire that included 8 items. Each item was one demand and a set of up to eight supplies, or one supply and a set of up to eight demands. Volunteers were asked to rank, according to their judgement, elements of each set with respect to the given advertisement, with the following question: Order the following Demands(Supply) with respect to the given Supply (Demands) as you would contact them had you issued the given Supply (Demand). Volunteers were given unlimited time and in average it took approximately half hour to complete the questionnaire. Then the same sets of items were submitted to the reasoner. As a general consideration, the response of the reasoner was quite close to the users' ones, and considering average volunteers orderings the system ranking was in agreement with the human judgement almost always. Average percentage of deviation from the mean values determined by the users on the whole experiment was 8.3.

As an example, we show here in more detail results of the evaluation for a single demand/supplies matching process, extracted from the experimental setting. Let us consider the example demand: *student looking for a nice 1/2 bed flat, ch, furnished, kitchen, washing machine. Price: 150*

Also consider the following advertisements submitted as

```
demand (and Bedroom (all toLetFor Student) (at-least 1 hasBed)
(at-most 2 hasBed) (at-least 1 hasFacilities) (all hasFacilities
(and WashingMachine NoAutonomousHeating FullyFurnished)) (all
hasServices Kitchen) (all price (maximum 150)))

supply1 (and Bedroom (all price (minimum 120)) (all priceIncludes
(and Bill TVPrice)) (all hasFacilities (and FullyFurnished
Spatious)))

supply2 (and (all price (minimum 150))(at-least 2 hasRoom)(at-most
2 hasRoom)(all hasRoom Room)(at-least 2 toLetFor)(at-most 2
toLetFor)(all toLetFor (and Couple Student))(at-least 1
depositRequired)(all depositRequired Yes))

supply3 (and (at-least 1 depositRequired)(all depositRequired
Yes)(all price (minimum 80)) Bedroom (all toLetFor (and Student
NoSmoker Worker (all sex Female))))

supply4 (and SingleRoom (all toLetFor (and NoSmoker
Student))(at-least 2 occupants)(all price (minimum 120)))

supply5 (and DoubleRoom (all toLetFor Single)(all hasFacilities
(and Lounge Garden))(all hasServices Kitchen)(all price (minimum
85))(all priceIncludes CouncilTax))

supply6 (and Bedroom (at-least 1 occupants) (all price (minimum
81)) (all toLetFor Female)(all hasFacilities (and WashingMachine
TV VCR)))

supply7 (and Bedroom (all price (minimum 95)) (all occupants
Family) (at-least 2 occupants) (all toLetFor (and Student
Professional (all sex Female))))

supply8 (and Flat (at-least 2 hasBed) (at-most 2 hasBed) (all
toLetFor Student) (all hasFacilities ADSL) (all price (minimum
95)))
```

Figure 1: Neoclassic descriptions of demand and supplies

supplies:

- *Supply1: large, fully furnished room, price includes cable TV and bills. Price: 120*
- *Supply2: double room, suit a couple or two girls, required deposit. Price: 150*
- *Supply3: room to rent, suit a non smoking female student with worker, international preferred, deposit required. Price: 80*
- *Supply4: single room in clean flat for nosmoker quiet student, sharing with 2 others. Price: 120*
- *Supply5: dbl room in shared house, suit single person, use of lounge, kitchen, garden, rent includes council tax. Price: 85*
- *Supply6: female to share a room in a residential area flatshare, washing machine, TV, VCR. Price: 81*
- *Supply7: large room in family houseshare with 2 adults, suit prof / student female, viewing recommended. Price: 95*
- *Supply8: 2 bed flat, perfect for student, dble bed ADSL computer beneath. Price: 600*

The translation of these advertisements into Classic, in accordance with our ontology is pictured in Figure 1. Table 1 shows results provided from the matchmaking algorithm comparing the system provided ranking and the average volunteers orderings for demands w.r.t. to a supply. Figure 2 shows the same results in a graphical form. Notice that, while volunteers gave strictly ranked orderings, the system could also provide equal ranking for various matches. Polygons represent a range for the system total order of supplies, when there are equally ranked supplies. For example, Supply 3 and 7 are given the same rank by the system, so they could be equally put in fourth or fifth position when supplies are given a total order. Hence we consider correct system

matched pairs	system ranking	averaged users ranking
demand,supply1	2-3	2
demand,supply2	8	6
demand,supply3	4-5	4
demand,supply4	2-3	3
demand,supply5	1	1
demand,supply6	6	5
demand,supply7	4-5	8
demand,supply8	7	7

Table 1: System results and user preferences

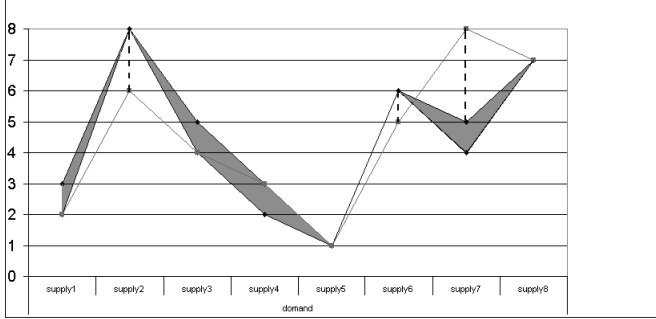


Figure 2: System and averaged users ranking

ranks for Supply 3 and 4, because users' rank fall inside the polygons, while we consider incorrect system placement of Supply 2 and 7 (dotted lines mark the difference).

5 Conclusion

Motivated by the matchmaking problem in electronic marketplaces, we have studied abduction in DLs. We have presented suitable definitions of the problem, and have shown how they can model commonsense reasoning usually employed in analyzing typical commercial advertisements. Although we used the simple DL $\mathcal{ALN}$, our definitions and logical framework can be used also when a more expressive DL is used. Of course in this case more complex algorithms should be devised. To substantiate these ideas we have implemented a prototype based on the devised specifications adapting a NeoClassic reasoner, and presented experiments that show the correspondence between the system behavior with human users judgement.

Acknowledgements

We thank Peter Patel-Schneider for valuable suggestions on CLASSIC, Simona Colucci and Marco Mottola for useful implementations. This work has been supported by MURST project CLUSTER22, by EU-POP project "Negotiation Agents for the Electronic Marketplace", by PON project "Tecnologie innovative per la valorizzazione e la fruizione dei Beni Culturali", by CNR projects LAICO, and "Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio".

References

- [Baader and Hollunder, 1991] F. Baader and B. Hollunder. *KRIS: Knowledge Representation and Inference System*. *SIGART Bulletin*, 2(3):8–14, 1991.
- [Borgida and Patel-Schneider, 1994] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *J. of Artif. Intell. Res.*, 1:277–308, 1994.
- [Borgida et al., 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. Alperin Resnick. CLASSIC: A structural data model for objects. In *ACM SIGMOD*, pages 59–67, 1989.
- [Calvanese, 1996] D. Calvanese. Reasoning with inclusion axioms in description logics. In *ECAI'96*, pages 303–307. John Wiley & Sons, 1996.
- [Di Noia et al., 2003] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In *SAC'03*, pages 582–586. ACM, 2003.
- [Di Sciascio et al., 2001] E. Di Sciascio, F.M. Donini, M. Mongiello, and G. Piscitelli. A knowledge-based system for person-to-person e-commerce. In *ADL Workshop (KI 2001)*, 2001.
- [Donini, 2003] F. M. Donini. Complexity of reasoning. In *Description Logics Handbook*, chapter 3. Cambridge University Press, 2003.
- [Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *JACM*, 42(1):3–42, 1995.
- [Horrocks and Sattler, 2001] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In *IJCAI 2001*, pages 199–204, 2001.
- [Kuokka and Harada, 1996] D. Kuokka and L. Harada. Integrating information via matchmaking. *J. of Intelligent Information Systems*, 6:261–279, 1996.
- [Madhavan et al., 2001] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proc. of VLDB '01*, pages 49–58, 2001.
- [Nebel, 1990] B. Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43:235–249, 1990.
- [Paolucci et al., 2002] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *ISWC 2002*, pages 333–347, 2002.
- [Sycara et al., 2002] K. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.
- [Trastour et al., 2002] D. Trastour, C. Bartolini, and C. Priest. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW'02*, pages 89–98. ACM, 2002.
- [Vardi and Wolper, 1986] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Comp. and Syst. Sci.*, 32:183–221, 1986.

Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms

Ian Horrocks

Department of Computer Science,
University of Manchester, UK
horrocks@cs.man.ac.uk

Ulrike Sattler

Institut für Theoretische Informatik,
TU Dresden, Germany
sattler@tcs.inf.tu-dresden.de

Abstract

Motivated by medical terminology applications, we investigate the decidability of the well known expressive DL, $\mathcal{SHIQ}$, extended with role inclusion axioms (RIAs) of the form $R \circ S \sqsubseteq P$. We show that this extension is undecidable even when RIAs are restricted to the forms $R \circ S \sqsubseteq R$ or $S \circ R \sqsubseteq R$, but that decidability can be regained by further restricting RIAs to be acyclic. We present a tableau algorithm for this DL and report on its implementation, which behaves well in practise and provides important additional functionality in a medical terminology application.

1 Motivation

The description logic (DL) $\mathcal{SHIQ}$ [Horrocks *et al.*, 1999; Horrocks and Sattler, 2002b] is an expressive knowledge representation formalism that extends $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] (a notational variant of the multi modal logic $\mathbf{K}$ [Schild, 1991]) with qualifying number restrictions, inverse roles, role inclusion axioms (RIAs) $R \sqsubseteq S$, and transitive roles. The development of $\mathcal{SHIQ}$ was motivated by several applications, one of which was the representation of knowledge about complex physically structured domains found, e.g., in chemical engineering [Sattler, 2000] and medical terminology [Rector and Horrocks, 1997].

Although $\mathcal{SHIQ}$ allows many important properties of such domains to be captured (e.g., transitive and inverse roles), one extremely useful feature that it cannot express is the “propagation” of one property along another property [Padgham and Lambrix, 1994; Rector, 2002; Spackman, 2000]. E.g., it may be useful to express the fact that certain locative properties are transferred across certain partonomic properties so that a trauma or lesion located in a part of a body structure is recognised as being located in the body structure as a whole. This enables highly desirable inferences such as a fracture of the neck of the femur being inferred to be a kind of fracture of the femur, or an ulcer located in the gastric mucosa being inferred to be a kind of stomach ulcer.

The importance of these kinds of inference, particularly in medical terminology applications, is illustrated by the fact that the Grail DL [Rector *et al.*, 1997], which was specifically

designed for use with medical terminology, is able to represent these kinds of propagation (although it is quite weak in other respects). Moreover, in another medical terminology application using the comparatively inexpressive DL $\mathcal{ALC}$, a rather complex “work around” is performed in order to represent similar propagations [Schulz and Hahn, 2001].¹ Similar expressiveness was also provided in the CycL language by the `transfersThro` statement [Lenat and Guha, 1989].

It is quite straightforward to extend $\mathcal{SHIQ}$ so that this kind of propagation can be expressed: simply allow for role inclusion axioms of the form $R \circ S \sqsubseteq P$, which then enforces all models $\mathcal{I}$ to interpret the composition of $R^{\mathcal{I}}$ with $S^{\mathcal{I}}$ as a sub-relation of $P^{\mathcal{I}}$. E.g., the above examples translate into

`hasLocation ◦ isDivisionOf ⊆ hasLocation,`

which implies that

`Fracture ⊆ ∃hasLocation.(Neck ⊆ isDivisionOf.Femur)`

is subsumed by/a specialization of

`Fracture ⊆ ∃hasLocation.Femur`

Unfortunately, this extension leads to the undecidability of the interesting inference problems; see [Wessel, 2001] for an undecidability proof and [Baldoni, 1998; Baldoni *et al.*, 1998; Demri, 2001] for the closely related family of *Grammar Logics*. On closer inspection of the problem, we observe that only RIAs of the form $R \circ S \sqsubseteq S$ or $S \circ R \sqsubseteq S$ are required in order to express propagation. Surprisingly, it turns out that $\mathcal{SHIQ}$ extended with this restricted form of RIAs is still undecidable. Decidability can be regained, however, by further restricting the set of RIAs to be *acyclic* (in a non-standard way). This additional restriction does not seem too severe: the above examples are still covered, acyclic sets of RIAs should suffice for many applications, and cycles in RIAs may even be an indicator of modelling flaws [Rector, 2002]. We call this decidable logic $\mathcal{RIQ}$.

Here, we present the above undecidability result and prove the decidability of $\mathcal{SHIQ}$ with acyclic RIAs via a tableau-based decision procedure for concept satisfiability. The algorithm works by transforming concepts of the form $\forall R.C$, where R is a role, into concepts of the form $\forall \mathcal{A}.C$, where $\mathcal{A}$ is a non-deterministic finite automaton (NFA). These automata

¹In this approach, so-called *SEP-triplets* are used both to compensate for the absence of transitive roles in $\mathcal{ALC}$, and to express the propagation of properties across a distinguished “part-of” role.

are derived from a set of RIAs $\mathcal{R}$ by first *unfolding* $\mathcal{R}$ into a set of implications $\text{exp}(\mathcal{R})$ between regular expressions and roles, and then transforming the regular expressions into automata. The algorithm is of the same complexity as the one for $\mathcal{SHIQ}$ —in the size of $\text{exp}(\mathcal{R})$ and the length of the input concept—but, unfortunately, $\text{exp}(\mathcal{R})$ is exponential in $\mathcal{R}$. We present a syntactic restriction that avoids this blow-up; investigating whether this blow-up can be avoided in general will be part of future work. Finally, in order to evaluate the practicability of this algorithm, we have extended the DL system FaCT [Horrocks, 1998] to deal with acyclic RIAs. We discuss how the properties of NFAs are exploited in the implementation, and we present some preliminary results showing that the performance of the extended system is comparable with that of the original, and that it is able to compute inferences of the kind mentioned above w.r.t. the well known Galen medical terminology knowledge base [Rector and Horrocks, 1997; Horrocks, 1998].

For full proofs, the interested reader is referred to [Horrocks and Sattler, 2002a].

2 Preliminaries

In this section, we introduce the DL $\mathcal{SH}^+\mathcal{IQ}$. This includes the definition of syntax, semantics, and inference problems.

Definition 1 Let $\mathbf{C}$ and $\mathbf{R}$ be sets of concept and role names. The set of roles is $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. For roles R_i (each of which can be inverse), a role inclusion axiom (RIA) is an expression of the form $R_1 \sqsubseteq R_2$, $R_1 \circ R_2 \sqsubseteq R_1$, or $R_1 \circ R_2 \sqsubseteq R_2$. A generalised role box (g-RBox) is a set of RIAs.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ associates, with each role name R , a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Inverse roles are interpreted as usual, i.e.,

$$(R^-)^{\mathcal{I}} = \{\langle y, x \rangle \mid \langle x, y \rangle \in R^{\mathcal{I}}\} \text{ for each role } R \in \mathbf{R}.$$

An interpretation $\mathcal{I}$ is a model of a g-RBox $\mathcal{R}$ if it satisfies each inclusion assertion in $\mathcal{R}$, i.e., if

$$\begin{aligned} R_1^{\mathcal{I}} &\subseteq R_2^{\mathcal{I}} && \text{for each } R_1 \sqsubseteq R_2 \in \mathcal{R} \text{ and} \\ R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} &\subseteq R_3^{\mathcal{I}} && \text{for each } R_1 \circ R_2 \sqsubseteq R_3 \in \mathcal{R}, \end{aligned}$$

where $\circ$ stands for the composition of binary relations.

Transitive role names were not introduced since $R \circ R \sqsubseteq R$ is equivalent to saying that R is a transitive role.

To avoid considering roles such as R^{--} , we define a function Inv on roles as follows: $\text{Inv}(R) = R^-$ if R is a role name, and $\text{Inv}(R) = S$ if $R = S^-$.

Obviously, if $S \circ R \sqsubseteq S \in \mathcal{R}$ ($R \circ S \sqsubseteq S \in \mathcal{R}$ or $R \sqsubseteq S \in \mathcal{R}$), then each model of $\mathcal{R}$ also satisfies $\text{Inv}(R) \circ \text{Inv}(S) \sqsubseteq \text{Inv}(S)$ ($\text{Inv}(S) \circ \text{Inv}(R) \sqsubseteq \text{Inv}(S)$ and $\text{Inv}(R) \sqsubseteq \text{Inv}(S)$). Thus, in the following, we assume that a g-RBox always contains both “directions” of a RIA.

For a g-RBox $\mathcal{R}$, we define the relation $\sqsubseteq^*$ to be the transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R}$.

Definition 2 A role S is simple if it does not have implied sub-roles, i.e., if $S_1 \circ S_2 \sqsubseteq S_3$ implies $S_3 \sqsubseteq^* S$ does not hold.

The set of $\mathcal{SH}^+\mathcal{IQ}$ -concepts is the smallest set such that (i) every concept name is a concept, and, (ii) if C , D are concepts, R is a role (possibly inverse), S is a simple role (possibly inverse), and n is a nonnegative integer, then $C \sqcap D$,

$C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $(\geq n.S.C)$, and $(\leq n.S.C)$ are also concepts.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$, called the domain of $\mathcal{I}$, and a valuation $\cdot^{\mathcal{I}}$ which maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for all concepts C , D , roles R , S , and non-negative integers n , the following equations are satisfied, where $\#M$ denotes the cardinality of a set M :

$$\begin{aligned} (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\ (\exists R.C)^{\mathcal{I}} &= \{x \mid \exists y. \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}, \\ (\forall R.C)^{\mathcal{I}} &= \{x \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}, \\ (\geq nR.C)^{\mathcal{I}} &= \{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}, \\ (\leq nR.C)^{\mathcal{I}} &= \{x \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\}. \end{aligned}$$

A concept C is called satisfiable w.r.t. a g-RBox $\mathcal{R}$ iff there is a model $\mathcal{I}$ of $\mathcal{R}$ with $C^{\mathcal{I}} \neq \emptyset$. A concept D subsumes a concept C w.r.t. $\mathcal{R}$ (written $C \sqsubseteq_{\mathcal{R}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{R}$. For an interpretation $\mathcal{I}$, an element $x \in \Delta^{\mathcal{I}}$ is called an instance of a concept C iff $x \in C^{\mathcal{I}}$.

Remarks: number restrictions $(\geq nR.C)$ and $(\leq nR.C)$ are restricted to *simple* roles (intuitively these are (possibly inverse) roles that are not implied by others) since $\mathcal{SHIQ}$ without this restriction is undecidable [Horrocks *et al.*, 1999].

For DLs that are closed under negation, subsumption and (un)satisfiability can be mutually reduced: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable, and C is unsatisfiable iff $C \sqsubseteq A \sqcap \neg A$ for some concept name A . It is straightforward to extend these reductions to g-RBoxes and TBoxes. In contrast, the reduction of inference problems w.r.t. a TBox to pure concept inference problems (possibly w.r.t. a g-RBox), deserves special care: $\mathcal{SHIQ}$ is expressive enough to *internalise* TBoxes, i.e., to reduce reasoning w.r.t. TBoxes to reasoning without TBoxes [Schild, 1991; Horrocks *et al.*, 1999]. Thus, in the following, we restrict our attention to the satisfiability of $\mathcal{SH}^+\mathcal{IQ}$ -concepts.

2.1 Relationship with other formalisms

Grammar logics are a class of propositional multi modal logics where the accessibility relations are “axiomatised” through a grammar [Farin s del Cerro and Penttonen, 1988]. More precisely, for σ_i, τ_j modal parameters, the production rule $\sigma_1 \dots \sigma_m \rightarrow \tau_1 \dots \tau_n$ can be viewed as a notational variant for the RIA $\tau_1 \circ \dots \circ \tau_n \sqsubseteq \sigma_1 \circ \dots \circ \sigma_m$. Analogously to the DL case, the semantics of a grammar logic takes into account only those frames/relational structures that “satisfy the grammar”.

Now grammars are traditionally organised in (refinements of) the Chomsky hierarchy, which induces a hierarchy of grammar logics, e.g., *context free* grammar logics are those propositional multi modal logics where the accessibility relations can be axiomatised through a *context free* grammar. Unsurprisingly, the expressiveness of the grammars influences the expressiveness of the corresponding grammar logics. It was shown that satisfiability of *regular* grammar logics is ExpTime-complete [Demri, 2001], whereas this problem is undecidable for context free grammar logics [Baldoni, 1998; Baldoni *et al.*, 1998]. The latter result is closely related to the undecidability proof in [Wessel, 2001].

Here, we are concerned with (a) multi modal logics that provide for a converse operator on modal parameters and *graded* modalities (to restrict the number of accessible worlds; see, e.g., [Tobies, 2001]) and (b) a certain sub-class of context-free grammars. In our undecidability proof in Section 3, the main difficulty was to develop a grammar that generates the language $\{(ab)^n(cd)^n \mid n \geq 0\}$ using only productions of the form $R \rightarrow RS$ or $R \rightarrow SR$.² We can construct a “similar” grammar G with $L(G) \cap (ab)^*(cd)^* = \{(ab)^n(cd)^n \mid n \geq 0\}$. The production rules of G are

$$\begin{aligned} D &\rightarrow AD, & A &\rightarrow AC, \\ C &\rightarrow BC, & B &\rightarrow BD, & A &\rightarrow a, \dots D \rightarrow d. \end{aligned}$$

Role value maps (RVMs) [Brachman and Schmolze, 1985; Schmidt-Schauss, 1989] are closely related to the RIAs investigated here. RVMs are concepts of the form $R_1 \dots R_m \sqsubseteq S_1 \dots S_n$, for R_i, S_i roles, whose interpretation $(R_1 \dots R_m \sqsubseteq S_1 \dots S_n)^{\mathcal{I}}$ is defined as follows:

$$\{x \in \Delta^{\mathcal{I}} \mid (R_1 \dots R_m)^{\mathcal{I}}(x) \subseteq (S_1 \dots S_n)^{\mathcal{I}}(x)\},$$

where $(R_1 \dots R_m)^{\mathcal{I}}(x)$ denotes the set of those $y \in \Delta^{\mathcal{I}}$ that are reachable from x via $R_1^{\mathcal{I}} \circ \dots \circ R_m^{\mathcal{I}}$. Thus the RIA $R \circ S \sqsubseteq T$ is equivalent to saying that each individual must be an instance of $RS \sqsubseteq T$. The undecidability proof of KL-ONE [Schmidt-Schauss, 1989] also involves RVMs $T \sqsubseteq RS$, and thus cannot be adapted easily to our logic.

3 $\mathcal{SH}^+ \mathcal{IQ}$ is undecidable

Due to the syntactic restriction on RIAs, we were not able to adapt the undecidability proof for $\mathcal{ALC}$ with context-free or linear grammars in [Baldoni, 1998; Baldoni *et al.*, 1998; Demri, 2001], the one for $\mathcal{ALC}$ with role boxes [Wessel, 2001], or the one for KL-ONE [Schmidt-Schauss, 1989] to prove undecidability of $\mathcal{SH}^+ \mathcal{IQ}$. In the following, we sketch the reduction of the undecidable domino problem [Berger, 1966] to $\mathcal{SH}^+ \mathcal{IQ}$ satisfiability.

Definition 3 A domino system $\mathcal{D} = (D, H, V)$ consists of a non-empty set of domino types $D = \{D_1, \dots, D_n\}$, and of sets of horizontally and vertically matching pairs $H \subseteq D \times D$ and $V \subseteq D \times D$. The problem is to determine if, for a given $\mathcal{D}$, there exists a tiling $t : \mathbb{N} \times \mathbb{N} \rightarrow D$ such that for all $m, n \in \mathbb{N}$, $\langle t(m, n), t(m+1, n) \rangle \in H$ and $\langle t(m, n), t(m, n+1) \rangle \in V$.

For a domino system $\mathcal{D}$, we define a $\mathcal{SH}^+ \mathcal{IQ}$ -concept $C_{\mathcal{D}}$ and a g-RBox $\mathcal{R}_{\mathcal{D}}$ such that $\mathcal{D}$ has a tiling iff $C_{\mathcal{D}}$ is satisfiable w.r.t. $\mathcal{R}_{\mathcal{D}}$. Due to space limitation, we only present $\mathcal{R}_{\mathcal{D}}$:

$$\begin{aligned} &\{v_i \sqsubseteq y_i, v_i \sqsubseteq v, h_i \sqsubseteq x_i, h_i \sqsubseteq h \mid 0 \leq i \leq 3\} \cup \\ &\{x_{i \oplus 1}^- y_i \sqsubseteq y_i, x_{i \oplus 1}^- x_i \sqsubseteq x_{i \oplus 1}^- \mid 0 \leq i \leq 3\} \\ &\{y_{i \oplus 1}^- x_i \sqsubseteq x_i, y_{i \oplus 1}^- y_i \sqsubseteq y_{i \oplus 1}^- \mid 0 \leq i \leq 3\} \end{aligned}$$

where $\oplus$ and $\ominus$ denotes addition and subtraction modulo four.

Existential and number restrictions on roles h and v (for the horizontal and vertical neighbours) are used to ensure that a point has at most one vertical and at most one horizontal

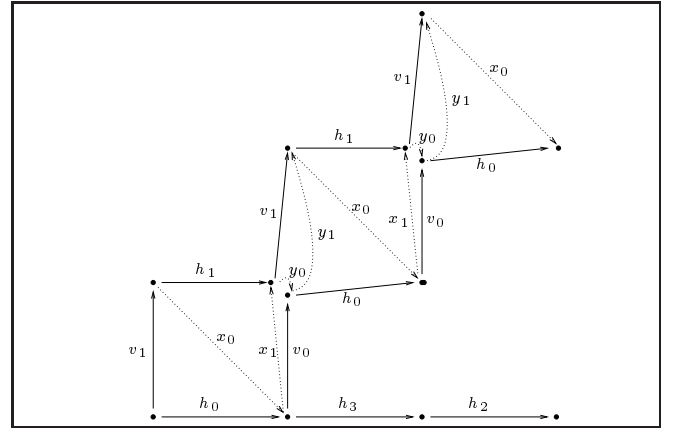


Figure 1: The staircase structure and the effects of $\mathcal{R}_{\mathcal{D}}$.

successor, and that these successors satisfy the horizontal and vertical matching conditions induced by H and V ; this, as well as ensuring that each point is associated with exactly one domino type, is standard in domino reductions.

The next step is rather special: we do not enforce a grid structure, but a structure with “staircases”, which is illustrated in Figure 1. To this purpose, we introduce four sub-roles $v_0, \dots, v_3$ of v and four sub-roles $h_0, \dots, h_3$ of h , and ensure that we only have “staircases”. An i -staircase is an alternating chain of v_i and h_i edges, without any other v_j - or h_j -successors. At each point on the x -axis, two staircases start that need not meet again, one i -staircase starting with v_i and one $i \ominus 1$ -staircase starting with $h_{i \ominus 1}$. A symmetric behaviour is enforced for the nodes on the y -axis.

It only remains to ensure that, if two elements b, b' represent the same point in the grid, then they are associated with the same domino type: b and b' “represent the same point” if there is an n and an instance a on the x - or the y -axis such that both b and b' are reachable by following a staircase starting at a for n steps, i.e., if there is a $v_i h_i$ -path (resp. $h_i v_i$ -path) of length n from a to b , and a $h_{i \oplus 1} v_{i \oplus 1}$ -path (resp. $v_{i \oplus 1} h_{i \oplus 1}$ -path) of length n from a to b' .

To this purpose, we add super roles x_i of h_i and y_i of v_i (for which we use dashed arrows in Figure 1), and the last group of RIAs in $\mathcal{R}_{\mathcal{D}}$. These role inclusion axioms enforce appropriate, additional role successorships between elements, and we use the additional roles x_i and y_i since we only want to have at most one v_i or h_i -successor. For each 2 staircases starting at the same element on one of the axes, these RIAs ensure that each pair of elements representing the same point is related by y_i . To see this, consider the consequences of the RIAs for elements representing the four points $(1, 0), \dots, (2, 1)$, and “apply” the RIA $y_1^- x_0 \sqsubseteq x_0$. Next, “apply” $x_1^- x_0 \sqsubseteq x_1^-$, and finally $x_1^- y_0 \sqsubseteq y_0$, which yields the y_0 -link between the two elements representing $(2, 1)$. Then, starting with $y_1^- y_0 \sqsubseteq y_1$, we can continue with the points $(2, 1), \dots, (3, 2)$ and work up the role inclusion axioms and up the staircase.

The above observations imply that the concept $C_{\mathcal{D}}$ is satisfiable w.r.t. $\mathcal{T}_{\mathcal{D}}$ and $\mathcal{R}_{\mathcal{D}}$ iff $\mathcal{D}$ has a solution.

²Thanks to Christof Löding at RWTH Aachen!

Theorem 1 *Satisfiability of $\mathcal{SH}^+\mathcal{IQ}$ -concepts w.r.t. generalised RBoxes is undecidable.*

4 $\mathcal{RIQ}$ is decidable

In this section, we show that $\mathcal{SHIQ}$ with *acyclic* generalised RBoxes, $\mathcal{RIQ}$, is decidable. We present a tableau-based algorithm that decides satisfiability of $\mathcal{RIQ}$ -concepts, and therefore also subsumption in $\mathcal{RIQ}$ and, using internalisation, both inferences w.r.t. TBoxes. The tableau algorithm implemented in the FaCT system [Horrocks, 1998] was extended to the one presented here, and the empirical results are reported in Section 5.

Definition 4 Let $\mathcal{R}$ be a *g-RBox* (containing always both directions of a RIA; see above). A role R directly affects a role S if $R \neq S$ and either $R \sqsubseteq S \in \mathcal{R}$, $R \circ S \sqsubseteq S \in \mathcal{R}$, or $S \circ R \sqsubseteq S \in \mathcal{R}$. Let “affects” be the transitive closure of “directly affects”. An acyclic generalised RBox (a-RBox) is a *g-RBox* where “affects” has no cycles. $\mathcal{RIQ}$ is the restriction of $\mathcal{SH}^+\mathcal{IQ}$ to a-RBoxes.

Please note that, in a-RBoxes, we can no longer say that a role R is symmetric using $R \sqsubseteq R^-$ and $R^- \sqsubseteq R$ since this would yield an “affects” cycle of length 2.

Syntactic transformations Before specifying this algorithm, we transform the RBox to make the presentation of the algorithm easier—basically, we *unfold* the role hierarchy to make all implications explicit.

Firstly, for each (possibly inverse) role R we define two regular expressions as follows:

$$\begin{aligned} \tau_R &:= \left(\bigcup_{\substack{S \circ R \sqsubseteq R \in \mathcal{R} \\ S \neq R}} S^* R \left(\bigcup_{\substack{R \circ T \sqsubseteq R \in \mathcal{R} \\ T \neq R}} T \right)^* \right) \\ \rho_R &:= \begin{cases} \tau_R & \text{if } R \circ R \sqsubseteq R \notin \mathcal{R} \\ (\tau_R)^+ & \text{if } R \circ R \sqsubseteq R \in \mathcal{R} \end{cases} \end{aligned}$$

Secondly, we iteratively replace roles in ρ_R with unions of regular expressions of roles, working our way up the affecting relation. We start with roles “almost” minimal w.r.t. the affecting relation, i.e., we start with roles R such that all roles S which affect R are *not* affected. We proceed with roles directly affected by roles that are either already treated or not affected by other roles, and do the following:

$$\begin{aligned} \rho_R &:= (\rho_R \text{ with } R \text{ replaced with } R \cup \bigcup_{\substack{P \sqsubseteq_1 R \\ P \neq R}} \rho_P) \text{ and,} \\ \text{for each } S \neq R \text{ occurring in } \rho_R & \\ \rho_R &:= (\rho_R \text{ with } S \text{ replaced with } \bigcup_{P \sqsubseteq_1 S} \rho_P). \end{aligned}$$

After this recursion, we define $\exp(\mathcal{R}) := \{\rho_R \sqsubseteq R \mid R \text{ occurs in } \mathcal{R}\}$.

Due to the acyclicity of $\mathcal{R}$, the recursion in this transformation terminates after at most n steps for n the number of role inclusion axioms in $\mathcal{R}$. Please note that, by construction, for each (possibly inverse) role R occurring in $\mathcal{R}$, $\exp(\mathcal{R})$ contains exactly one inclusion $\rho_R \sqsubseteq R$.

For example, for the RIAs $\mathcal{R}$

$$\begin{aligned} R \circ S &\sqsubseteq S, & S \circ W &\sqsubseteq S, & T_1 \circ R_1 &\sqsubseteq R_1, \\ R_2 \circ T_2 &\sqsubseteq R_2, & V \circ T_1 &\sqsubseteq T_1, & R_1 &\sqsubseteq R, & R_2 &\sqsubseteq R \end{aligned}$$

the above transformation yield a set $\exp(\mathcal{R})$ containing

$$\begin{aligned} (R \cup R_2 T_2^* \cup (V^* T_1)^* R_1)^* S W^* &\sqsubseteq S, \\ R \cup R_2 T_2^* \cup (V^* T_1)^* R_1 &\sqsubseteq R, \\ R_2 T_2^* &\sqsubseteq R_2, & (V^* T_1)^* R_1 &\sqsubseteq R_1, & V^* T_1 &\sqsubseteq T_1. \end{aligned}$$

Unfortunately, the size of $\exp(\mathcal{R})$ can be exponential in the size of $\mathcal{R}$. A further syntactic restriction which avoids this exponential blow-up is described in Section 4.1.

The regular role terms on the left hand side of $\exp(\mathcal{R})$ are read with the standard semantics for regular role expressions, (i.e., using union, composition, and transitive closure of binary relations, see, e.g., [Schild, 1991]). We use $L(\rho)$ to denote the language described by a regular expression ρ .

Lemma 1 *An interpretation $\mathcal{I}$ is a model of an acyclic generalised RBox $\mathcal{R}$ iff $\mathcal{I}$ is a model of $\exp(\mathcal{R})$.*

The Tableau Algorithm tries to construct, for an input $\mathcal{RIQ}$ -concept D and an a-RBox $\mathcal{R}$, a tableau (an abstraction of a model) for D w.r.t. $\mathcal{R}$. We can prove that this algorithm constructs a tableau for D and $\mathcal{R}$ iff D is satisfiable w.r.t. $\mathcal{R}$, and thus decides satisfiability of $\mathcal{RIQ}$ concepts w.r.t. an a-RBox. But for the use of NFAs introduced below, this algorithm is quite similar to the one for $\mathcal{SHIQ}$ [Horrocks *et al.*, 1999; Horrocks and Sattler, 2002b].

If R occurs in $\mathcal{R}$, then $\rho_R \sqsubseteq R \in \exp(\mathcal{R})$, and we can build a non-deterministic finite automaton (NFA) $\mathcal{A}^R$ with $L(\mathcal{A}^R) = L(\rho_R)$. Due to the use of non-deterministic automata, $\mathcal{A}^R$ can be of size linear in $|\rho_R|$. Otherwise, $\mathcal{A}^R$ is a (two-state) automaton with $L(\mathcal{A}^R) = \{R\}$.

For $\mathcal{A}$ an NFA and q a state in $\mathcal{A}$, $\mathcal{A}_q$ denotes the NFA obtained from $\mathcal{A}$ by making q the (only) initial state of $\mathcal{A}$, and we use $q \rightarrow_S q' \in \mathcal{A}$ to denote that $\mathcal{A}$ has a transition labelled with S from q to q' .

As usual, each concept can be easily transformed into an equivalent one in negation normal form (NNF, i.e., negation occurs in front of concept names only), and we use $\neg C$ for the NNF of a concept C . For a concept C , $\text{clos}(C)$ is the smallest set that contains C and that is closed under sub-concepts and $\neg$. Then $\text{fclos}(C, \mathcal{R})$ is the superset of $\text{clos}(C, \mathcal{R})$ that contains $\forall A_q^S.D$ for each S occurring in $\mathcal{R}$ or C with q a state in $\mathcal{A}^S$ and $\forall S.D \in \text{clos}(C)$.

A *completion tree* $\mathbf{T}$ for a $\mathcal{RIQ}$ concept D and an a-RBox $\mathcal{R}$ is a tree where each node x is labelled with a set $\mathcal{L}(x) \subseteq \text{fclos}(D, \mathcal{R})$ and each edge $\langle x, y \rangle$ from a node x to its successor y is labelled with a non-empty set $\mathcal{L}(\langle x, y \rangle) \subseteq \mathcal{R}$ of (possibly inverse) roles occurring in D and $\mathcal{R}$. Finally, completion trees come with an explicit inequality relation $\neq$ on nodes which is implicitly assumed to be symmetric.

If $R \in \mathcal{L}(\langle x, y \rangle)$ for a node x and its successor y and $R \sqsubseteq S$, then y is called an *S-successor* of x and x is called an *Inv(S)-predecessor* of y . If y is an *S-successor* or an *Inv(S)-predecessor* of x , then y is called an *S-neighbour* of x . Finally, *ancestor* is the transitive closure of *predecessor*.

For a role S , a concept C and a node x in $\mathbf{T}$ we define $S^{\mathbf{T}}(x, C) := \{y \mid y \text{ is an } S\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}$.

A node is *blocked* iff it is either directly or indirectly blocked. A node x is *directly blocked* iff none of its ancestors are blocked, and it has ancestors x' , y and y' such that (1) y is not the root node; (2) x is a successor of x' and y is a successor of y' ; and (3) $\mathcal{L}(x) = \mathcal{L}(y)$, $\mathcal{L}(x') = \mathcal{L}(y')$, $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle)$. A node y is *indirectly blocked* if one of its ancestors is blocked.

$R\sqcap$: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C_1, C_2\}$
$R\sqcup$: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{E\}$ for some $E \in \{C_1, C_2\}$
$R\exists$: if $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and x has no S -neighbour y with $C \in \mathcal{L}(y)$ then create a new node y with $\mathcal{L}(\langle x, y \rangle) := \{S\}$ and $\mathcal{L}(y) := \{C\}$
$R\forall_1$: if $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and $\forall A^S.C \notin \mathcal{L}(x)$ then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{\forall A^S.C\}$
$R\forall_2$: if $\forall A_p.C \in \mathcal{L}(x)$, x is not indirectly blocked, $p \rightarrow_S q$ in A_p , and y is an S -neighbour of x with $\forall A_q.C \notin \mathcal{L}(y)$ then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{\forall A_q.C\}$
$R\forall_3$: if $\forall A.C \in \mathcal{L}(x)$, x is not indirectly blocked, $\varepsilon \in L(A)$, and $C \notin \mathcal{L}(x)$ then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{C\}$
$R?$: if $(\leq nS.C) \in \mathcal{L}(x)$, x is not indir. blocked, and y is an S -neighbour of x with $\{C, \neg C\} \cap \mathcal{L}(y) = \emptyset$ then $\mathcal{L}(y) := \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \neg C\}$
$R\geq$: if $(\geq nS.C) \in \mathcal{L}(x)$, x is not blocked, and there are no $y_1, \dots, y_n \in S^T(x, C)$ with $y_i \neq y_j$ for each $i \neq j$ then create n new nodes y_i with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$, $\mathcal{L}(y_i) = \{C\}$, and $y_i \neq y_j$ for $1 \leq i < j \leq n$.
$R\leq$: if $(\leq nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked, $\#S^T(x, C) > n$, there are $y, z \in S^T(x, C)$ with $\text{not } y \neq z$ and y is not an ancestor of z , then $\mathcal{L}(z) := \mathcal{L}(z) \cup \mathcal{L}(y)$ and if z is an ancestor of x then $\mathcal{L}(\langle z, x \rangle) := \mathcal{L}(\langle z, x \rangle) \cup \text{Inv}(\mathcal{L}(\langle x, y \rangle))$ else $\mathcal{L}(\langle x, z \rangle) := \mathcal{L}(\langle x, z \rangle) \cup \mathcal{L}(\langle x, y \rangle)$ and remove y and the sub-tree below y

Figure 2: The Expansion Rules for $\mathcal{RIQ}$.

For a node x , $\mathcal{L}(x)$ is said to contain a *clash* if, for some concept name A , $\{A, \neg A\} \subseteq \mathcal{L}(x)$, or if there is some concept $(\leq nS.C) \in \mathcal{L}(x)$ and x has $n + 1$ S -neighbours $y_0, \dots, y_n$ with $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for all $0 \leq i < j \leq n$. A completion tree is *clash-free* if none of its nodes contains a clash, and it is *complete* if no rule from Figure 2 can be applied to it.

For a $\mathcal{RIQ}$ -concept D , the algorithm starts with the completion tree consisting of a single root node x with $\mathcal{L}(x) = \{D\}$ and $\neq \emptyset$. It applies the expansion rules in Figure 2, stopping when a clash occurs, and answers “ D is satisfiable w.r.t. $\mathcal{R}$ ” iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree, and “ D is unsatisfiable w.r.t. $\mathcal{R}$ ” otherwise.

Most of the rules have been used before for fragments of $\mathcal{RIQ}$ —only the three $\forall_i$ -rules are new: they are elegant generalisations of standard rules for value restrictions taking into account automata.

As usual, we can prove termination, soundness, and completeness of the tableau algorithm to show that it indeed de-

cides satisfiability of $\mathcal{RIQ}$ -concepts w.r.t. a-RBoxes.

Theorem 2 *The tableau algorithm decides satisfiability and subsumption of $\mathcal{RIQ}$ -concepts w.r.t. a-RBoxes and TBoxes.*

4.1 Avoiding the blow-up

So far, the satisfiability algorithm presented here involves an exponential blow-up compared to similar algorithms that are implemented in state-of-the-art description logic reasoners [Horrocks, 1998; Haarslev and Möller, 2001]: the closure $\text{fclos}(D, \mathcal{R})$ is exponential in $\mathcal{R}$ since we have “unfolded” the a-RBox $\mathcal{R}$ into the possibly exponentially large $\text{exp}(\mathcal{R})$. While investigating whether and how this exponential blow-up can be avoided, we observe that a further restriction of the syntax of a-RBoxes avoids this blow-up:

An a-RBox $\mathcal{R}$ is called *simple* if, whenever $R_1 \circ S \sqsubseteq S$ and $S \circ R_2 \sqsubseteq S$ are in $\mathcal{R}$, then R_1 and R_2 do not have a common subrole R' that occurs on the right hand side of an axiom $R' \circ S' \sqsubseteq R'$ or $S' \circ R' \sqsubseteq R'$.

For a *simple* a-RBox $\mathcal{R}$, $\text{exp}(\mathcal{R})$ is only polynomial in the size of $\mathcal{R}$ since each term used in the substitution step of the construction of $\text{exp}(\mathcal{R})$ from $\mathcal{R}$ is at most used once in each other axiom.

Thus, for simple role hierarchies, the tableau algorithm presented here is of the same worst case complexity as for $\mathcal{SHIQ}$, namely 2NExpTime. A detailed investigation of the exact complexity will be part of future work.

5 Empirical Evaluation

In order to evaluate the practicability of the above algorithm, we have extended the DL system FaCT [Horrocks, 1998] to deal with $\mathcal{RIQ}$, and we have carried out a preliminary empirical evaluation.

From a practical point of view, one potential problem with the $\mathcal{RIQ}$ algorithm is that the number of different automata, and hence the number of different $\forall A.C$ concepts, could be very large. Moreover, many of these automata could be equivalent (i.e., accept the same languages). This could adversely effect blocking, and thus lead to a serious degradation in performance [Horrocks and Sattler, 2002b].

The FaCT implementation addresses these possible problems by transforming all of the initial NFAs into minimal deterministic finite automata (DFAs) using the AT&T FSM LibraryTM [Mohri *et al.*, 1998]. One DFA is constructed for each role, the states in each automaton are uniquely numbered, and the implementation uses concepts of the form $\forall A.C$, where A is the number of a state in one of the automata. Because the automata are deterministic, for each concept of the form $\forall A.C$ in the label of a node x , the $R\forall_2$ -rule can add at most one concept to the label of a given neighbouring node y per role in the label of the edge $\langle x, y \rangle$. Moreover, because the automata are minimal, if $\forall A.C$ leads to the presence of $\forall A'.C$ in some successor node as a result of repeated applications of the $R\forall_2$ -rule, then $\forall A.C$ is equivalent to $\forall A'.C$ iff $A = A'$. As A and A' are numbers, such comparisons are very easy, and minimisation of automata avoids unnecessary blocking delays.

The implementation is still at the “beta” stage, but it has been possible to carry out some preliminary tests using the

well known Galen medical terminology KB [Rector and Horrocks, 1997; Horrocks, 1998]. This KB contains 2,740 named concepts and 413 roles, 26 of which are transitive. The roles are arranged in a relatively complex hierarchy with a maximum depth of 10. Classifying this KB using FaCT's *SHIQ* reasoner takes 116s on an 800 MHz Pentium III equipped Linux PC. Classifying the same KB using the new *RIQ* reasoner took a total of 275s, but this includes 135s to compute the minimal DFAs for the role box (it should be noted that this is an unusually large and complex role box, and that computing the DFAs is a preprocessing step that will not need to be repeated when the remainder of the KB is extended, modified, or queried). This result is encouraging as it shows that, in the case of the Galen KB at least, using automata in $\forall A.C$ concepts does not, in itself, lead to an unacceptable degradation in performance.

The KB was then extended with several RIAs that express the propagation of location across various partonomic roles. These included

hasLocation ◦ isSolidDivisionOf $\sqsubseteq$ hasLocation,
hasLocation ◦ isLayerOf $\sqsubseteq$ hasLocation.

Classifying the extended KB took 280s, an increase of only 2% (3.5% if we exclude the DFA computation time). Subsumption queries w.r.t. this KB revealed that, e.g.,

Fracture $\sqcap$ $\exists$ hasLocation.NeckOfFemur
was implicitly a kind of
Fracture $\sqcap$ $\exists$ hasLocation.Femur

(NeckOfFemur is a solid division of Femur), and

Ulcer $\sqcap$ $\exists$ hasLocation.GastricMucosa
was implicitly a kind of
Ulcer $\sqcap$ $\exists$ hasLocation.Stomach

(GastricMucosa is a layer of Stomach). None of these subsumption relationships held w.r.t. the original KB. The times taken to compute these relationships w.r.t. the classified KB could not be measured accurately as they were of the same order as a system clock tick (10ms).

References

- [Baldoni *et al.*, 1998] M. Baldoni, L. Giordano, and A. Martelli. A tableau calculus for multimodal logics and some (un)decidability results. In *Proc. of TABLEAUX-98*, vol. 1397 of *LNAI*. Springer-Verlag, 1998.
- [Baldoni, 1998] M. Baldoni. *Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension*. PhD thesis, Dip. di Informatica, Univ. degli Studi di Torino, Italy, 1998.
- [Berger, 1966] R. Berger. The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66, 1966.
- [Brachman and Schmolze, 1985] R. J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [Demri, 2001] S. Demri. The complexity of regularity in grammar logics and related modal logics. *J. of Logic and Computation*, 11(6), 2001.
- [Farin  s del Cerro and Penttonen, 1988] L. Farin  s del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121-122:123–134, 1988.
- [Haarslev and M  ller, 2001] V. Haarslev and R. M  ller. RACER system description. In *IJCAR-01*, vol. 2083 of *LNAI*. Springer-Verlag, 2001.
- [Horrocks and Sattler, 2002a] I. Horrocks and U. Sattler. Decidability of *SHIQ* with complex role inclusion axioms. LTCS-Report 02-06, Dresden University of Technology, Germany, 2002, available at <http://lat.inf.tu-dresden.de/research/reports.html>.
- [Horrocks and Sattler, 2002b] I. Horrocks and U. Sattler. Optimised reasoning for shiq. In *Proc. of ECAI 2002*, IOS Press, 2002.
- [Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, vol. 1705 in *LNAI*. Springer-Verlag, 1999.
- [Horrocks, 1998] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proc. of KR-98*, Morgan Kaufmann, 1998.
- [Lenat and Guha, 1989] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems*. Addison Wesley, 1989.
- [Mohri *et al.*, 1998] M. Mohri, F. C. N. Pereira, and M. Riley. A Rational Design for a Weighted Finite-State Transducer Library. vol. 1436 in *LNCS*. Springer-Verlag, 1998.
- [Padgham and Lambrix, 1994] L. Padgham and P. Lambrix. A framework for part-of hierarchies in terminological logics. In *Proc. of KR-94*, Morgan Kaufmann, 1994.
- [Rector and Horrocks, 1997] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. of the WS on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, 1997.
- [Rector *et al.*, 1997] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.
- [Rector, 2002] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proc. of DL 2002*, CEUR-WS, 2002.
- [Sattler, 2000] U. Sattler. Description logics for the representation of aggregated objects. In *Proc. of ECAI 2000*. IOS Press, 2000.
- [Schild, 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, 1991.
- [Schmidt-Schau   and Smolka, 1991] M. Schmidt-Schau   and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
- [Schmidt-Schauss, 1989] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proc. of KR-89*, Morgan Kaufmann, 1989.
- [Schulz and Hahn, 2001] S. Schulz and U. Hahn. Parts, locations, and holes - formal reasoning about anatomical structures. In *Proc. of AIME 2001*, vol. 2101 of *LNAI*. Springer-Verlag, 2001.
- [Spackman, 2000] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.
- [Tobies, 2001] S. Tobies. PSPACE reasoning for graded modal logics. *J. of Logic and Computation*, 11(1):85–106, 2001.
- [Wessel, 2001] M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In *Proc. of DL 2001*, CEUR-WS, 2001.

Keys, Nominals, and Concrete Domains

Carsten Lutz¹

Carlos Areces²

Ian Horrocks³

Ulrike Sattler¹

¹Inst. for Theoretical Computer Science
Technical University Dresden
Germany
<lastname>@tcs.inf.tu-dresden.de

² INRIA Lorraine
Nancy
France
Carlos.Areces@loria.fr

³Dept. of Computer Science
University of Manchester
UK
horrocks@cs.man.ac.uk

Abstract

Many description logics (DLs) combine knowledge representation on an abstract, logical level with an interface to “concrete” domains such as numbers and strings. We propose to extend such DLs with key constraints that allow the expression of statements like “US citizens are uniquely identified by their social security number”. Based on this idea, we introduce a number of natural description logics and present (un)decidability results and tight NEXPTIME complexity bounds.

1 Introduction

Description Logics (DLs) are a family of popular knowledge representation formalisms. Many expressive DLs combine powerful logical languages with an interface to concrete domains (e.g., integers, reals, strings) and built-in predicates (e.g., $<$, sub-string-of) [Lutz, 2002b]. These can be used to form descriptions such as “employee working for the government and earning more than her boss” that combine “abstract” logical components (e.g., working for the government) with components using concrete domains and predicates (e.g., a numerical comparison of earnings).

DLs with concrete domains have turned out to be useful for reasoning about conceptual (database) models [Lutz, 2002e], and as the basis for expressive ontology languages [Horrocks *et al.*, 2002]. So far, however, they have not been able to express *key constraints*, i.e., constraints expressing the fact that certain “concrete features” uniquely determine the identity of the instances of a certain class. E.g., the concrete feature “social security number (SSN)” might serve as a key for citizens of the US, and the combination of identification number and manufacturer might serve as a key for vehicles. Such constraints are important both in databases and in realistic ontology applications. In a DL context, key constraints have so far only been considered on logical, “non-concrete” domains [Borgida and Weddell, 1997; Calvanese *et al.*, 2000; Khizder *et al.*, 2001; Toman and Weddell, 2002].

It is easy to see that concrete keys can express *nominals*, i.e., concepts to be interpreted as singleton sets (closely related to the “one-of” operator): e.g., if SSN is a key for Human (SSN keyfor Human), then the concept “Human with SSN 1234” ($\text{Human} \sqcap \exists \text{SSN}. =_{1234}$) has at most one instance.

In this paper, we extend the well-known DLs with concrete domains $\mathcal{ALC}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$ [Baader and Hanschke, 1991; Horrocks and Sattler, 2001] with key constraints and analyse the complexity of reasoning with the resulting logics $\mathcal{ALCOK}(\mathcal{D})$ and $\mathcal{SHOQK}(\mathcal{D})$. We show that allowing complex concepts to occur in key constraints dramatically increases the complexity of $\mathcal{ALC}(\mathcal{D})$ (which is PSPACE-complete): it becomes undecidable. Restricting key constraints to atomic concepts (such as “Human” in the above example) still yields a NEXPTIME-hard formalism, even for rather simple (PTIME) concrete domains. We show several variants of this result that depend on other characteristics of key constraints, such as the number of concrete features and the “path length”. This effect is consistent with the observation that the PSPACE upper bound for $\mathcal{ALC}(\mathcal{D})$ is not robust [Lutz, 2003].

Additionally, we prove the NEXPTIME bounds to be tight by presenting tableau algorithms for $\mathcal{ALCOK}(\mathcal{D})$ and $\mathcal{SHOQK}(\mathcal{D})$ with *key admissible* concrete domains that are in NP, where key admissibility is a simple and natural property. We have chosen to devise tableau algorithms since they have the potential to be implemented in efficient reasoners and have been shown to behave well in practise [Horrocks *et al.*, 2000]. Due to space restrictions, we can only sketch proofs and refer to [Lutz *et al.*, 2002] for more details.

2 Preliminaries

First, we formally introduce the description logic $\mathcal{ALCOK}(\mathcal{D})$.

Definition 1. A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a set and $\Phi_{\mathcal{D}}$ a set of predicate names. Each predicate name $P \in \Phi_{\mathcal{D}}$ is associated with an arity n and an n -ary predicate $P^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$.

Let N_C, N_O, N_R, N_{CF} be pairwise disjoint and countably infinite sets of *concept names*, *nominals*, *role names*, and *concrete features*. We assume that N_R has a countably infinite subset N_{aF} of *abstract features*. A *path* u is a composition $f_1 \cdots f_n g$ of n abstract features $f_1, \dots, f_n$ ($n \geq 0$) and a concrete feature g . Let $\mathcal{D}$ be a concrete domain. The set of $\mathcal{ALCOK}(\mathcal{D})$ -concepts is the smallest set such that (i) every concept name and every nominal is a concept, and (ii) if C and D are concepts, R is a role name, g is a concrete feature, $u_1, \dots, u_n$ are paths, and $P \in \Phi_{\mathcal{D}}$ is a predicate of arity n ,

then the following expressions are also concepts:

$\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\exists u_1, \dots, u_n.P$, and $g \uparrow$.

A *key definition* is an expression $(u_1, \dots, u_k \text{ keyfor } C)$ for $u_1, \dots, u_k$ ($k \geq 1$) paths and C a concept. A finite set of key definitions is called a *key box*. $\diamond$

As usual, we use $\top$ to denote an arbitrary propositional tautology. Throughout this paper, we will consider several fragments of the logic $\mathcal{ALCO}(\mathcal{D})$: $\mathcal{ALCO}(\mathcal{D})$ is obtained from $\mathcal{ALCO}(\mathcal{D})$ by admitting only empty key boxes; by disallowing the use of nominals, we obtain the fragment $\mathcal{ALC}(\mathcal{D})$ of $\mathcal{ALCO}(\mathcal{D})$ and $\mathcal{ALCK}(\mathcal{D})$ of $\mathcal{ALCO}(\mathcal{D})$.

The description logic $\mathcal{ALCO}(\mathcal{D})$ is equipped with a Tarski-style set-theoretic semantics. Along with the semantics, we introduce the standard inference problems: concept satisfiability and concept subsumption.

Definition 2. An *interpretation* $\mathcal{I}$ is a pair $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta_{\mathcal{I}}$ is a non-empty set, called the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function*. The interpretation function maps each concept name C to a subset $C^{\mathcal{I}}$ of $\Delta_{\mathcal{I}}$, each nominal N to a singleton subset $N^{\mathcal{I}}$ of $\Delta_{\mathcal{I}}$, each role name R to a subset $R^{\mathcal{I}}$ of $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$, each abstract feature f to a partial function $f^{\mathcal{I}}$ from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{I}}$, and each concrete feature g to a partial function $g^{\mathcal{I}}$ from $\Delta_{\mathcal{I}}$ to $\Delta_{\mathcal{D}}$.

If $u = f_1 \dots f_n g$ is a path and $d \in \Delta_{\mathcal{I}}$, then $u^{\mathcal{I}}(d)$ is defined as $g^{\mathcal{I}}(f_n^{\mathcal{I}} \dots (f_1^{\mathcal{I}}(d)) \dots)$. The interpretation function is extended to arbitrary concepts as follows:

$$\begin{aligned} (\neg C)^{\mathcal{I}} &:= \Delta_{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (C \sqcup D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\ (\exists R.C)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \exists e \in \Delta_{\mathcal{I}} : (d, e) \in R^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \\ (\forall R.C)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \forall e \in \Delta_{\mathcal{I}} : (d, e) \in R^{\mathcal{I}} \rightarrow e \in C^{\mathcal{I}}\} \\ (g \uparrow)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid g^{\mathcal{I}}(d) \text{ undefined}\} \\ (\exists u_1, \dots, u_n.P)^{\mathcal{I}} &:= \{d \in \Delta_{\mathcal{I}} \mid \exists x_1, \dots, x_n \in \Delta_{\mathcal{D}} : \\ &\quad u_i^{\mathcal{I}}(d) = x_i \text{ and } (x_1, \dots, x_n) \in P^{\mathcal{D}}\} \end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* of a concept C iff $C^{\mathcal{I}} \neq \emptyset$. Moreover, $\mathcal{I}$ *satisfies* a key definition $(u_1, \dots, u_n \text{ keyfor } C)$ if, for any $a, b \in C^{\mathcal{I}}$, $u_i^{\mathcal{I}}(a) = u_i^{\mathcal{I}}(b)$ for $1 \leq i \leq n$ implies $a = b$. $\mathcal{I}$ is a *model* of a key box $\mathcal{K}$ iff $\mathcal{I}$ satisfies all key definitions in $\mathcal{K}$. A concept C is *satisfiable w.r.t. a key box* $\mathcal{K}$ iff C and $\mathcal{K}$ have a common model. C is *subsumed by* a concept D w.r.t. $\mathcal{K}$ (written $C \sqsubseteq_{\mathcal{K}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{K}$. $\diamond$

It is well-known that, in DLs providing for negation, subsumption can be reduced to (un)satisfiability and vice versa: $C \sqsubseteq_{\mathcal{K}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{K}$ and C is satisfiable w.r.t. $\mathcal{K}$ iff $C \not\sqsubseteq_{\mathcal{K}} \neg \top$. Thus we can concentrate on concept satisfiability when investigating the complexity of reasoning: the above reduction implies the corresponding bounds for subsumption and the complementary complexity class (usually co-NEXPTIME in this paper).

When devising decision procedures for DLs which are not tied to a *particular* concrete domain, *admissibility* of the concrete domain usually serves as a well-defined interface between the decision procedure and concrete domain reasoners [Baader and Hanschke, 1991; Lutz, 2002b]:

Definition 3. Let $\mathcal{D}$ be a concrete domain. A $\mathcal{D}$ -*conjunction* is a (finite) predicate conjunction of the form

$$c = \bigwedge_{i < k} P_i(x_0^{(i)}, \dots, x_{n_i}^{(i)}),$$

where P_i is an n_i -ary predicate for $i < k$ and the $x_j^{(i)}$ are variables. A $\mathcal{D}$ -conjunction c is *satisfiable* iff there exists a function δ mapping the variables in c to elements of $\Delta_{\mathcal{D}}$ such that $(\delta(x_0^{(i)}), \dots, \delta(x_{n_i}^{(i)})) \in P_i^{\mathcal{D}}$ for each $i < k$. We say that the concrete domain $\mathcal{D}$ is *admissible* iff (i) $\Phi_{\mathcal{D}}$ contains a name $\top_{\mathcal{D}}$ for $\Delta_{\mathcal{D}}$; (ii) $\Phi_{\mathcal{D}}$ is closed under negation, and (iii) satisfiability of $\mathcal{D}$ -conjunctions is decidable. We refer to the satisfiability of $\mathcal{D}$ -conjunctions as $\mathcal{D}$ -*satisfiability*. $\diamond$

As we shall see, it sometimes makes a considerable difference w.r.t. complexity and decidability to restrict key boxes in various ways. Because of this, it is convenient to introduce the following notions:

Definition 4. A key box $\mathcal{K}$ is called *Boolean* if all concepts appearing in (key definitions in) $\mathcal{K}$ are Boolean combinations of concept names; *path-free* if all key definitions in $\mathcal{K}$ are of the form $(g_1, \dots, g_n \text{ keyfor } C)$ with $g_1, \dots, g_n \in \mathcal{N}_{\text{CF}}$; *simple* if it is both path-free and Boolean; and a *unary key box* if all key definitions in $\mathcal{K}$ are of the form $(u \text{ keyfor } C)$. A concept C is called *path-free* if, in all its subconcepts of the form $\exists u_1, \dots, u_n.P$, $u_1, \dots, u_n$ are concrete features. $\diamond$

To emphasize that a key box must *not* necessarily be Boolean or path-free, we sometimes call such a key box *general*. Similarly, to emphasize that a key box is not necessarily a unary key box, we sometimes call such a key box *n-ary key box*.

3 Lower Complexity Bounds

In this section, we present lower complexity bounds for DLs with concrete domains, key boxes and nominals. We start by showing that satisfiability of $\mathcal{ALCK}(\mathcal{D})$ -concepts w.r.t. general key boxes is undecidable for many interesting concrete domains. This discouraging result is relativized by the fact that, as shown in Section 4, the restriction to Boolean key boxes recovers decidability. Next, we prove that satisfiability of path-free $\mathcal{ALCK}(\mathcal{D})$ -concepts w.r.t. simple key boxes is NEXPTIME-hard for many concrete domains and that this holds even if we restrict ourselves to unary key boxes. Finally, we identify a concrete domain such that $\mathcal{ALCO}(\mathcal{D})$ -concept satisfiability (without key boxes) is already NEXPTIME-hard.

Undecidability of $\mathcal{ALCK}(\mathcal{D})$ -concept satisfiability w.r.t. general key boxes is proved by reduction of the undecidable Post Correspondence Problem (PCP) [Post, 1946].

Definition 5. An *instance* P of the PCP is given by a finite, non-empty list $(\ell_1, r_1), \dots, (\ell_k, r_k)$ of pairs of words over some alphabet Σ . A sequence of integers $i_1, \dots, i_m$, with $m \geq 1$, is called a *solution* for P iff $\ell_{i_1} \dots \ell_{i_m} = r_{i_1} \dots r_{i_m}$. The problem is to decide whether a given instance P has a solution. $\diamond$

The reduction uses the admissible concrete domain $\mathcal{W}$ introduced in [Lutz, 2003], whose domain is the set of words over Σ and whose predicates express concatenation of words. For each PCP instance $P = (\ell_1, r_1), \dots, (\ell_k, r_k)$, we define a

concept C_P and unary key box K_P such that P has no solution iff C_P is satisfiable w.r.t. K_P . Intuitively, C_P and K_P enforce an infinite, k -ary tree, where each node represents a sequence of integers, i.e. a potential solution. The role of the key box is to guarantee that the tree is of infinite depth; concrete features are used to store the left and right concatenations corresponding to the potential solutions; and concatenation predicates from the concrete domain W are used to compute them. Finally, an inequality predicate also provided by W is used to guarantee that none of the potential solutions is indeed a solution. Since it is known that W -satisfiability is in PTIME [Lutz, 2003], we obtain the following theorem.

Theorem 6. *There exists a concrete domain $\mathcal{D}$ such that $\mathcal{D}$ -satisfiability is in PTIME and satisfiability of $\mathcal{ALCK}(\mathcal{D})$ -concepts w.r.t. (general) unary key boxes is undecidable.*

As shown in [Lutz, 2003; Lutz et al., 2002], the reduction can easily be adapted to more natural concrete domains such as numerical ones based on the integers and providing predicates for equality to zero and one, binary equality, addition, and multiplication.

We now establish lower bounds for $\mathcal{ALCK}(\mathcal{D})$ with Boolean key boxes and for $\mathcal{ALCO}(\mathcal{D})$. These results are obtained using a NEXPTIME-complete variant of the well-known, undecidable domino problem [Knuth, 1968].

Definition 7. A domino system $\mathfrak{D}$ is a triple (T, H, V) , where $T \subset \mathbb{N}$ is a finite set of tile types and $H, V \subseteq T \times T$ represent the horizontal and vertical matching conditions. For $\mathfrak{D}$ a domino system and $a = a_0, \dots, a_{n-1} \in T^n$ an initial condition, a mapping $\tau : \{0, \dots, 2^{n+1}\} \times \{0, \dots, 2^{n+1}\} \rightarrow T$ is a solution for $\mathfrak{D}$ and a iff, for all $x, y < 2^{n+1}$, the following holds: (i) if $\tau(x, y) = t$ and $\tau(x+1 \bmod 2^{n+1}, y) = t'$, then $(t, t') \in H$; (ii) if $\tau(x, y) = t$ and $\tau(x, y+1 \bmod 2^{n+1}) = t'$, then $(t, t') \in V$; and (iii) $\tau(i, 0) = a_i$ for $i < n$. $\diamond$

This variant of the domino problem is NEXPTIME-complete [Lutz, 2003]. The three NEXPTIME lower bounds are obtained by using suitable and admissible concrete domains D_1 , D_2 , and D_3 to reduce the above domino problem. More precisely, the simplest concrete domain D_1 is used in the reduction to $\mathcal{ALCK}(D_1)$ -concept satisfiability w.r.t. Boolean (n -ary) key boxes, the slightly more complex D_2 is used in the reduction to $\mathcal{ALCK}(D_2)$ -concept satisfiability w.r.t. Boolean unary key boxes, and the most powerful concrete domain D_3 is used in the reduction to $\mathcal{ALCO}(D_3)$ -concept satisfiability without key boxes.

The idea underlying all three reductions is to use concept names $X_0, \dots, X_n, Y_0, \dots, Y_n$ to represent positions in the $2^{n+1} \times 2^{n+1}$ -torus: if a is a domain element representing the position (i, j) , then $a \in X_\ell^T$ expresses that the ℓ -th bit in the binary coding of i is 1, and $a \in Y_\ell^T$ expresses that the ℓ -th bit of j is 1. We use standard methods to enforce that there exists a domain element for every position in the torus. The main difference between the three reductions is how it is ensured that no position is represented by two different domain elements—we call this *uniqueness of positions*.

The first reduction uses the very simple concrete domain D_1 , which is based on the set $\{0, 1\}$ and only provides unary predicates $=_0$, $=_1$, and their negations. Uniqueness of positions is ensured by translating the position (i, j)

of a domain element a into concrete domain values: for $xpos_\ell \in N_{cF}$, we enforce that $xpos_\ell^T(a) = 1$ if $a \in X_\ell^T$ and 0 otherwise (analogously for $ypos_\ell$ and Y_ℓ^T). Then the key definition $(xpos_0, \dots, xpos_n, ypos_0, \dots, ypos_n \text{ keyfor } \top)$ obviously ensures uniqueness of positions. Since the reduction concept is path-free and D_1 -satisfiability is easily seen to be in PTIME, we obtain the following:

Theorem 8. *D_1 -satisfiability is in PTIME and satisfiability of path-free $\mathcal{ALCK}(D_1)$ -concepts w.r.t. simple key boxes is NEXPTIME-hard.*

The (somewhat artificial) concrete domain D_1 can be replaced by many natural concrete domains $\mathcal{D}$ proposed in the literature [Baader and Hanschke, 1992; Haarslev and Möller, 2002; Lutz, 2002b; 2002d]: it suffices that $\mathcal{D}$ provides two unary predicates denoting disjoint singleton sets.

The second reduction uses the more complex concrete domain D_2 , which “stores” whole bit vectors rather than only single bits. In D_2 , we can translate the position (i, j) of an element a from concepts X_ℓ, Y_k into a single bit vector of length $2(n+1)$ that is then stored as a bv-successor of a , where bv is a concrete feature. Since we replaced the $2(n+1)$ concrete features used in the first reduction (one for each bit) by the single feature bv, it now suffices to use the simple unary key box (bv keyfor $\top$) to ensure uniqueness of positions. As in D_1 , the reduction concept is path-free. In [Lutz et al., 2002], it is shown that D_2 -satisfiability is in PTIME.

Theorem 9. *D_2 -satisfiability is in PTIME and satisfiability of path-free $\mathcal{ALCK}(D_2)$ -concepts w.r.t. simple unary key boxes is NEXPTIME-hard.*

Again, the artificial concrete domain D_2 can be replaced by more natural ones: we can simulate bit vectors using integers and the necessary operations on bit vectors by unary predicates $=_n$ for every integer n and a ternary addition predicate—for more details see [Lutz et al., 2002].

The last lower bound is concerned with the DL $\mathcal{ALCO}(\mathcal{D})$. In the absence of key boxes, we need a different reduction strategy and the more complex concrete domain D_3 , which extends D_2 with so-called domino arrays that allow us to store the tiling of the whole torus in a single concrete domain value. We can then ensure uniqueness of positions using a single nominal. Computationally, the concrete domain D_3 is still very simple, namely in PTIME. However, it no longer suffices to use only path-free concepts.

Theorem 10. *D_3 -satisfiability is in PTIME and satisfiability of $\mathcal{ALCO}(D_3)$ -concepts is NEXPTIME-hard.*

4 Reasoning Procedures

We describe two tableau-based decision procedures for concept satisfiability in DLs with concrete domains, nominals, and keys. The first is for $\mathcal{ALCOK}(\mathcal{D})$ -concepts w.r.t. Boolean key boxes. This algorithm yields a NEXPTIME upper complexity bound matching the lower bounds established in Section 3. The second procedure is for $\mathcal{SHOQK}(\mathcal{D})$ w.r.t. path-free key boxes and also yields a tight NEXPTIME upper complexity bound. $\mathcal{SHOQK}(\mathcal{D})$ is an extension of the DL $\mathcal{SHOQ}(\mathcal{D})$ introduced in [Horrocks and Sattler, 2001;

Pan and Horrocks, 2002], which provides a wealth of expressive possibilities such as transitive roles, role hierarchies, nominals, qualifying number restrictions, and general TBoxes with a path-free concrete domain constructor and path-free key boxes. Path-freeness of $\mathcal{SHOQK}(\mathcal{D})$'s concrete domain constructor is crucial for decidability. Moreover, it allows us to admit general rather than only Boolean key boxes.

Tableau algorithms decide the satisfiability of the input concept (in our case w.r.t. the input key box) by attempting to construct a model for it: starting with an initial data structure induced by the input concept, the algorithm repeatedly applies completion rules. Eventually, the algorithm either finds an obvious contradiction or it encounters a contradiction-free situation in which no more completion rules are applicable. In the former case the input concept is unsatisfiable, while in the latter case it is satisfiable.

Existing tableau algorithms for DLs with concrete domains use admissibility as an “interface” between the tableau algorithm and a concrete domain reasoner [Lutz, 2002b; Baader and Hanschke, 1991]. In the presence of keys, this is not enough: besides knowing whether a given $\mathcal{D}$ -conjunction is satisfiable, the concrete domain reasoner has to provide information on variables that must take the same value in solutions. As an example, consider the concrete domain $\mathbb{N} = (\mathbb{N}, \{<_n \mid n \in \mathbb{N}\})$ and the $\mathbb{N}$ -conjunction $c = <_2(v_1) \wedge <_2(v_2) \wedge <_2(v_3)$. Obviously, every solution δ for c identifies two of the variables v_1, v_2, v_3 . This information has to be passed from the concrete domain reasoner to the tableau algorithm since, in the presence of key boxes, it may have an impact on the satisfiability of the input concept. E.g., this information transfer reveals the unsatisfiability of $\exists R. A \sqcap \exists R. (\neg A \sqcap B) \sqcap \exists R. (\neg A \sqcap \neg B) \sqcap \forall R. \exists g. <_2$ w.r.t. g key for $\mathbb{T}$. To formalize this requirement, we strengthen the notion of admissibility into *key-admissibility*.

Definition 11. A concrete domain $\mathcal{D}$ is *key-admissible* iff (i) $\Phi_{\mathcal{D}}$ contains a name $\top_{\mathcal{D}}$ for $\Delta_{\mathcal{D}}$; (ii) $\Phi_{\mathcal{D}}$ is closed under negation, and (iii) there exists an algorithm that takes as input a $\mathcal{D}$ -conjunction c , returns clash if c is unsatisfiable, and otherwise non-deterministically outputs an equivalence relation $\sim$ on the set of variables V used in c such that there exists a solution δ for c with the following property: for all $v, v' \in V$ $\delta(v) = \delta(v')$ iff $v \sim v'$. Such an equivalence relation is henceforth called a *concrete equivalence*. We say that *extended $\mathcal{D}$ -satisfiability is in NP* if there exists an algorithm as above running in polynomial time. $\diamond$

It can easily be seen that any concrete domain that is admissible and provides for an equality predicate is also key-admissible [Lutz et al., 2002].

In the following, we assume that all concepts (the input concept and those occurring in key boxes) are in *negation normal form (NNF)*, i.e., negation occurs only in front of concept names and nominals; if the concrete domain $\mathcal{D}$ is admissible, then every $\mathcal{ALCOK}(\mathcal{D})$ -concept can be converted into an equivalent one in NNF [Lutz et al., 2002]. We use $\neg C$ to denote the result of converting the concept $\neg C$ into NNF, $\text{sub}(C)$ to denote the set of subconcepts of C , and $\text{sub}(\mathcal{K})$ to denote the set of subconcepts of all concepts occurring in key

box $\mathcal{K}$. Moreover, we use $\text{cl}(C, \mathcal{K})$ as abbreviation for the set

$$\text{sub}(C) \cup \text{sub}(\mathcal{K}) \cup \{\neg D \mid D \in \text{sub}(\mathcal{K})\}.$$

Complexity of $\mathcal{ALCOK}(\mathcal{D})$

We start the presentation of the $\mathcal{ALCOK}(\mathcal{D})$ tableau algorithm by introducing the underlying data structure.

Definition 12. Let O_a and O_c be disjoint and countably infinite sets of *abstract* and *concrete nodes*. A *completion tree* for an $\mathcal{ALCOK}(\mathcal{D})$ -concept C and a key box $\mathcal{K}$ is a finite, labeled tree $(V_a, V_c, E, \mathcal{L})$ with a set of nodes $V_a \cup V_c$ such that $V_a \subseteq O_a$, $V_c \subseteq O_c$, and all nodes from V_c are leaves. Each node $a \in V_a$ of the tree is labeled with a subset $\mathcal{L}(a)$ of $\text{cl}(C, \mathcal{K})$; each edge $(a, b) \in E$ with $a, b \in V_a$ is labeled with a role name $\mathcal{L}(a, b)$ occurring in C or $\mathcal{K}$; and each edge $(a, x) \in E$ with $a \in V_a$ and $x \in V_c$ is labeled with a concrete feature $\mathcal{L}(a, x)$ occurring in C or $\mathcal{K}$.

For $\mathbf{T} = (V_a, V_c, E, \mathcal{L})$ and $a \in V_a$, we use $\text{lev}_{\mathbf{T}}(a)$ to denote the depth at which a occurs in $\mathbf{T}$ (starting with the root node at depth 0). A *completion system* for an $\mathcal{ALCOK}(\mathcal{D})$ -concept C and a key box $\mathcal{K}$ is a tuple $(\mathbf{T}, \mathcal{P}, \prec, \sim)$, where $\mathbf{T} = (V_a, V_c, E, \mathcal{L})$ is a completion tree for C and $\mathcal{K}$, $\mathcal{P}$ is a function mapping each $P \in \Phi_{\mathcal{D}}$ with arity n appearing in C to a subset of V_c^n , $\prec$ is a linear ordering of V_a such that $\text{lev}_{\mathbf{T}}(a) \leq \text{lev}_{\mathbf{T}}(b)$ implies $a \prec b$, and $\sim$ is an equivalence relation on V_c .

Let $\mathbf{T} = (V_a, V_c, E, \mathcal{L})$ be a completion tree. A node $b \in V_a$ is an *R-successor* of a node $a \in V_a$ if $(a, b) \in E$ and $\mathcal{L}(a, b) = R$. Similarly, a node $x \in V_c$ is a *g-successor* of a if $(a, x) \in E$ and $\mathcal{L}(a, x) = g$. For paths u , the notion *u-successor* is defined in the obvious way. $\diamond$

Intuitively, the relation $\sim$ records equalities between concrete nodes that have been found during the model construction process. The relation $\sim$ induces an equivalence relation $\approx_a$ on abstract nodes which, in turn, yields the equivalence relation $\approx_c \supseteq \sim$ on concrete nodes.

Definition 13. Let $S = (\mathbf{T}, \mathcal{P}, \prec, \sim)$ be a completion system for a concept C and a key box $\mathcal{K}$ with $\mathbf{T} = (V_a, V_c, E, \mathcal{L})$, and let $\approx$ be an equivalence relation on V_a . For each $R \in \mathbb{N}_R$, a node $b \in V_a$ is an *R/ $\approx$ -neighbor* of a node $a \in V_a$ if there exists a node $c \in V_a$ such that $a \approx c$ and b is an R -successor of c . For paths u , the notion *u/ $\approx$ -neighbor* is defined analogously.

We define a sequence of equivalence relations $\approx_a^0 \subseteq \approx_a^1 \subseteq \dots$ on V_a as follows:

$$\begin{aligned} \approx_a^0 &= \{(a, a) \mid a \in V_a\} \\ &\cup \{(a, b) \in V_a^2 \mid \exists N \in \mathbb{N}_O : N \in \mathcal{L}(a) \cap \mathcal{L}(b)\} \\ \approx_a^{i+1} &= \approx_a^i \cup \\ &\quad \{(a, b) \in V_a^2 \mid \exists c \in V_a, f \in \mathbb{N}_{aF} : \\ &\quad \quad a \text{ and } b \text{ are } f/\approx_a^i\text{-neighbors of } c\} \cup \\ &\quad \{(a, b) \in V_a^2 \mid \exists (u_1, \dots, u_n \text{ key for } D) \in \mathcal{K}, \\ &\quad \quad \exists x_1, \dots, x_n : x_j \text{ is } u_j/\approx_a^i\text{-neighbor of } a, \\ &\quad \quad \exists y_1, \dots, y_n : y_j \text{ is } u_j/\approx_a^i\text{-neighbor of } b, \\ &\quad \quad D \in \mathcal{L}(a) \cap \mathcal{L}(b) \wedge (\forall 1 \leq j \leq n : x_j \sim y_j)\} \end{aligned}$$

Finally, set $\approx_a = \bigcup_{i \geq 0} \approx_a^i$, and define $x \approx_c y$ if $x \sim y$ or there are $a \in V_a$ and $g \in N_{cF}$ such that x and y are $g/\approx_a$ -neighbors of a . $\diamond$

Intuitively, if we have $a \approx_a b$, then a and b describe the same domain element of the constructed model (and similarly for the $\approx_c$ relation on concrete nodes).

Let $\mathcal{D}$ be a key-admissible concrete domain. To decide the satisfiability of an $\mathcal{ALCOK}(\mathcal{D})$ -concept C_0 w.r.t. a Boolean key box $\mathcal{K}$ (both in NNF), the tableau algorithm is started with the *initial completion system* $S_{C_0} = (\mathbf{T}_{C_0}, \mathcal{P}_\emptyset, \emptyset, \emptyset)$, where $\mathbf{T}_{C_0} = (\{a_0\}, \emptyset, \emptyset, \{a_0 \mapsto \{C_0\}\})$ and $\mathcal{P}_\emptyset$ maps each $P \in \Phi_{\mathcal{D}}$ occurring in C_0 to $\emptyset$. We now introduce an operation that is used by the completion rules to add new nodes to completion trees.

Definition 14. Let $S = (\mathbf{T}, \mathcal{P}, \prec, \sim)$ be a completion system with $\mathbf{T} = (V_a, V_c, E, \mathcal{L})$. An element of O_a or O_c is called *fresh* in $\mathbf{T}$ if it does not appear in $\mathbf{T}$. We use the following notions:

$S + aRb$: Let $a \in V_a$, $b \in O_a$ fresh in $\mathbf{T}$, and $R \in N_R$. We write $S + aRb$ to denote the completion system S' that can be obtained from S by adding (a, b) to E and setting $\mathcal{L}(a, b) = R$ and $\mathcal{L}(b) = \emptyset$. Moreover, b is inserted into $\prec$ such that $b \prec c$ implies $\text{lev}_{\mathbf{T}}(b) \leq \text{lev}_{\mathbf{T}}(c)$.

$S + agx$: Let $a \in V_a$, $x \in O_c$ fresh in $\mathbf{T}$ and $g \in N_{cF}$. We write $S + agx$ to denote the completion system S' that can be obtained from S by adding (a, x) to E and setting $\mathcal{L}(a, x) = g$.

When nesting $+$, we omit brackets writing, e.g., $S + aR_1b + bR_2c$ for $(S + aR_1b) + bR_2c$. Let $u = f_1 \cdots f_n g$ be a path. With $S + aux$, where $a \in V_a$ and $x \in O_c$ is fresh in $\mathbf{T}$, we denote the completion system S' that can be obtained from S by taking fresh nodes $b_1, \dots, b_n \in O_a$ and setting

$$S' := S + af_1b_1 + \cdots + b_{n-1}f_nb_n + b_n g x. \quad \diamond$$

The completion rules are given in Figure 1, where we assume that newly introduced nodes are always fresh. The $R\sqcup$ and Rch rules are non-deterministic and the upper five rules are well-known from existing tableau algorithms for $\mathcal{ALC}(\mathcal{D})$ -concept satisfiability (c.f. for example [Lutz, 2002d]). Only $R\forall$ deserves a comment: it considers $R/\approx_a$ -neighbors rather than R -successors since $\approx_a$ relates nodes denoting the same domain element.

The last two rules are necessary for dealing with key boxes. The “choose rule” Rch (c.f. [Hollunder and Baader, 1991; Horrocks *et al.*, 2000]) guesses whether an abstract node a satisfies C in case of C occurring in a key definition and a having neighbors for all paths u_i in this key definition. The Rp rule deals with equalities between abstract nodes as recorded by the $\approx_a$ relation: if $a \approx_a b$, then a and b describe the same element, and thus their node labels should be identical. We choose one representative for each equivalence class of $\approx_a$ (the node that is minimal w.r.t. $\prec$) and make sure that the representative’s node label contains the labels of all the nodes it represents.

Definition 15. Let $S = (\mathbf{T}, \mathcal{P}, \prec, \sim)$ be a completion system for a concept C and a key box $\mathcal{K}$ with $\mathbf{T} = (V_a, V_c, \prec, \sim)$. We

$R\sqcap$	if $C_1 \sqcap C_2 \in \mathcal{L}(a)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(a)$ then $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C_1, C_2\}$
$R\sqcup$	if $C_1 \sqcup C_2 \in \mathcal{L}(a)$ and $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$ then $\mathcal{L}(a) := \mathcal{L}(a) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
$R\exists$	if $\exists R.C \in \mathcal{L}(a)$ and there is no $R/\approx_a$ -neighbor b of a such that $C \in \mathcal{L}(b)$, then $S := S + aRb$ and $\mathcal{L}(b) := \{C\}$
$R\forall$	if $\forall R.C \in \mathcal{L}(a)$, b is an $R/\approx_a$ -neighbor of a , and $C \notin \mathcal{L}(b)$ then $\mathcal{L}(b) := \mathcal{L}(b) \cup \{C\}$
$R\exists c$	if $\exists u_1, \dots, u_n.P \in \mathcal{L}(a)$ and there are no u_i -successors x_i of a (for $1 \leq i \leq n$) with $(x_1, \dots, x_n) \in \mathcal{P}(P)$ then $S := (S + au_1x_1 + \cdots + au_nx_n)$ and $\mathcal{P}(P) := \mathcal{P}(P) \cup \{(x_1, \dots, x_n)\}$
Rch	if $(u_1, \dots, u_n \text{ keyfor } C) \in \mathcal{K}$, $\{C, \neg C\} \cap \mathcal{L}(a) = \emptyset$, and there are $u_i/\approx_a$ -neighbors x_i of a (for $1 \leq i \leq n$) then $\mathcal{L}(a) := \mathcal{L}(a) \cup \{D\}$ for some $D \in \{C, \neg C\}$
Rp	if $\mathcal{L}(b) \not\subseteq \mathcal{L}(a)$ and $a \in V_a$ is minimal w.r.t. $\prec$ such that $a \approx_a b$ then $\mathcal{L}(a) := \mathcal{L}(a) \cup \mathcal{L}(b)$

Figure 1: Completion rules for $\mathcal{ALCOK}(\mathcal{D})$.

say that the completion system S is *concrete domain satisfiable* iff the conjunction

$$\zeta_S = \bigwedge_{\substack{P \text{ used in } C \\ (x_1, \dots, x_n) \in \mathcal{P}(P)}} P(x_1, \dots, x_n) \wedge \bigwedge_{x \approx_c y} \neg(x, y)$$

is satisfiable. S contains a *clash* iff (i) there is an $a \in V_a$ and an $A \in N_c$ such that $\{A, \neg A\} \subseteq \mathcal{L}(a)$; (ii) there are $a \in V_a$ and $x \in V_c$ such that $g \uparrow \in \mathcal{L}(a)$ and x is $g/\approx_a$ -neighbor of a ; or (iii) S is not concrete domain satisfiable. If S does not contain a clash, then S is called *clash-free*. S is *complete* if no completion rule is applicable to S . $\diamond$

We now give the tableau algorithm in pseudocode notation, where *check* denotes the algorithm computing concrete equivalences as described in Definition 11:

```

define procedure sat( $S$ )
  do
    if  $S$  contains a clash then return unsatisfiable
     $\sim := \text{check}(\zeta_S)$ 
    compute  $\approx_a$  and then  $\approx_c$ 
  while  $\sim \neq \approx_c$ 
  if  $S$  contains a clash then return unsatisfiable
  if  $S$  is complete then return satisfiable
  apply a completion rule to  $S$  yielding  $S'$ 
  return sat( $S'$ )

```

The algorithm realizes a tight coupling between the concrete domain reasoner and the tableau algorithm: if the concrete domain reasoner finds that two concrete nodes are equal, the tableau algorithm may use this to deduce (via the computation of $\approx_a$ and $\approx_c$) even more equalities between concrete nodes. The concrete domain reasoner may then return in *check*(ζ_S) further “equalities” $\sim$ and so forth.

A similar interplay takes place in the course of several recursion steps: equalities of concrete nodes provided by the

concrete domain reasoner may make new rules applicable (for example R_p and then $R_{\exists c}$) which changes $\mathcal{P}$ and thus also ζ_S . This may subsequently lead to the detection of more equalities between concrete nodes by the concrete domain reasoner, and so forth. Note that, in the absence of keys boxes, there is much less interaction: it suffices to apply the concrete domain satisfiability check only once after the completion rules have been exhaustively applied [Baader and Hanschke, 1991].

In [Lutz *et al.*, 2002], we prove that the algorithm runs in non-deterministic exponential time: there are exponential bounds on the number of abstract and concrete nodes in the completion system, on the number of while loop iterations in each recursion step, and on the size of ζ_S . This yields the following upper bound, which is tight by Theorem 9.

Theorem 16. *For $\mathcal{D}$ a key-admissible concrete domain such that extended $\mathcal{D}$ -satisfiability is in NP, $\mathcal{ALCOK}(\mathcal{D})$ -concept satisfiability w.r.t. Boolean key boxes is in NEXPTIME.*

Complexity of $\mathcal{SHOQK}(\mathcal{D})$

We have designed a tableau algorithm for $\mathcal{SHOQK}(\mathcal{D})$ as a combination of the one for $\mathcal{SHOQ}(\mathcal{D})$ in [Horrocks and Sattler, 2001] and the one for $\mathcal{ALCOK}(\mathcal{D})$ presented above. It is restricted to path-free concepts and path-free key boxes, but can handle complex concepts in key boxes. The most important difference from the $\mathcal{ALCOK}(\mathcal{D})$ algorithm is as follows: in the presence of non-Boolean key boxes, the R_{ch} rule may add concepts of positive “role depth” to arbitrary nodes in the completion tree. Thus the role depth does not automatically decrease with the depth of nodes in the tree (as in the case of $\mathcal{ALCOK}(\mathcal{D})$) and a naive tableau algorithm would construct infinite trees. However, even for $\mathcal{SHOQ}(\mathcal{D})$ without key boxes, one has to enforce termination artificially by using a cycle detection mechanism called *blocking*—whereas the $\mathcal{ALCOK}(\mathcal{D})$ algorithm terminates “naturally”. It can be shown that blocking can be used in the presence of key boxes without corrupting soundness or completeness. A detailed description of this algorithm and a correctness proof is given in [Lutz *et al.*, 2002]. As a by-product of the $\mathcal{SHOQK}(\mathcal{D})$ tableau algorithm, we obtain a *small model property*: every satisfiable $\mathcal{SHOQK}(\mathcal{D})$ -concept has a model of size exponential in the concept length. Thus we obtain the following upper bound, which is tight by Theorem 9.

Theorem 17. *For $\mathcal{D}$ a key-admissible concrete domain such that $\mathcal{D}$ -satisfiability is in NP, $\mathcal{SHOQK}(\mathcal{D})$ -concept satisfiability w.r.t. path-free key boxes is in NEXPTIME.*

5 Summary

We have identified key boxes as an interesting extension of description logics with concrete domains, introduced a number of natural description logics, and provided a comprehensive analysis of the decidability and complexity of reasoning. Moreover, we have proposed tableau algorithms for two such (NEXPTIME-complete) logics.

The main result of our investigations is that key constraints are rather powerful, since they dramatically increase the complexity of reasoning: PSPACE $\mathcal{ALC}(\mathcal{D})$ becomes undecidable with unrestricted key boxes, and NEXPTIME-complete with Boolean key boxes—provided that the concrete domain $\mathcal{D}$ is not too complex, i.e., extended $\mathcal{D}$ -satisfiability is in NP.

References

- [Baader and Hanschke, 1991] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of IJCAI-91*, Morgan-Kaufmann, 1991.
- [Baader and Hanschke, 1992] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of GWAI-92*, vol. 671 of LNCS, Springer, 1992.
- [Borgida and Weddell, 1997] A. Borgida and G.E. Weddell. Adding uniqueness constraints to description logics (preliminary report). In *Proc. of DOOD97*, volume 1341 of LNCS, pages 85–102. Springer, 1997.
- [Calvanese *et al.*, 2000] D. Calvanese, G. De Giacomo, and M. Lenzerini. Keys for Free in Description Logics. In *Proc. of DL2000*, number 33 in <http://CEUR-WS.org/>, 2000.
- [Haarslev and Möller, 2002] V. Haarslev and R. Möller. Practical reasoning in racer with a concrete domain for linear inequations. In *Proc. of DL2002*, number 53 in <http://CEUR-WS.org/>, 2002.
- [Hollunder and Baader, 1991] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of KR'91*, Morgan-Kaufmann, 1991.
- [Horrocks and Sattler, 2001] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ description logic. In *Proc. of IJCAI-01*, Morgan-Kaufmann, 2001.
- [Horrocks *et al.*, 2000] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.
- [Horrocks *et al.*, 2002] I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In *Proc. of AAAI 2002*, 2002.
- [Khizder *et al.*, 2001] V. Khizder, D. Toman, and G. Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *Proc. of ICDT 2001*, 2001.
- [Knuth, 1968] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.
- [Lutz *et al.*, 2002] C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. LTCS-Report 02-04, Technical University Dresden, 2002. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [Lutz, 2002b] C. Lutz. Description logics with concrete domains—a survey. In *Proc. of AiML2002*, 2002.
- [Lutz, 2002d] C. Lutz. PSPACE reasoning with the description logic $\mathcal{ALCF}(\mathcal{D})$. *Logic J. of the IGPL*, 10(5):535–568, 2002.
- [Lutz, 2002e] C. Lutz. Reasoning about entity relationship diagrams with complex attribute dependencies. In *Proc. of DL2002*, number 53 in <http://CEUR-WS.org/>, 2002.
- [Lutz, 2003] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 2003. To appear.
- [Pan and Horrocks, 2002] J. Z. Pan and I. Horrocks. Reasoning in the $\mathcal{SHOQ}(\mathcal{D}_n)$ description logic. In *Proc. of DL2002*, number 53 in <http://CEUR-WS.org/>, 2002.
- [Post, 1946] E. M. Post. A variant of a recursively unsolvable problem. *Bull. of the Amer. Math. Soc.*, 52:264–268, 1946.
- [Toman and Weddell, 2002] D. Toman and G. Weddell. On reasoning about structural equality in XML: A description logic approach. In *Proc. of ICDT2003*, number 2572 in LNCS, pages 96–110, Springer, 2002.

Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies

Stefan Schlobach

Language and Inference Technology, ILLC
Universiteit van Amsterdam, Netherlands
schlobac@science.uva.nl

Ronald Cornet

Academic Medical Center
Universiteit van Amsterdam, Netherlands
R.Cornet@amc.uva.nl

Abstract

Current Description Logic reasoning systems provide only limited support for debugging logically erroneous knowledge bases. In this paper we propose new non-standard reasoning services which we designed and implemented to pinpoint logical contradictions when developing the medical terminology DICE. We provide complete algorithms for unfoldable $\mathcal{ALC}$ -TBoxes based on minimisation of axioms using Boolean methods for *minimal unsatisfiability-preserving sub-TBoxes*, and an incomplete bottom-up method for *generalised incoherence-preserving terminologies*.

1 Introduction

Our work was motivated by the development of the DICE¹ terminology. DICE implements frame-based definitions of diagnostic information for the unambiguous and unified classification of patients in Intensive Care medicine. The representation of DICE is currently being migrated to an expressive Description Logic (henceforth DL) to facilitate logical inferences. Figure 1 shows an extract of the DICE terminology. In [Cornet and Abu-Hanna, 2002] the authors describe the migration process in more detail. The resulting DL terminology (usually called a “TBox”) contains axioms such as the following, where classes (like `BODYPART`) are translated as concepts, and slots (like `REGION`) as roles:

$$\begin{aligned} \text{Brain} &\sqsubseteq \text{CentralNervousSystem} \sqcap \exists \text{systempart.NervousSystem} \sqcap \\ &\text{BodyPart} \sqcap \exists \text{region.HeadAndNeck} \sqcap \forall \text{region.HeadAndNeck} \\ \text{CentralNervousSystem} &\sqsubseteq \text{NervousSystem} \end{aligned}$$

Developing a coherent terminology is a time-consuming and error-prone process. DICE defines more than 2400 concepts and uses 45 relations. To illustrate some of the problems, take the definition of a “brain” which is incorrectly specified, among others, as a “central nervous-system” and “body-part” located in the head. This definition is contradictory as nervous-systems and body-parts are declared disjoint in DICE. Fortunately, current Description Logic reasoners,

such as RACER [Haarslev and Möller, 2001] or FaCT [Horrocks, 1998], can detect this type of inconsistency and the knowledge engineer can identify the cause of the problem. Unfortunately, many other concepts are defined based on the erroneous definition of “brain” forcing each of them to be erroneous as well. In practice, DL reasoners provide lists of hundreds of unsatisfiable concepts for the DICE TBox and the debugging remains a jigsaw to be solved by human experts, with little additional explanation to support this process.

By *debugging* we understand the identification and elimination of modelling errors when detecting logical contradictions in a knowledge base. Debugging requires an *explanation* for the logical incorrectness and, as a second step, its *correction*. In this paper we will focus on the former as the latter requires an understanding of the meaning of represented concepts. We define a number of new non-standard reasoning services to explain incoherences through *pinpointing*.

Our experience with debugging DICE provides some hands-on examples for the problem at hand: take the contradictory definition of brains in the DICE anatomy specification. What information is useful for correcting the knowledge base? First, we have to identify the precise position of errors within a TBox; that is, we need a procedure to single out the axioms causing the contradiction. The axioms for *Brain* and *CentralNervousSystem* form such a minimal incoherent subset of the DICE terminology. Formally, we introduce *minimal unsatisfiability-preserving sub-TBoxes* (abbreviated MUPS) and *minimal incoherence-preserving sub-TBoxes* (MIPS) as the smallest subsets of axioms of an incoherent terminology preserving unsatisfiability of a particular, respectively of at least one unsatisfiable concept. Secondly, we highlight the elements of these definitions containing the faulty specification. An axiom, e.g., $\text{Brain} \sqsubseteq \text{BodyPart} \sqcap \text{NervousSystem}$ points to the core of the erroneously modelled knowledge. For this purpose we define *generalised incoherence-preserving terminologies* (GIT) as sets of incoherent axioms, which are syntactically related to the original axioms, more general and have minimal structural complexity.

The remainder of this paper is organised as follows: Section 2 introduces the relevant DL concepts to make the paper self-contained. In Section 3 we introduce the new reasoning services and provide algorithms in Section 4. The paper finishes with a discussion of the results we obtained with our new methods on the terminology DICE.

¹DICE stands for “Diagnoses for Intensive Care Evaluation”. The development of the DICE terminology has been supported by the NICE foundation.

CLASS	SUPERCLASS	SLOT	SLOT-VALUE
BRAIN	BODYPART CENTRAL NERVOUS SYSTEM	REGION SYSTEM PART	HEAD AND NECK NERVOUS SYSTEM
CENTRAL NERVOUS SYSTEM	NERVOUS SYSTEM		

Figure 1: An extract from the DICE terminology (frame-based).

2 Description Logics

We shall not give a formal introduction into Description Logics here, but point to the new handbook [Baader *et al.*, 2003]. Briefly, DLs are set description languages with concepts (usually denoted by capital letters), interpreted as subsets of a domain, and roles which are binary relations, which we denote by small letters. In a terminological component $\mathcal{T}$ (called TBox) the interpretations of concepts can be restricted to the *models* of $\mathcal{T}$ by defining *axioms* of the form $C \sqsubseteq D$.² Based on this formal model-theoretic semantics, a TBox can be checked for *incoherence*, i.e., whether there are *unsatisfiable* concepts: concepts which are necessarily interpreted as the empty set in all models of the TBox. Other standard reasoning services include *subsumption* of two concepts C and D (a subset relation w.r.t. all models of $\mathcal{T}$). Subsumption without reference to a TBox is called *concept subsumption* and we write $C \sqsubseteq D$. Recently, a number of non-standard reasoning services have been defined, such as the *least common subsumer* of concepts or the *minimal rewriting* of concepts (see [Küsters, 2001] for an overview).

$\mathcal{ALC}$ is a simple yet relatively expressive DL with conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), negation ($\neg C$) and universal ($\forall r.C$) and existential quantification ($\exists r.C$). A TBox is called *unfoldable* if the left-hand sides of the axioms (the defined concepts) are atomic, and if the right-hand sides (the definitions) contain no direct or indirect reference to the defined concept [Nebel, 1990].

3 Explaining Logical Incoherences

In this section we study ways of *explaining incoherences* in DL terminologies. We propose to simplify a terminology $\mathcal{T}$ in order to reduce the available information to the root of the incoherence. More concretely we first exclude axioms which are irrelevant to the incoherence and then provide simplified definitions highlighting the exact position of a contradiction within the axioms of this reduced TBox. We will call the former *axiom pinpointing*, the latter *concept pinpointing*.

In this section we will formally introduce axiom and concept pinpointing for a general TBox without restrictions on the underlying representation language. In Section 4 we will give algorithms for the case of unfoldable $\mathcal{ALC}$ -TBoxes.

3.1 Axiom Pinpointing

Axiom pinpointing means identifying debugging-relevant axioms, where an axiom is *relevant* if a contradictory TBox becomes coherent once the axiom is removed or if, at least, a particular, previously unsatisfiable concept turns satisfiable.

²We will not consider assertional components in this paper.

Consider the following (incoherent) TBox $\mathcal{T}_1$, where A, B and C are primitive and $A_1, \dots, A_7$ defined concept names:

$ax_1: A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3$	$ax_2: A_2 \sqsubseteq A \sqcap A_4$
$ax_3: A_3 \sqsubseteq A_4 \sqcap A_5$	$ax_4: A_4 \sqsubseteq \forall s.B \sqcap C$
$ax_5: A_5 \sqsubseteq \exists s.\neg B$	$ax_6: A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$
$ax_7: A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$	

The set of unsatisfiable concept names as returned by a DL reasoner is $\{A_1, A_3, A_6, A_7\}$. Although this is still of manageable size, it hides crucial information, e.g., that unsatisfiability of A_1 depends on unsatisfiability of A_3 , which is incoherent because of the contradictions between A_4 and A_5 . We will use this example to explain our debugging methods.

Minimal unsatisfiability-preserving sub-TBoxes (MUPS)
Unsatisfiability-preserving sub-TBoxes of a TBox $\mathcal{T}$ and an unsatisfiable concept A are subsets of $\mathcal{T}$ in which A is unsatisfiable. In general there are several of these sub-TBoxes and we select the minimal ones, i.e., those containing only axioms that are necessary to preserve unsatisfiability.

Definition 3.1 Let A be a concept which is unsatisfiable in a TBox $\mathcal{T}$. A set $\mathcal{T}' \subseteq \mathcal{T}$ is a *minimal unsatisfiability-preserving sub-TBox (MUPS)* of $\mathcal{T}$ if A is unsatisfiable in $\mathcal{T}'$, and A is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$.

We will abbreviate the set of MUPS of $\mathcal{T}$ and A by $mups(\mathcal{T}, A)$. MUPS for our example TBox $\mathcal{T}_1$ and its unsatisfiable concepts are:

$$\begin{aligned}
mups(\mathcal{T}_1, A_1) &= \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\} \\
mups(\mathcal{T}_1, A_3) &= \{\{ax_3, ax_4, ax_5\}\} \\
mups(\mathcal{T}_1, A_6) &= \{\{ax_1, ax_2, ax_4, ax_6\}, \\
&\quad \{ax_1, ax_3, ax_4, ax_5, ax_6\}\} \\
mups(\mathcal{T}_1, A_7) &= \{\{ax_4, ax_7\}\}
\end{aligned}$$

MUPS are useful for relating unsatisfiability to sets of axioms but we will also use them in Section 4 to calculate MIPS.

Minimal incoherence-preserving sub-TBoxes (MIPS)

MIPS are the smallest subsets of an original TBox preserving unsatisfiability of at least one atomic concept.

Definition 3.2 Let $\mathcal{T}$ be an incoherent TBox. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a *minimal incoherence-preserving sub-TBox (MIPS)* of $\mathcal{T}$ if $\mathcal{T}'$ is incoherent, and every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$ is coherent.

We will abbreviate the set of MIPS of $\mathcal{T}$ by $mips(\mathcal{T})$. For $\mathcal{T}_1$ we get three MIPS:

$$mips(\mathcal{T}_1) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$$

It can easily be checked that each of the three incoherent TBoxes in $mips(\mathcal{T}_1)$ is indeed a MIPS as taking away a single axiom renders each of the three coherent. The first one signifies, for example, that the first two axioms are already contradictory without reference to any other axiom, which suggests a modelling error already in these two axioms.

Cores

Minimal incoherence-preserving sub-TBoxes identify smallest sets of TBox axioms causing the original TBox to be incoherent. In terminologies such as DICE, which are created through migration from other representation formalisms, there are several such sub-TBoxes, each corresponding to a particular contradictory terminology. *Cores* are now sets of axioms occurring in several of these incoherent TBoxes. The more MIPS such a core belongs to, the more likely its axioms will be the cause of contradictions.

Definition 3.3 Let $\mathcal{T}$ be a TBox. A non-empty intersection of n different MIPS in $mips(\mathcal{T})$ (with $n \geq 1$) is called a *MIPS-core of arity n* (or simply *n -ary core*) for $\mathcal{T}$.

Every set containing precisely one MIPS is, at least, a 1-ary core. The most interesting cores of a TBox, $\mathcal{T}$, are those with axioms that are present in as many MIPS of $\mathcal{T}$ as possible, i.e., having maximal arity. On the other hand, the size of a core is also significant, as a bigger size points to clusters of axioms causing contradictions in combination only.

In our running example, axiom ax_4 occurs both in $\{ax_3, ax_4, ax_5\}$ and $\{ax_4, ax_7\}$, which makes $\{ax_4\}$ a core of arity 2 for $\mathcal{T}_1$, which is the core of maximal arity in this example.

3.2 Concept Pinpointing

The next step in the debugging process is to simplify the definitions in order to obtain more concise descriptions of errors within an incoherent TBox.

Generalised Terminologies

Incoherence of a TBox can be regarded as an over-specification of one or more concepts in the relevant definitions. *Generalised terminologies* are terminologies where some of the definitions have been generalised.³ Furthermore, we require generalised definitions to be *syntactically related* to the original axioms. For different representation languages and types of knowledge bases we will have to formally specify what we consider to be syntactically related concepts. For the time being let us keep the definition abstract and assume that there is such a relation $rel(C, C')$ denoting that two concepts C and C' are related. Formally, a concept C' is then a *syntactic generalisation* of a concept C if $rel(C, C')$ and $C \sqsubseteq C'$ (independent of $\mathcal{T}$). Note that we have to take care of the case when C is unsatisfiable w.r.t. $\mathcal{T}$. Then C is equivalent to $\perp$, and everything is more general. Therefore we generalise C without referencing to the original terminology $\mathcal{T}$, i.e., we check for simple concept subsumption, only.

Now, generalised incoherence-preserving terminologies (GITs) are TBoxes where the defining concepts of the axioms are maximally generalised without losing incoherence.

Definition 3.4 Let $\mathcal{T} = \{C_1 \sqsubseteq D_1, \dots, C_n \sqsubseteq D_n\}$ be an incoherent TBox. An incoherent TBox $\mathcal{T}' = \{C_1 \sqsubseteq D'_1, \dots, C_n \sqsubseteq D'_n\}$ is a *generalised incoherence-preserving terminology (GIT)* of $\mathcal{T}$ if, and only if,

- each D'_i is a syntactic generalisation of D_i ($1 \leq i \leq n$),

³To simplify matters we generalise the right-hand side of axioms only as we are currently working with unfoldable TBoxes anyway.

- every TBox $\mathcal{T}' = \{C_1 \sqsubseteq D'_1, \dots, C_n \sqsubseteq D'_n\}$ with a syntactic generalisation D'_i of D_i where $D'_i \sqsubseteq D_i$ and $D'_i \not\sqsubseteq D_i$ (for some $1 \leq i \leq n$) is coherent.

We abbreviate the set of generalised incoherence-preserving terminologies of a TBox $\mathcal{T}$ by $git(\mathcal{T})$. Note that the set of GITs of a TBox $\mathcal{T}$ is equivalent to the union of GITs for the MIPS of $\mathcal{T}$. Take a simple syntactic relation relating a concept with the syntactic sub-concepts of its unfolded version with the same polarity and quantifier depth.⁴ As this particular definition of a syntactic relation depends on unfolding the related concepts, all but one axiom per GIT can be trivially generalised to $A_i \sqsubseteq \top$. Of all possible GITs we conjecture that the simplest ones, e.g., those with *syntactically minimal* generalisations, are most likely to be useful for the identification of errors. In our experiments we use two alternative formalisations, the first with respect to minimal size of axioms, the second with respect to the number of concept names occurring in the GIT. Three minimal sized GITs exist for our example TBox $\mathcal{T}_1$, where we only show the non-trivial axioms:

$$git(\mathcal{T}_1) = \{\{A_1 \sqsubseteq \neg A \sqcap A\}, \{A_3 \sqsubseteq \forall s. B \sqcap \exists s. \neg B\}, \{A_7 \sqsubseteq \forall s. B \sqcap \exists s. \neg B\}\}.$$

4 Debugging Unfoldable $\mathcal{ALC}$ -TBoxes

Practical experience has shown that applying our methods on a simplified version of DICE can already provide valuable debugging information. We will therefore only provide algorithms for unfoldable $\mathcal{ALC}$ -TBoxes [Nebel, 1990] as this significantly improves both the computational properties and the readability of the algorithm.

4.1 Algorithms for Axiom Pointing

The calculation of MIPS depends on the MIPS only, and we will provide an algorithm to calculate these minimal unsatisfiability-preserving sub-TBoxes based on Boolean minimisation of terminological axioms needed to close a standard tableau ([Baader *et al.*, 2003] Chapter 2).

Usually, unsatisfiability of a concept is detected with a fully saturated tableau (expanded with rules similar to those in Figure 2) where all branches contain a contradiction (or close, as we say). The information which axioms are relevant for the closure is contained in a simple label which is added to each formula in a branch. A *labelled formula* has the form $(a : C)^x$ where a is an individual name, C a concept and x a set of axioms, which we will refer to as *label*. A labelled branch is a set of labelled formulas and a tableau is a set of labelled branches. A formula can occur with different labels on the same branch. A branch is closed if it contains a clash, i.e. if there is at least one pair of formulas with contradictory atoms on the same individual. The notions of open branch and closed and open tableau are defined as usual and do not depend on the labels. We will always assume that any

⁴The polarity of a concept name A specifies whether A occurs within odd or even numbers of negations, the quantifier depth is the sequence of roles over which the concept is quantified. Such a relation will formally be defined for unfoldable $\mathcal{ALC}$ -TBoxes in Definition 4.3.

($\sqcap$):	if	$(a : C_1 \sqcap C_2)^{label} \in B$, but not both $(a : C_1)^{label} \in B$ and $(a : C_2)^{label} \in B$
	then	$B' := B \cup \{(a : C_1)^{label}, (a : C_2)^{label}\}$.
($\sqcup$):	if	$(a : C_1 \sqcup C_2)^{label} \in B$, but neither $(a : C_1)^{label} \in B$ nor $(a : C_2)^{label} \in B$
	then	$B' := B \cup \{(a : C_1)^{label}\}$ and $B'' := B \cup \{(a : C_2)^{label}\}$.
(Ax):	if	$(a : A)^{label} \in B$ and $(A \sqsubseteq C) \in \mathcal{T}$
	then	$B' := B \cup \{(a : C)^{label} \mid A \sqsubseteq C\}$.
($\exists$):	if	$(a : \exists R_i.C)^{label} \in B$, $R_i \in N_R$ and all other rules have been applied on all formulas over a , and if $\{(a : \forall R_i.C_1)^{label_1}, \dots, (a : \forall R_i.C_n)^{label_n}\} \subseteq B$ is the set of universal formulas for a w.r.t. R_i in B ,
	then	$B' := \{(b : C)^{label}, (b : C_1)^{label_1 \cup label}, \dots, (b : C_n)^{label_n \cup label}\}$ where b is a new individual name not occurring in B .

Figure 2: Tableau Rules for $\mathcal{ALC}$ -Satisfiability w.r.t. a TBox $\mathcal{T}$ (with Labels)

formula is in *negation normal form* (nnf) and newly created formulas are immediately transformed. We usually omit the prefix “labelled”.

To calculate a minimal unsatisfiability-preserving TBox for a concept name A w.r.t. an unfoldable TBox $\mathcal{T}$ we construct a tableau from a branch B initially containing only $(a : A)^\varnothing$ (for a new individual name a) by applying the rules in Figure 2 as long as possible. The rules are standard $\mathcal{ALC}$ -tableau rules with lazy unfolding, and have to be read as follows: assume that there is a tableau $T = \{B, B_1, \dots, B_n\}$ with $n+1$ branches. Application of one of the rules on B yields the tableau $T' := \{B', B_1, \dots, B_n\}$ for the ($\sqcap$), ($\exists$) and (Ax)-rule, $T'' := \{B', B'', B_1, \dots, B_n\}$ in case of the ($\sqcup$)-rule.

Once no more rules can be applied, we know which atoms are needed to close a saturated branch and can construct a minimisation function for A and $\mathcal{T}$ according to the rules in Figure 3. A propositional formula φ is called a *minimisation function for A and $\mathcal{T}$* if A is unsatisfiable in every subset of $\mathcal{T}$ containing the axioms which are true in an assignment making φ true. In our case axioms are used as propositional variables in φ . As we can identify unsatisfiability of A w.r.t. a set S of axioms with a closed tableau using only the axioms in S for unfolding, branching on a disjunctive rule implies that we need to join the functions of the appropriate sub-branches conjunctively. If an existential rule has been applied, the new branch B' might not necessarily be closed on formulas for both individuals. Assume that B' closes on the individual a but not on b . In this case $\min_function(a, B, \mathcal{T}) = \perp$, which means that the related disjunct does not influence the calculation of the minimal incoherent TBox.

Based on the minimisation function $\min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ (let us call it φ) which we calculated using the rules in Figure 3 we can now calculate the MUPS for A w.r.t. $\mathcal{T}$. The idea is to use prime implicants of φ . A prime implicant $ax_1 \wedge \dots \wedge ax_n$ is the smallest conjunction of literals⁵ implying φ [Quine, 1952]. As φ is a minimisation function every implicant of φ must be a minimisation function as well and therefore also the prime implicant. But this implies that the concept A must be unsatisfiable w.r.t. the set of axioms $\{ax_1, \dots, ax_n\}$. As $ax_1 \wedge \dots \wedge ax_n$ is the smallest implicant we also know that $\{ax_1, \dots, ax_n\}$ must be minimal, i.e. a MUPS. Theorem 4.1 captures this result formally.

⁵Note that in our case all literals are non-negated axioms.

Theorem 4.1 *Let A be a concept name, which is unsatisfiable w.r.t. an unfoldable $\mathcal{ALC}$ -TBox $\mathcal{T}$. The set of prime implicants of the minimisation function $\min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ is the set $mups(\mathcal{T}, A)$ of minimal unsatisfiability-preserving sub-TBoxes of A and $\mathcal{T}$.*

Proof: We first prove the claim that the propositional formula $\varphi := \min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ is indeed a minimisation function for the MUPS problem w.r.t. an unsatisfiable concept A and a TBox $\mathcal{T}$. We show that a tableau starting on a single branch $B := \{(a : A)^\varnothing\}$ closes on all branches by unfolding axioms only, that are evaluated as true in an assignment making φ true. This saturated tableau Tab^* is a particular sub-tableau of the original saturated tableau Tab which we used to calculate $\min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$, and it is this connection that we make use of to prove our first claim. Every branch in the new tableau is a subset of a branch occurring in the original one and we define *visible formulas* as those labelled formulas occurring in both tableaux. By induction over the rules applied to saturate Tab we can then show that each branch in the original tableau closes on at least one pair of visible formulas. If A is unsatisfiable w.r.t. $\mathcal{T}$, the tableau starting with the branch $\{(a : A)^\varnothing\}$ closes w.r.t. $\mathcal{T}$. As we have shown that this tableau closes w.r.t. $\mathcal{T}$ on visible formulas, it follows that Tab^* is closed on all branches, which proves the first claim. By another induction over the application of the rules in Figure 3 we can prove that φ is a *maximal* minimisation function, which means that $\psi \rightarrow \varphi$ for every minimisation function ψ . This proves the first part of the proof; the first claim (and the argument from above) implies that every implicant of a minimisation function identifies an unsatisfiability-preserving TBox, and maximality implies that prime implicants identify the minimal ones.

To show that the conjunction of every MUPS $\{ax_1, \dots, ax_n\}$ is a prime implicant of $\min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ is trivial as $ax_1 \wedge \dots \wedge ax_n$ is a minimisation function by definition. But as we know that $\min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ is maximal we know that $ax_1 \wedge \dots \wedge ax_n \rightarrow \min_function(a, \{(a : A)^\varnothing\}, \mathcal{T})$ which implies that $ax_1 \wedge \dots \wedge ax_n$ must be prime as otherwise $\{ax_1, \dots, ax_n\}$ would not be minimal. ■

Satisfiability in $\mathcal{ALC}$ is PSPACE-complete, and calculating MUPS does not increase the complexity as we can construct the minimisation function in a depth-first way, allowing us to keep only one single branch in memory at a time. However,

if rule = $(\sqcap)$ has been applied to $(a : C_1 \sqcap C_2)^{label}$ and B' is the new branch **return** $min_function(a, B', \mathcal{T})$;
if rule = $(\sqcup)$ has been applied to $(a : C_1 \sqcup C_2)^{label}$ and B' and B'' are the new branches
return $min_function(a, B', \mathcal{T}) \wedge min_function(a, B'', \mathcal{T})$;
if rule = $(\exists)$ has been applied to $(a : \exists R.C)^{label}$, B' is the new branch and b the new variable
return $min_function(a, B', \mathcal{T}) \vee min_function(b, B', \mathcal{T})$;
if rule = $(\forall x)$ has been applied and B' is the new branch **return** $min_function(a, B', \mathcal{T})$;
if no further rule can be applied **return**: $\bigvee_{(a : A)^x \in B, (a : \neg A)^y \in B} (\bigwedge_{ax \in x} ax \wedge \bigwedge_{ay \in y} ay)$;

Figure 3: $min_function(a, B, \mathcal{T})$: Minimisation-function for the MUPS-problem

we calculate prime implicants of a minimisation function the size of which can be exponential in the number of axioms in the TBox. Therefore, approximation methods have to be considered in practice avoiding the construction of fully saturated tableaux in order to reduce the size of the minimisation functions.

From MUPS we can easily calculate MIPS, but we need an additional operation on sets of TBoxes, called *subset-reduction*. Let $M = \{\mathcal{T}_1, \dots, \mathcal{T}_m\}$ be a set of TBoxes. The *subset-reduction* of M is the smallest subset $sr(M) \subseteq M$ such that for all $\mathcal{T} \in M$ there is a set $\mathcal{T}' \in sr(M)$ such that $\mathcal{T}' \subseteq \mathcal{T}$. A simple algorithm for the calculation of MIPS for $\mathcal{T}$ now simply follows from Theorem 4.2, which is a direct consequence of the definitions of MIPS and MUPS.

Theorem 4.2 *Let $\mathcal{T}$ be an incoherent TBox with unsatisfiable concepts $\Delta^{\mathcal{T}}$. Then, $mips(\mathcal{T}) = sr(\bigcup_{A \in \Delta^{\mathcal{T}}} mups(\mathcal{T}, A))$.*

Checking elements of $mips(\mathcal{T})$ for cores of maximal arity requires exponentially many checks in the size of $mips(\mathcal{T})$. In practice, we therefore apply a bottom-up method searching for maximal cores of increasing size stopping once the arity of the cores is smaller than 2.

4.2 Algorithms for Concept Pinpointing

Calculating GITs depends on the definition of $rel(C, C')$. In our case we will define this syntactic relation described in Footnote 4 including quantifier depth and polarity.

Definition 4.3 Let $\mathcal{T}$ be an unfoldable $\mathcal{ALC}$ -TBox. A concept C' is *syntactically related* to a concept C (notation: $rel(C, C')$) if $C' \subseteq qs_{\mathcal{T}}(C)$, where the set of *qualified sub-concepts*, $qs_{\mathcal{T}}(C)$, is defined inductively:

$$\begin{aligned}
 qs_{\mathcal{T}}(C \sqcap D) &= \{C', D', C' \sqcap D' \mid C' \in qs_{\mathcal{T}}(C), D' \in qs_{\mathcal{T}}(D)\} \\
 qs_{\mathcal{T}}(C \sqcup D) &= \{C', D', C' \sqcup D' \mid C' \in qs_{\mathcal{T}}(C), D' \in qs_{\mathcal{T}}(D)\} \\
 qs_{\mathcal{T}}(\exists r.C) &= qs_{\mathcal{T}}(\forall r.C) = \{\exists r.C', \forall r.C' \mid C' \in qs_{\mathcal{T}}(C)\} \\
 qs_{\mathcal{T}}(\neg C) &= \{\neg C' \mid C' \in qs_{\mathcal{T}}(C)\} \\
 qs_{\mathcal{T}}(A) &= \begin{cases} \{A\} \cup qs_{\mathcal{T}}(D) & \text{if } A \sqsubseteq D \in \mathcal{T} \\ \{A\} & \text{otherwise} \end{cases}
 \end{aligned}$$

Moreover, $qs_{\mathcal{T}}(C)$ also contains $\perp$ and $\top$ for all concepts C .

Note that we include terminological information from unfolded TBox axioms in the definition of syntactic relatedness. This choice allows us to create smaller and more concise GITs but is non-essential. As the TBox is finite and cycle-free, and as the concept size decreases in every recursion step, $qs_{\mathcal{T}}(C)$ is finite. Therefore there is only a finite set of concepts related to any axiom $A \sqsubseteq C$. A simple algorithm to calculate GITs for an incoherent unfoldable $\mathcal{ALC}$ -TBox $\mathcal{T}$ is therefore:

For all MIPS $M \in mips(\mathcal{T})$ (with $|mips(\mathcal{T})| = m$)

1. Let $\mathcal{T}'$ be the TBox where all axioms $A \sqsubseteq C \notin M$ in $\mathcal{T}$ have been replaced by $A \sqsubseteq \top$.
 2. Calculate different TBoxes $\mathcal{T}''$ where an arbitrary number of axioms $A_i \sqsubseteq C_i \in M$ have been replaced in $\mathcal{T}'$ by generalised axioms $A_i \sqsubseteq C'_i$, i.e., where $C'_i \in qs_{\mathcal{T}}(C_i)$ and $C_i \sqsubseteq C'_i$ for some $i \in \{1, \dots, m\}$.
 3. Return those incoherent TBoxes $\mathcal{T}''$ with minimal syntactic generalisations according to the subsumption relation, and the size or number of concept names.
-

The algorithm described above is a naive algorithm which might not terminate in reasonable time on large terminologies with complex definitions as the number of syntactically related concepts for a given concept C is exponential in the size of C . A more efficient algorithm to calculate GITs is based on the fact that the syntactically related concepts can be created ordered by size starting with atomic concepts. To find GITs of minimal size we therefore apply a bottom-up strategy checking more and more complex terminologies for coherence in Step 2 stopping once we find an incoherent one.

5 Evaluation

The algorithms for axiom pinpointing have been implemented in JAVA using RACER, which provides the set of unsatisfiable concepts. We evaluated the methods on both the anatomy fragment of DICE and the full DICE terminology. The fragment defines 529 concept names, 76 of which were unsatisfiable at first.⁶ There are 5 MIPS containing 2 axioms each, but no axiom occurs in all MIPS. However, there is a core of arity 3 and size 1: the axiom defining “central nervous system” as a “nervous system”. Note that the definition of “nervous system” is *not* contradictory but that its *use* is erroneous, e.g., the concept *brain* should be defined as a “part of” and not as a subconcept of the concept *nervous system*. For each of the 5 MIPS minimal size GITs have been calculated, and they point to the exact position of the logical incorrectness. For the MIPS related to the error described

⁶The high number of unsatisfiable concepts is due to the fact that the DL terminology for DICE has been created by migration from a frame-based terminological system. In order to make the semantics as explicit as possible a very restrictive translation has been chosen to highlight as many ambiguities as possible. Moreover, many concepts were defined as sub-concepts of unsatisfiable concepts. See [Cornet and Abu-Hanna, 2002] for details.

in the introduction the minimal size GIT is simply $Brain \sqsubseteq NervousSystem \sqcap BodyPart$ which, with the information that $BodyPart$ and $NervousSystem$ are disjoint concepts, identifies the erroneous specification.

The full DICE terminology defines more than 2400 concepts, of which more than 750 were unsatisfiable given the chosen migration method. Our implemented algorithms found the MUPS for all but 7 unsatisfiable concepts. In these seven cases the algorithm failed on the calculation of the prime implicants as we use a naive method in our current implementation. Based on the correctly calculated MUPS we approximated the set of MIPS which contained more than 350 TBoxes. We implemented an iterative approach, calculating the core Max of size 1 with maximal arity for $mips(\mathcal{T}_{dice})$, then the same for the remaining TBoxes in $mips(\mathcal{T}_{dice}) \setminus Max$ and so on. This way we identified 10 cores of size 1 with an arity of 25 to 10, covering almost half of the errors. For some of these cores we calculated GITs (by hand) which lead to the identification of a number of modelling errors. Even though the developed reasoning services are not yet fully integrated into the knowledge modelling environment, they are useful to pinpoint to the core of the logical contradiction, which is then investigated and eliminated by a domain expert.

Although the theoretical complexity of the MUPS problem is exponential in the size of the TBox the calculation of most MUPS for DICE was not problematic due to the relatively simple structure of the tableaux for the unsatisfiability proofs. Runtime analysis suggests that most CPU-time is spent by RACER identifying the unsatisfiable concepts, whereas computing MUPS usually takes less than 10 percent of the overall runtime⁷.

6 Conclusion & Further Work

Explanation has been a research topic right from the first days of research in Artificial Intelligence (e.g., introducing TMS [Doyle, 1979] or Diagnosis [Reiter, 1987]) and Theorem Proving (recently [Fiedler, 2001]). Despite a significant interest in explanation of DL reasoning recently shown in the DL community (as [DIG, 2002] suggests) relatively little work has been published on the subject. One exception is [McGuinness, 1996] where the author provides explanation for subsumption and non-subsumption. Her approach, based on *explanation as proof fragments*, uses structural subsumption for CLASSIC and has been extended to $\mathcal{ALC}$ -tableau reasoning in [Borgida et al., 1999]. In contrast to this approach, our non-standard reasoning services for axiom and concept pinpointing focus on the reduction of information, and are independent of particular calculi or implementations.

We have introduced non-standard reasoning services facilitating the debugging of logically incoherent DL terminologies and algorithms to calculate them. Whereas the ideas for axiom pinpointing are more evolved and efficiently implemented, only a relatively naive algorithm for concept pinpointing has been developed as yet. We are currently working on more efficient methods and alternative debugging services, and tests are under way on the DICE terminology.

⁷For the full DICE terminology the overall runtime to calculate MIPS is currently about 40 minutes on a PC with a 1GHz Pentium.

As our methods are developed for the particular application of the DICE terminology some restrictions apply, most importantly to use $\mathcal{ALC}$ and to consider unfoldable TBoxes, only. Neither is essential, and we conjecture that it is not too hard to extend our algorithms to find MIPS and GITs both for more expressive languages and for general TBoxes. Both issues will be addressed in future investigations. Further work must also extend the application domain to other terminologies as we believe that the success we have with debugging DICE is not tied to the structure of this particular representation. Of particular interest should be applications related to the Semantic Web effort as our methods are particularly geared to support the debugging of existing terminologies and logical contradictions caused by migration or merging of terminologies.

References

- [Baader et al., 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
- [Borgida et al., 1999] A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Explaining $\mathcal{ALC}$ subsumption. In *DL-99*, pages 37–40, 1999.
- [Cornet and Abu-Hanna, 2002] R. Cornet and A. Abu-Hanna. Evaluation of a frame-based ontology. A formalization-oriented approach. In *Proceedings of MIE2002, Studies in Health Technology & Information*, volume 90, pages 488–93, 2002.
- [DIG, 2002] Minutes of the DL Implementation Group-Workshop, 2002. <http://dl.kr.org/dig/minutes-012002.html>, visited on January 9, 2003.
- [Doyle, 1979] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12(3):231–272, 1979.
- [Fiedler, 2001] A. Fiedler. *P.r*ex: An interactive proof explainer. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, number 2083 in LNAI, pages 416–420, 2001.
- [Haarslev and Möller, 2001] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, number 2083 in LNAI, 2001.
- [Horrocks, 1998] I. Horrocks. The FaCT system. In H. de Swart, editor, *Tableaux’98*, number 1397 in LNAI, pages 307–312, 1998.
- [Küstters, 2001] R. Küstters. *Non-Standard Inferences in Description Logics*, volume 2100 of LNAI. 2001.
- [McGuinness, 1996] Deborah McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Department of Computer Science, Rutgers University, 1996.
- [Nebel, 1990] B. Nebel. Terminological reasoning is inherently intractable. *AI*, 43:235–249, 1990.
- [Quine, 1952] W.V. Quine. The problem of simplifying truth functions. *American Math. Monthly*, 59:521–531, 1952.
- [Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

DIAGNOSIS

Formal Verification of Diagnosability via Symbolic Model Checking

Alessandro Cimatti

ITC-irst
Povo, Trento, Italy
cimatti@irst.itc.it

Charles Pecheur

RIACS/NASA Ames Research Center
Moffett Field, CA, U.S.A.
pecheur@ptolemy.arc.nasa.gov

Roberto Cavada

ITC-irst
Povo, Trento, Italy
cavada@irst.itc.it

Abstract

This paper addresses the formal verification of diagnosis systems. We tackle the problem of diagnosability: given a partially observable dynamic system, and a diagnosis system observing its evolution over time, we discuss how to verify (at design time) if the diagnosis system will be able to infer (at run-time) the required information on the hidden part of the dynamic state. We tackle the problem by looking for pairs of scenarios that are observationally indistinguishable, but lead to situations that are required to be distinguished. We reduce the problem to a model checking problem. The finite state machine modeling the dynamic system is replicated to construct such pairs of scenarios; the diagnosability conditions are formally expressed in temporal logic; the check for diagnosability is carried out by solving a model checking problem. We focus on the practical applicability of the method. We show how the formalism is adequate to represent diagnosability problems arising from a significant, real-world application. Symbolic model checking techniques are used to formally verify and incrementally refine the diagnosability conditions.

1 Introduction

Diagnosis is the process of inferring the (most plausible) causes for the behavior of a given system, given a set of observations. In many control applications, ranging from industrial plants (e.g. production, power) to transportation (e.g. railways, avionics, space), diagnosis needs to be carried out *on-line*, in parallel with the control process. This is needed to identify whether the controlled system is working correctly or not, and to provide the controller with information on the degraded conditions (e.g. what are the malfunctioning devices), so that the appropriate counter-measures can be taken. The ability to validate such diagnosis systems becomes very important, in particular in the case of applications operating in hazardous or inaccessible conditions and carrying out vital functions.

In this paper, we focus on the key issue of *diagnosability*, i.e. the possibility for an ideal diagnosis system to infer accurate and sufficient run-time information on the behavior of the

observed system. We propose a new, practical approach to the *verification* of diagnosability, making the following contributions. First, we provide a formal characterization of diagnosability problem, using the idea of *context*, that explicitly takes into account the run-time conditions under which it should be possible to acquire certain information.

Second, we show that a diagnosability condition for a given plant is violated if and only if a *critical pair* can be found. A critical pair is a pair of executions that are indistinguishable (i.e. share the same inputs and outputs), but hide conditions that should be distinguished (for instance, to prevent simple failures to stay undetected and degenerate into catastrophic events). We define the *coupled twin model* of the plant, and show that it can be used to search for critical pairs.

Third, we recast the problem in the framework of *Model Checking* [Clarke *et al.*, 1999], a verification technique that is gaining increasing interest also in AI. With model checking, it is possible to exhaustively analyze the (possibly infinite) behaviors of (large sized) finite state machines, and to check if requirements expressed in terms of Temporal Logics [Emerson, 1990] are met. We show how to represent a diagnosability problem in terms of temporal logic formulae, and how to reduce it to a model checking problem over the coupled twin model.

Finally, we demonstrate the practical applicability within the Livingstone framework, a model-based diagnosis system developed at NASA Ames Research Center [Williams and Nayak, 1996]. We developed a platform able to generate formal models for the twin plant, starting from Livingstone models. Several diagnosability problems corresponding to interesting scenarios from real-world applications were tackled by means of the NuSMV model checker [Cimatti *et al.*, 2002]. An experimental analysis shows that the verification of diagnosability can be practical: large Livingstone models of space transportation systems are automatically analyzed within seconds by means of SAT-based symbolic model checking techniques.

The paper is organized as follows. In Section 2, we state our working hypotheses. In Section 3 we formalize the problem, while in Section 4 we characterize our approach. In Section 5, we discuss the application of model checking techniques. In Section 6, we describe the experimental framework. Finally, Section 7 reviews similar work, and Section 8 draws some conclusions and outlines future lines of activity.

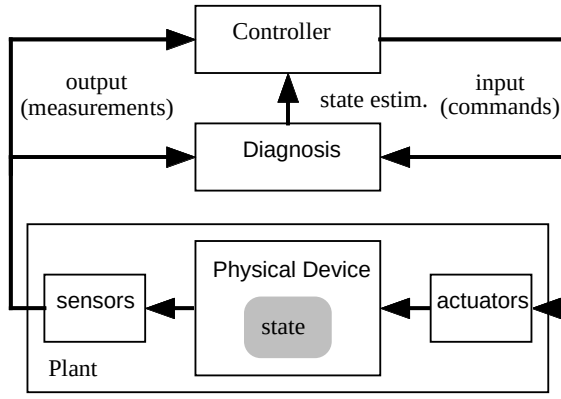


Figure 1: Architecture of a diagnosis system

2 Working Hypotheses

We consider a *diagnosis system* connected to a feedback control loop between a *plant* and its *controller* (Figure 1). The inputs of the plant are the *commands* issued by the controller; its outputs are *measurements* returned back to the controller. The role of the diagnosis system is to observe the inputs and the outputs to the plant, and to report a *state estimation*, tracking the evolution of the unobservable state of the plant. This task is substantially more complex than diagnosis for a combinatorial, memory-less system. An estimation consists of a set of possible states of the plant, in the following referred to as a *belief state*. Although diagnosis might rank these states based on likelihood, as a first approach we ignore that aspect. We also disregard issues such as the correspondence between the model of the plant and the plant itself. We will focus on plants modeled as finite-state discrete systems. We assume that we have a model of the plant as a partially observable transition system, according to the following definition.

Definition 1 A (partially observable) plant is a structure $\langle X, U, Y, \delta, \lambda \rangle$, where X, U, Y are finite sets, respectively called the state space, input space and output space, $\delta \subset X \times U \times X$ is the transition relation, and $\lambda \subset X \times Y$ is the observation relation.

We assume that a plant $P = \langle X, U, Y, \delta, \lambda \rangle$ is given. We use $x, x_0, x_1, \dots$ to denote states of P , $u, u_0, u_1, \dots$ to denote inputs of P , $y, y_0, y_1, \dots$ to denote outputs of P . We write $x \xrightarrow{u} x'$ for $(x, u, x') \in \delta$, and x/y for $(x, y) \in \lambda$. The state is the “hidden” part of the plant: only the sequences of inputs and outputs are observable. P covers all types of behaviors that diagnosis is expected to handle — including faulty behaviors, with X containing faulty states. In general, P need not be deterministic. Thus, the state after a transition may not be uniquely determined the state before the transition and by the input. Observability is modeled by associating to each state a (non empty) set of possible outputs. It is therefore possible to model different forms of observation, e.g. when the information conveyed by the sensors is also uncertain. In this paper, we present the (input, state and output) spaces of a plant by means of assignment to (input, state and output) variables ranging over finite sets of values; the transition and observation relation can be compactly presented with boolean

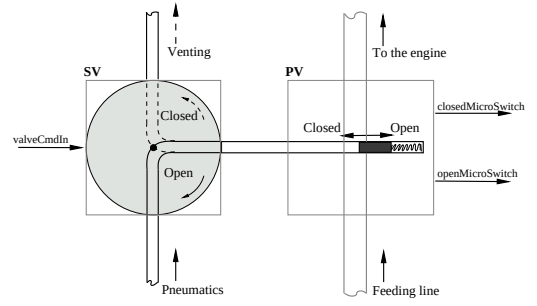


Figure 2: A simple example

formulae. (Notice however that the content of the paper is independent of the specific description used to present P .)

Definition 2 A feasible execution of $k \geq 0$ steps in P is a sequence $\sigma = x_0, y_0, u_1, x_1, y_1, u_2, \dots, u_k, x_k, y_k$, such that $x_{i-1} \xrightarrow{u_i} x_i$ for $1 \leq i \leq k$, and x_i/y_i for $0 \leq i \leq k$. We define Σ_P as the set of all feasible executions of P . The observable trace of a feasible execution σ is $w = y_0, u_1, y_1, \dots, u_k, y_k$. We write $\sigma : x_0 \xrightarrow{w} x_k$, if such a σ exists.

The above definition defines the dynamics of a plant and its observable counterpart. Notice that if an execution σ has k steps, then the corresponding trace $w \in Y \times (U \times Y)^k$. The set of traces is in general a subset of $Y \times (U \times Y)^*$. In the following we use σ to denote a feasible execution, and w to denote the corresponding (observable) trace. For explanatory purposes, we consider the plant outlined in Figure 2, that is a simplified segment of the real-world application described in Section 6. A pneumatic valve (PV) controls the flow of a feeding line to the engine, and it is closed in its resting position (determined by the action of a spring). A solenoid valve (SV), when open, can open PV by directing the pneumatic flow against the spring. When SV is closed, the action of the spring is no longer contrasted and PV can close. SV can receive an input (open/close), but has no observables. PV has two sensors, each providing information on a position (e.g. whether it is open or not). Both valves can become stuck in their current position. The position of the valves influences the pneumatic and engine flows. Each of the valves has two unobservable variables: *state*, with values *open* and *closed*, and *failure_mode*, with values *stuck* and *ok*. The input variable can take the values *no-cmd*, *open*, and *close*. The effect of a command depends on the current status. Failures can occur at any time, and are permanent. Sensors of PV are also associated with a *failure_mode* variable, that determines what information is conveyed.

3 Diagnosability

Run-time diagnosis starts from a (possibly partial) initial knowledge, observes the sequence of inputs and outputs, and tries to update a belief state that is an estimate of the possible states of the plant. For instance, if SV is commanded open, and the PV is not sensed open, then some problem should be diagnosed. It could be that SV or PV are stuck, or even that

both PV sensors are not working correctly. Ideally, a diagnosis function should return belief states that are as specific (i.e. as small) as possible, but include the actual state of the plant.

Definition 3 A diagnosis function for P is a function $\Delta : 2^X \times Y \times (U \times Y)^* \rightarrow 2^X$. Let $\hat{x}_0 \subset 2^X$. A diagnosis value $\hat{x} = \Delta(\hat{x}_0, w)$ is correct w.r.t. $\hat{x}_0$ and w , if and only if, for any $x_0 \in \hat{x}_0$ and x such that $x_0 \xrightarrow{w} x$, we have $x \in \hat{x}$. Δ is correct if and only if $\Delta(\hat{x}_0, w)$ is correct for any $\hat{x}_0$ and w .

The intuition behind correct diagnosis values is that they encompass all potential current states, so that they cannot miss the actual state. In the following, we restrict ourselves to correct diagnosis functions.

Diagnosability aims at detecting if (parts of) the hidden state can be accurately tracked by looking at the observable traces. We use diagnosis conditions to specify which information on the state we are interested in distinguishing.

Definition 4 A diagnosis condition for a plant P is a pair of nonempty sets of states $c_1, c_2 \subset X$, written $c_1 \perp c_2$.

We can express *fault detection*, i.e. telling if any fault is present ($\text{fault} \perp \neg\text{fault}$), or *fault separation*, i.e. distinguishing between different faults (or fault classes) ($\text{fault}_a \perp \text{fault}_b$). Intuitively, a diagnosis value is not satisfactory if it intersects with both sides of the condition. In the example, a fault separation condition is not satisfied if we have a belief state containing both a state where PV is faulty, and one where SV is faulty.

However, it would be unrealistic to require that diagnosis provide correct and exact state estimations, instantaneously and under all circumstances. For instance, a stuck PV will stay unnoticed at least as long as SV is not operated. Rather, we require a diagnosis system to be able to decide between alternative conditions on the state of the system in the context where the distinction becomes critical.

Definition 5 A diagnosis context for P is a structure $C = \langle \theta, \Sigma_{12} \rangle$, where θ is an equivalence relation on X , and $\Sigma_{12} \subset \Sigma_P \times \Sigma_P$. $\hat{x}_0$ satisfies θ ($\hat{x}_0 \models \theta$), iff $\hat{x}_0 \times \hat{x}_0 \subset \theta$. $(\hat{x}_0, w) \models C$ iff $\hat{x}_0 \models \theta$, and there exist $(\sigma_1, \sigma_2) \in \Sigma_{12}$, $x_{01}, x_{02} \in \hat{x}_0$, such that $\sigma_1 : x_{01} \xrightarrow{w} x_1$ and $\sigma_2 : x_{02} \xrightarrow{w} x_2$.

Intuitively, θ defines the initial conditions under which diagnosability is to be investigated, by inducing of a set of disjoint belief states. Basically, the initial belief state $\hat{x}_0$ must fall within one of the belief states induced by θ (the condition on θ can be stated as $(x_{01}, x_{02}) \in \theta$ for all $x_{01}, x_{02} \in \hat{x}_0$). For example, θ can partition states according to the positions of PV and SV, to capture the assumption that we initially (only) know those positions. Σ_{12} characterizes pairs of relevant executions. For instance, we may want to express the fact that the controller commands SV to be open and closed at least one time. In the case of fault detection, we might want to state that the elements of Σ_{12} are pairs of traces, there the first are without faults, while the second ones have exactly one fault. Notice that expressing Σ_{12} as $\Sigma_1 \times \Sigma_2$ would be inadequate: for instance, in fault separation it would be impossible to constrain the two runs to have different failures. In the following we assume that a context $C = \langle \theta, \Sigma_{12} \rangle$ is given. The notion of diagnosability is precisely characterized as follows.

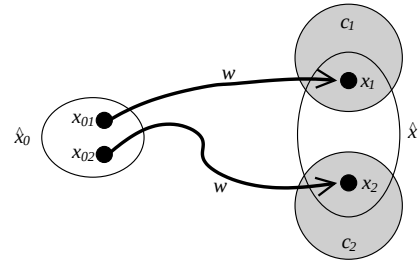


Figure 3: Critical pair

Definition 6 A diagnosis value $\hat{x}$ satisfies $c_1 \perp c_2$, written $\hat{x} \models c_1 \perp c_2$, if and only if either $\hat{x} \cap c_1 = \emptyset$ or $\hat{x} \cap c_2 = \emptyset$. A diagnosis function Δ satisfies $c_1 \perp c_2$ over C , written $\Delta, C \models c_1 \perp c_2$, if and only if, for all $(\hat{x}_0, w) \models C$, we have $\Delta(\hat{x}_0, w) \models c_1 \perp c_2$. A condition $c_1 \perp c_2$ is diagnosable in P over C if and only if there exists a correct diagnosis function that satisfies it.

4 Diagnosability as Coupled Reachability

Our approach to reasoning about diagnosability is based on refutation, i.e. we search for ways in which the diagnosability property can be violated. Intuitively, we verify the diagnosability of $c_1 \perp c_2$ by checking that P does not have a *critical pair*, i.e. two executions with identical observable traces, one leading to c_1 , one leading to c_2 (see Figure 3).

Definition 7 A critical pair of a plant P , with trace w , for a diagnosis condition $c_1 \perp c_2$, is a pair of feasible executions $\sigma_1 : x_{01} \xrightarrow{w} x_1$ and $\sigma_2 : x_{02} \xrightarrow{w} x_2$, written $\sigma_1 | \sigma_2 : x_{01} | x_{02} \xrightarrow{w} x_1 | x_2$, such that $x_1 \in c_1$ and $x_2 \in c_2$. The critical pair $\sigma_1 | \sigma_2$ is covered by a diagnosis context $C = \langle \theta, \Sigma_{12} \rangle$, iff $(x_{01}, x_{02}) \in \theta$, $(\sigma_1, \sigma_2) \in \Sigma_{12}$.

The absence of critical pairs for $c_1 \perp c_2$ in a given context is a necessary and sufficient condition for $c_1 \perp c_2$ to be diagnosable in that context.

Theorem 8 $c_1 \perp c_2$ is diagnosable over C in P , if and only if P has no critical pair for $c_1 \perp c_2$ in C .

To prove the theorem, we introduce the notion of *perfect diagnosis*, written $\overline{\Delta}$, i.e. the most specific correct diagnosis function that can be made assuming full knowledge of P .

Definition 9 The perfect diagnosis for a plant P is the diagnosis function $\overline{\Delta}(\hat{x}_0, w) = \{x \mid \exists x_0 \in \hat{x}_0. x_0 \xrightarrow{w} x\}$.

Given an initial belief state $\hat{x}_0$ and trace w , $\overline{\Delta}(\hat{x}_0, w)$ returns exactly all states that can be reached from $\hat{x}_0$ through w . It is easy to see that $\overline{\Delta}$ is correct, and that it is the most informative diagnosis function, i.e. $\overline{\Delta}(\hat{x}_0, w) \subset \Delta(\hat{x}_0, w)$ for any correct Δ . We observe that a condition $c_1 \perp c_2$ is diagnosable in a plant P over a context C if and only if $\overline{\Delta}, C \models c_1 \perp c_2$: from Definition 6, if $\hat{x} \subset \hat{x}'$ and $\hat{x}' \models c_1 \perp c_2$, then $\hat{x} \models c_1 \perp c_2$. Therefore, $\overline{\Delta}$ satisfies any diagnosable condition, since its diagnosis values are more specific than any other correct diagnosis function.

Proof of Theorem 8 $c_1 \perp c_2$ is not diagnosable in P over C ; if and only if there exist $\hat{x}_0$ and w such that $\overline{\Delta}(\hat{x}_0, w), C \not\models$

$c_1 \perp c_2$; if and only if there exist $(\sigma_1, \sigma_2) \in \Sigma_{12}$ and $x_{01}, x_{02} \in \hat{x}_0$ such that $\sigma_1 : x_{01} \xrightarrow{w} x_1$ and $\sigma_2 : x_{02} \xrightarrow{w} x_2$; if and only if there exists a critical pair for $\sigma_1 | \sigma_2$ for $c_1 \perp c_2$ in C . Q.E.D.

In order to search for critical pairs in P , we build the *coupled twin plant* for P , written $P \cdot P$, i.e. a “Siamese twins” plant, made out of two copies of P , whose inputs and outputs are forced to be identical.

Definition 10 *The coupled twin plant of P is the plant $P \cdot P = \langle X \cdot X, U, Y, \delta \cdot \delta, \lambda \cdot \lambda \rangle$, where $X \cdot X \subset X^2$, and, for all $x_1, x_2 \in X$, $(x_1, x_2) \in X \cdot X$ iff there exists $y \in Y$ such that $(x_1, y) \in \lambda$ and $(x_2, y) \in \lambda$; $((x_1, x_2), u, (x'_1, x'_2)) \in \delta \cdot \delta$ iff $(x_1, u, x'_1) \in \delta$ and $(x_2, u, x'_2) \in \delta$; $((x_1, x_2), y) \in \lambda \cdot \lambda$ iff $(x_1, y) \in \lambda$ and $(x_2, y) \in \lambda$.*

In the following, we assume that the coupled twin plant $P \cdot P$ of P is given. The importance of the coupled twin plant construction is shown by the following theorem.

Theorem 11 $\sigma_1 : x_{01} \xrightarrow{w} x_1$ and $\sigma_2 : x_{02} \xrightarrow{w} x_2$ are two feasible executions in P if and only if $\sigma_{12} : (x_{01}, x_{02}) \xrightarrow{w} (x_1, x_2)$ is a feasible execution in $P \cdot P$.

This is easily demonstrated by induction on the length k of executions. When $k = 0$, then $w = y_0$. From Definition 10, x_{01}/y_0 and x_{02}/y_0 iff $(x_{01}, x_{02}) \in X \cdot X$ and $((x_{01}, x_{02}), y_0) \in \lambda \cdot \lambda$. The step case is proved by extending with a transition the executions of length k , for which the theorem holds. From Definition 10, the transitions in $P \cdot P$ mimic the transitions in P . For the last observation, we reason as for the base case.

5 Diagnosability via Model Checking

We work in the framework of Model Checking [Clarke *et al.*, 1999], a formal verification technique that is increasingly applied to the design of industrial digital systems (e.g. communication protocols, hardware design). Model checking allows to verify if the (possibly infinite) behaviors of a system satisfy a given property. The system being analyzed is presented as a Kripke structure, while properties are expressed in a temporal logic [Emerson, 1990]. Model checking algorithms are based on the exhaustive exploration of the Kripke structure. The technique is completely automatic and, if a property is violated, it can produce a counterexample, i.e. a simulation witnessing the violation. Symbolic model checking [McMillan, 1993] is based on the manipulation of sets of states described by boolean formulae; efficient engines, such as Binary Decision Diagrams (BDDs) [Bryant, 1986] and SAT solvers [Moskewicz *et al.*, 2001], provide a basis for efficient representation and exploration of the search space.

Our approach to diagnosability inherits several elements from model checking. The first is that a plant P can be associated with a Kripke structure K_P representing its behavior. This makes it possible for us to directly analyze a plant with model checking techniques. A Kripke structure is basically a nondeterministic transition system, where transitions are not labeled, while information is associated with states (i.e. each state is associated with a valuation to the state variables of the structure). The mapping from plant to Kripke structure

is rather simple, and is based on the idea that the state, input and output spaces of the plant can be encoded into the state space of the Kripke structure. The information on the inputs, labeling the transitions in P , simply becomes part of the state of K_P , and is to be interpreted as “the input that will be processed in the next transition”. More formally, each state s in K_P is associated with a valuation that characterizes a triple $x/y \xrightarrow{u}$, i.e. P is in state x , output y is observed and input u is received. The dynamics of P directly induce the dynamics of K_P . For each feasible execution σ in P of the form $x_0, y_0, u_1, x_1, y_1, u_2, \dots, u_k, x_k, y_k$ there is a corresponding path π in K_P , i.e. a sequence of states $s_0, s_1, \dots, s_k$ where each s_i is associated with the triple $x_i/y_i \xrightarrow{u_{i+1}}$. The same mapping lifts to the coupled twin plant construction. (In the above description, we omit a few technical details, having to do in particular with the fact that a Kripke structure is assumed to be total, i.e. every state has at least one successor. In this way, it is possible to assume that the analyzed paths are infinite. Since inputs become part of the state, it is possible that the transition relation is no longer total. There are standard workarounds to this problem, routinely used in verification and in AI planning. See for instance [Cimatti and Roveri, 2000].) In the following, we assume that Kripke structure $K_{P \cdot P}$ corresponding to the coupled twin plant $P \cdot P$ is given.

The second element from model checking is the *symbolic representation*. The state of $K_{P \cdot P}$ is defined with a vector of variables $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}, \mathbf{y})$, respectively ranging over X, X, U and Y . We can use formulae to characterize sets of states. In $K_{P \cdot P}$ we express atomic propositions over such variables. We can have equalities between variables and the corresponding values: $P1.PV.input = close$ denotes the set of states where the first instance of PV receives the `close` command. Variables of the same type can be equated: $P1.SV.failure_mode = P2.SV.failure_mode$ describes the set of states where the two instances of SV in the twin plant have the same failure mode. Any subset c of $X \times X \times U \times Y$ can be described with a formula $c(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{u})$. For instance, the formula $c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2)$ expresses a state of the twin plant where the first instance is in c_1 and the second is in c_2 . Similarly, $\theta(\mathbf{x}_1, \mathbf{x}_2)$ is a (propositional) formula expressing, in symbolic form, the equivalence relation θ .

The third ingredient from model checking is the use of *temporal logics* to characterize behaviors of the system over time. We use LTL (Linear Temporal Logic [Emerson, 1990]), where formulae are constructed starting by combining atomic propositions with boolean operators and *temporal modalities*. If ϕ and ψ are LTL formulae, so are $F\phi$ (sometimes in the future ϕ), $G\phi$ (always in the future ϕ), $\psi U \phi$ (sometimes in the future ϕ and, until then, ψ), $X\phi$ (in the next time step ϕ). An LTL formula ϕ holds on a path π (written $\pi \models \phi$), iff ϕ is true in π at (step) 0 (written $\pi^0 \models \phi$). If p is an atomic proposition, then $\pi^i \models p$ iff p is true according to the assignment associated with the i -th state of π . Boolean connectives have the standard interpretation (e.g. $\pi^i \models \phi \wedge \psi$ iff $\pi^i \models \phi$ and $\pi^i \models \psi$). The interesting cases of temporal operators are as follows. $\pi^i \models F\phi$ iff there exists $j \geq i$ such that $\pi^j \models \phi$. $\pi^i \models G\phi$ iff for all $j \geq i$ $\pi^j \models \phi$. $\pi^i \models \psi U \phi$ iff there exists

```

MODULE SV_type (cmd_in)
...
MODULE PV_type ( )
VAR pneumaticsLineIn : {abvThresh,blwThresh};
VAR valvePositionValue : {open,closed};
VAR mode : {stuckOpen,nominal,stuckClosed};
...
DEFINE _faultmodes := {stuckOpen,stuckClosed};
DEFINE _broken := mode in _faultmodes;
DEFINE _brokencount := _broken;
DEFINE _brokenprob :=
  case
    mode = stuckOpen : 3;
    mode = stuckClosed : 3;
    else : 0;
  esac;
MODULE EX_type (sv_cmd_in, ...)
SV : SV_type(sv_cmd_in);
PV : PV_type( );
...
DEFINE _brokencount := SV._broken +
  PV._broken +
  ...

```

Figure 4: The SMV model for the example

$j \geq i$ such that $\pi^j \models \phi$ and, for all $i \leq h < j$, $\pi^h \models \psi$. $\pi^i \models X\phi$ iff $\pi^{i+1} \models \phi$. An (existential) model checking problem $K \models \phi$, read ϕ (existentially) holds in K , amounts to detecting if K admits a path where ϕ holds.

A diagnosis condition with a context is represented with temporal logic formula ϕ such that $K_{P.P} \models \phi$ holds iff there is a critical pair. We express reachability in $P \cdot P$ of a critical pair for a diagnosis condition $c_1 \perp c_2$ with the formula: $F(c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2))$. Model checking can be run on the problem $K_{P.P} \models F(c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2))$. If the answer is true, then we have a witness for the a critical pair. Given the context $C = \langle \theta, \Sigma_{12} \rangle$, we enforce the initial condition θ with the formula $\theta(\mathbf{x}_1, \mathbf{x}_2) \wedge F(c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2))$. The conjunction has the effect of restricting the analysis to the paths in $K_{P.P}$ starting from the states that satisfy θ . Σ_{12} is taken into account assuming that a characterization of the sets of traces is described by an LTL formula $\Sigma_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{u})$. The model checking problem corresponding to diagnosability is $K_{P.P} \models \theta(\mathbf{x}_1, \mathbf{x}_2) \wedge \Sigma_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{u}) \wedge F(c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2))$. In practice, several simplifications are possible. Often $\Sigma_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}, \mathbf{y})$ can be expressed in terms of propositional constraints $\phi_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}, \mathbf{y})$, that must hold on all the states of the execution; in such cases diagnosability in context C is basically represented by the LTL formula $\theta(\mathbf{x}_1, \mathbf{x}_2) \wedge (\phi_{12}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}, \mathbf{y}) \cup (c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2)))$. Notice that the formula holds if the path condition ϕ_{12} holds until the conditions c_1 and c_2 are reached. When the context does not constraint the executions, the above formula is equivalent to $\theta(\mathbf{x}_1, \mathbf{x}_2) \wedge F(c_1(\mathbf{x}_1) \wedge c_2(\mathbf{x}_2))$.

```

MODULE main
VAR
  sv_cmd : { no_cmd, open, close };
  P1 : EX_type(sv_cmd, ...);
  P2 : EX_type(sv_cmd, ...);
INVAR
  (P1.PV.sense_open = P2.PV.sense_open) & ...

```

Figure 5: The SMV schema for the twin plant

6 Experimental Evaluation

We analyzed the practical applicability of our approach within the Livingstone framework. Livingstone is a model-based health monitoring system developed at NASA Ames [Williams and Nayak, 1996]. It uses a model of a physical system, such as a spacecraft, to infer its state and diagnose faults from observations. Livingstone is one of the three parts of the Remote Agent (RA), an autonomous spacecraft controller developed by NASA Ames Research Center jointly with the Jet Propulsion Laboratory. RA was the first AI software to control an operational spacecraft [Muscettola *et al.*, 1998]. Livingstone has also been used in other applications such as the control of the In-Situ Propellant Production system (ISPP) for Mars missions [Clancy *et al.*, 1999], the monitoring of a mobile robot [Simmons *et al.*, 2001], and Intelligent Vehicle Health Management (IVHM) for experimental space transportation vehicles [Bajwa and Sweet, 2002].

Livingstone uses a qualitative relational model describing the evolution of observable and hidden variables. Continuous physical domains are abstracted into discrete intervals such as {low, nominal, high}. Each component has a mode variable identifying its nominal and fault modes. Livingstone models are specified in a hierarchical, declarative formalism called JMPL, or using a graphical development environment. Livingstone observes the commands issued to the plant and uses the model to predict the plant state. It then compares the predicted state against observations received from the actual sensors. If a discrepancy is found, Livingstone performs a diagnosis by searching for the most likely configuration of component modes that are consistent with the observations.

Livingstone models directly induce a synchronous transitions systems, very similar to a plant model. Pecheur and Simmons [Pecheur and Simmons, 2000] have developed a translator able to convert a Livingstone model and a related set of specifications in the language of the SMV model checker [McMillan, 1993], and to convert back the diagnostic traces in terms of the Livingstone model. Figure 4 outlines the structure of the example plant in SMV language. For each component type, there is a corresponding module. The first module statements define the model for the SV and the PV. For each of the components, a set of variables is defined, the dynamics of which is directly induced from the Livingstone model. Notice the `_brokenprob` variable, whose numerical value is the (negated) logarithm of the probability of failure for the component (e.g. 10^{-3}). This enables for a (rough) analysis of failure probabilities. These modules are instantiated in the EX module, with a parameter representing the commands to PV.

In order to tackle diagnosability, we devised a systematic way of constructing the SMV coupled twin plant of a Livingstone model. The construction is outlined in Figure 5. The EX plant is instantiated twice in the main, at top level, generating P1 and P2. The same input variable (`sv_cmd`) is given in input to both instances. Then, the outputs of the two instances of the plant are constrained to exhibit the same behavior by means of an `INVAR` statement, i.e. a condition that must hold in all states. The SMV language enables for the specification of context. The basic building blocks of the properties are propositional conditions over states, that can be expressed by means of the `DEFINE` construct. For instance, it is possible to express conditions on the number of failures, using the `_brokencount` variable defined in Figure 4. For instance, `P1._brokencount < 2` in the main module states that at most one failure can occur in the first instance of the circuit.

In the experimental analysis, we tackled several problems for the Livingstone model of the Main Propulsion System for the X-34, a next-generation vehicle for the Space Shuttle [Bajwa and Sweet, 2002]. We interacted with the NASA experts of diagnosis, to check the representational adequacy of our formalism, and to characterize diagnosability problems of practical relevance. The problems were defined starting from simulation runs, that tested a specific fault in a specific context. We remark the impact of our approach is far beyond the one of testing, since it performs an exhaustive analysis (though at a higher abstraction level) within the cases captured by the context. The experimental evaluation was carried out by running different symbolic model checking tools on models described in SMV language, such as the CMU SMV [McMillan, 1993], Bwolen Yang's version of SMV [Yang *et al.*, 1999], and NuSMV [Cimatti *et al.*, 2002]. While the former ones are based on Binary Decision Diagrams (BDDs) [Bryant, 1986], NuSMV also enables the use of SAT-based techniques [Moskewicz *et al.*, 2001]. This activity suggested several considerations. First, the experimental analysis was fundamental to tune the formalism. Several improvements (e.g. the notion of context) were conceived while trying to encompass representational issues arising in practice. Second, contexts are incrementally characterized. Although the requirements for the diagnosis system can usually suggest an initial version for the context, it is seldom the case that the precise conditions for diagnosability are known (or can be precisely stated) in advance. The ability of the model checker to find critical pairs was very useful in the refinement, since it helped to explain why diagnosability fails, i.e. to understand whether the context is not strict enough, or because a problem was found.

In terms of performance, the critical factor was the size of the models to be analyzed. The most significant plant we analyzed has about 800 scalar variables. This number almost doubles in the case of the twin plant, and (after the elimination of equivalent variables) we are left with models having about 600 state variables. Different versions of SMV were used to tackle the resulting diagnosability problems. All the BDD-based verification engines were defeated by the size of models for coupled twin plants (i.e. no solution found after running for 24hs). This failure occurred despite

the use of advanced techniques such as dynamic reordering, invariant discovery, and conjunctive partitioning. We remark that some of these BDD-based engines, most notably Bwolen Yang's SMV, were able to tackle verification problems on *single* plants very efficiently. On the other hand, the use of model checking techniques based on SAT solving, implemented in the NuSMV system [Cimatti *et al.*, 2002], proved to be very effective on these problems. We used a combined approach, integrating bounded model checking (oriented to finding bugs) and inductive reasoning. The SAT-based engine of NuSMV was able to solve all the verification problems in less than two seconds. The analysis of the results shows that Livingstone models tend to feature a huge state space but little depth; therefore, the symbolic processing provided by SAT turns out to be very appropriate. It is worth mentioning that, while trying to refine a diagnosability property, we discovered an unexpected behavior in the model of the X-34. Further analysis highlighted a missing statement in the description. This result is quite significant, since the model had been repeatedly tested.

7 Related Work

The idea of diagnosability has received a lot of attention in the framework of Discrete Event Systems. In [Sampath *et al.*, 1995; 1996], diagnosability is precisely defined and an algorithm for checking diagnosability is presented. The approach is limited to failures represented as reachability properties. Jiang and Kumar [Jiang and Kumar, 2002] generalize the approach to the case of failures described as formulae in linear temporal logics. The approach is based on a polynomial algorithm for testing the diagnosability, formulated with techniques from automata theory [Jiang *et al.*, 2001]. In particular, they define a self-product automaton similar to our twin plant. Console, Picardi and Ribaudo [Console *et al.*, 2000] propose the use of a particular form of process algebras, PEPA, for the formalization and the analysis of diagnosis and diagnosability problems.

Our work is rather different from the works mentioned above, that are mostly oriented to the definition of the theoretical framework, and do not address the problems related to the practical application of the proposed techniques. Our objective is the definition of an effective platform for the analysis of diagnosability, that can be practically applied in the development process of diagnosis systems. The "twin-models" approach allows us to directly reuse standard model checking tools, without having to reimplement a complex tableau construction described in [Jiang *et al.*, 2001]. Furthermore, our approach preserves the semantics of the problem, thus making it possible to tune the decision procedure to the application domain. In terms of expressivity, our work shares several underlying assumptions with [Sampath *et al.*, 1995; 1996], considering failures that can be represented as reachability conditions. Compared to [Jiang *et al.*, 2001], we only tackle zero-delay diagnosability, although it seems that our framework could be extended in this respect.

Our approach makes no hypothesis on the way the controller exploits the information provided by the diagnosis system. For this reason, we introduce the notion of context in

order to qualify the conditions under which diagnosability should hold. In active diagnosis [Sampath *et al.*, 1998], the controller is designed taking into account the issues of diagnosability. Similar problems are also tackled in planning under partial observability, where the planner can decide the most appropriate actions to diagnose the fault, e.g. by probing the system with actions that will provide suitable information, and recover from it (see for instance [Bertoli *et al.*, 2002]).

8 Conclusions

In this paper, we have proposed a novel approach to the verification of diagnosability, with emphasis on its practical applicability. Our work is based on a new conceptualization of the problem, with the twin plant construction and the use of temporal logic formulae to describe the context of a diagnosability problem. To the best of our knowledge, this is the first approach to diagnosability that enables the direct exploitation of symbolic model checking technology. We tackled significant diagnosability problems from a real-world application, discussed a practical methodology for the incremental refinement of diagnosis contexts, and were able to verify large-sized problems. In the future, we will try to take into account the fact that diagnosis can propose several candidates, with different degrees of likelihood. A compositional approach to verification, exploiting the modular structure of the design, will be investigated. In the longer term, we plan to tightly integrate the approach within the Livingston toolset, in order to allow Livingstone application developers to use model checking to assist them in designing and correcting their models, as part of their usual development environment.

References

- [Bajwa and Sweet, 2002] A. Bajwa and A. Sweet. The Livingstone Model of a Main Propulsion System. In *Proc. IEEE Aerospace Conference*. IEEE, 2002.
- [Bertoli *et al.*, 2002] P. Bertoli, A. Cimatti, J. Slaney, and S. Thiebaux. Solving power supply restoration problems with planning via symbolic model checking. In *Proc. ECAI'02*, Lyon, France, 2002.
- [Bryant, 1986] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
- [Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant Planning via Symbolic Model Checking. *Journal of Artificial Intelligence Research (JAIR)*, 13:305–338, 2000.
- [Cimatti *et al.*, 2002] A. Cimatti, E. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Integrating BDD-based and SAT-based symbolic model checking. In *Proc. FRODOS*, volume 2309 of *LNAI*, pages 49–56. Springer-Verlag, April 2002.
- [Clancy *et al.*, 1999] D. Clancy, W. Larson, C. Pecheur, P. Engrand, and C. Goodrich. Autonomous control of an in-situ propellant production plant. In *Proceedings of Technology 2009 Conference*, November 1999.
- [Clarke *et al.*, 1999] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.
- [Console *et al.*, 2000] L. Console, C. Picardi, and M. Ribaud. Diagnosis and diagnosability using Pepa. In *Proc. ECAI'00*, pages 131–136, Berlino, Germany, August 2000. IOS Press.
- [Emerson, 1990] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, chapter 16, pages 995–1072. Elsevier, 1990.
- [Jiang and Kumar, 2002] S. Jiang and R. Kumar. Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications, 2002. *IEEE Trans. on Automatic Control*.
- [Jiang *et al.*, 2001] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, August 2001.
- [McMillan, 1993] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publ., 1993.
- [Moskewicz *et al.*, 2001] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. of DAC-38*, pages 530–535. ACM, 2001.
- [Muscettola *et al.*, 1998] N. Muscettola, P. P. Nayak, B. Pell, and B. Williams. Remote Agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–48, August 1998.
- [Pecheur and Simmons, 2000] Charles Pecheur and Reid Simmons. From Livingstone to SMV: Formal verification for autonomous spacecrafts. In *Proc. First Goddard Workshop on Formal Approaches to Agent-Based Systems*, 2000.
- [Sampath *et al.*, 1995] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.
- [Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems*, 4(2):105–124, March 1996.
- [Sampath *et al.*, 1998] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, July 1998.
- [Simmons *et al.*, 2001] R. Simmons, R. Goodwin, K. Zita Haigh, S. Koenig, J. O'Sullivan, and M. Veloso. Xavier: Experience with a layered robot architecture. *Intelligence*, 2001.
- [Williams and Nayak, 1996] B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI-96*, 1996.
- [Yang *et al.*, 1999] B. Yang, R. Simmons, R. E. Bryant, and D.R. O'Hallaron. Optimizing symbolic model checking for constraint-rich models. In *Proc. CAV'99*, number 1633 in *LNCS*, pages 328–340. Springer-Verlag, 1999.

On the Design of Social Diagnosis Algorithms for Multi-Agent Teams

Meir Kalech and Gal A. Kaminka

The MAVERICK Group
Computer Science Department
Bar Ilan University
Ramat Gan, Israel
{kalechm, galk}@cs.biu.ac.il

Abstract

Teamwork demands agreement among team-members to collaborate and coordinate effectively. When a disagreement between teammates occurs (due to failures), team-members should ideally diagnose its causes, to resolve the disagreement. Such diagnosis of social failures can be expensive in communication and computation overhead, which previous work did not address. We present a novel design space of diagnosis algorithms, distinguishing several phases in the diagnosis process, and providing alternative algorithms for each phase. We then combine these algorithms in different ways to empirically explore specific design choices in a complex domain, on thousands of failure cases. The results show that centralizing the diagnosis disambiguation process is a key factor in reducing communications, while run-time is affected mainly by the amount of reasoning about other agents. These results contrast sharply with previous work in disagreement detection, in which distributed algorithms reduce communications.

1 Introduction

One of the key requirements in teamwork is that agents come to agree on specific aspects of their joint task [Cohen and Levesque, 1991; Grosz and Kraus, 1996; Tambe, 1997]. With increasing deployment of robotic and agent teams in complex, dynamic settings, there is an increasing need to also be able respond to failures that occur in teamwork [Tambe, 1997; Parker, 1998], in particular to be able to diagnose the causes for disagreements that may occur, in order to facilitate recovery and reestablish collaboration (e.g., by negotiations [Kraus *et al.*, 1998]). This type of diagnosis is called social diagnosis, since it focuses on finding causes for failures to maintain designer-specified social relationships [Kaminka and Tambe, 2000].

For instance, suppose a team of four robotic porters carry a table, when suddenly one of the robots puts the table down on the floor, while its teammates are still holding the table up. Team-members can easily identify a disagreement, but they also need to determine its causes, e.g., that the robot believed the table reached the goal location, while its teammates did

not. Given this diagnosis, the robots can negotiate to resolve the disagreement.

Unfortunately, while the problem of detection has been addressed in the literature (e.g., [Kaminka and Tambe, 2000]), social diagnosis remains an open question. Naive implementations of social diagnosis processes can require significant computation and communications overhead, which prohibits them from being effective as the number of agents is scaled up, or the number of failures to diagnose increases. Previous work did not rigorously address this concern: [Kaminka and Tambe, 2000] guarantee disagreement detection without communications, but their heuristic-based diagnosis often fails. [Dellarocas and Klein, 2000; Horling *et al.*, 1999] do not explicitly address communication overhead concerns. [Frohlich *et al.*, 1997; Roos *et al.*, 2001] assume fixed communication links, an assumption which does not hold in dynamic teams in which agents may choose their communication partners dynamically (see Section 5 for details).

We seek to examine the communication and computation overhead of social diagnosis in depth. We distinguish two phases of social diagnosis: (i) selection of the diagnosing agents; and (ii) diagnosis of the team state (by the selected agents). We provide alternative algorithms for these phases, and combine them in different ways, to present four diagnosis methods, corresponding to different design decisions. We then empirically evaluate the communications and run-time requirements of these methods in diagnosing thousands of systematically-generated failure cases, occurring in a team of behavior-based agents in a complex domain.

We draw general lessons about the design of social diagnosis algorithms from the empirical results. Specifically, the results show that centralizing the disambiguation process is a key factor in dramatically improving communications efficiency, but is not a determining factor in run-time efficiency. On the other hand, explicit reasoning about other agents is a key factor in determining run-time: Agents that reason explicitly about others incur significant computational costs, though they are sometimes able to reduce the amount of communications. These results contrast with previous work in disagreement detection, in which distributed algorithms reduce communications.

The paper is organized as follows: Section 2 motivates the research. Section 3 presents the diagnosis phases and alternative building blocks for diagnosis. Section 4 specifies diag-

nosis methods which combine the building blocks in different ways, and evaluates them empirically. Section 5 presents related work, and Section 6 concludes.

2 Motivation and Examples

Agreement (e.g., on a joint plan or goal) is key to establishment and maintenance of teamwork [Cohen and Levesque, 1991; Jennings, 1995; Grosz and Kraus, 1996; Tambe, 1997]. Unfortunately, complex, dynamic settings, sometimes lead to disagreements among team-members (e.g., due to sensing failures, or different interpretations of sensor readings) [Dellarocas and Klein, 2000]. Given the critical role agreement plays in coordinated, collaborative operation of teams, we focus on the diagnosis of disagreements in this paper.

The function of a diagnosis process is to go from fault detection (where an alarm is raised when a fault occurs), to fault identification, where the causes for the fault are discovered. In diagnosing disagreements, the idea is to go beyond simple detection that a disagreement exists, to identification of the differences in beliefs between the agents, that lead to the disagreement. Such differences in beliefs may be a result of differences in sensor readings or interpretation, in sensor malfunctions, or communication difficulties. Once differences in beliefs are known, then they can be negotiated and argued about, to resolve the disagreements [Kraus *et al.*, 1998]. Unfortunately, diagnosis of such failures can be extremely expensive both in terms of computation as well as in communications (see Section 5 for quantitative evaluation), and thus a key challenge is to minimize communications and computation.

To illustrate the problem we use an example, originally reported in [Kaminka and Tambe, 2000], from the ModSAF domain (a virtual battlefield environment with synthetic helicopter pilots). In the example a team of pilot agents is divided into two: scouts and attackers. In the beginning all teammates fly in formation looking for a specific way-point (a given position), where the scouts move forward towards the enemy, while the attackers land and wait for a signal. Kaminka and Tambe [2000] report on the following failure case (which their failure detection technique captures): There are two attackers and one scout flying in formation, when the attackers detect the way-point and land, while the scout fails to detect way-point and continues to fly.

A diagnosis system, running on at least one of the agents, must now isolate possible explanations for the disagreement, of which the real cause (agents different in their belief that the way-point was detected) is but one. Each such explanation is a diagnosis hypothesis since it reports a possible reason for the disagreement between the agents. Disambiguation between these hypotheses seems very straightforward: Each agent can generate its own hypotheses, and then exchange these hypotheses with its teammates to verify their correctness. Unfortunately, since each agent is potentially unsure of what its team-members are doing, the number of hypotheses can be quite large, which means communications will suffer greatly. Furthermore, having each agent simply report its internal beliefs to the others, while alleviating communications overhead somewhat, is also insufficient. Indeed, any

approach requiring high bandwidth is not likely to scale in the number of agents [Jennings, 1995].

3 Building blocks for diagnosis

We distinguish two phases of social diagnosis: (i) selecting who will carry out the diagnosis; (ii) having the selected agents generate and disambiguate diagnosis hypotheses. To explore these phases concretely, we focus on teams of behavior-based agents. The control process of such agents is relatively simple to model, and we can therefore focus on the core communications and computational requirements of the diagnosis.

We model an agent as having a decomposition hierarchy of behaviors, where each behavior (node in the hierarchy) has preconditions (which allow its selection for execution when satisfied), and termination conditions (which terminate its execution if the behavior was selected, and the conditions are satisfied). Each behavior may have actions associated with it, which it executes once selected. It may have child behaviors, which it can select based on their matching preconditions and preference rules. At any given time, the agent is controlled by a top-to-bottom path through the hierarchy, root-to-leaf. Only one behavior can be active in each level of the hierarchy. Such a generic representation allows us to model different behavior-based controllers (e.g. [Firby, 1987; Newell, 1990]).

We assume that a team of such agents coordinates through designer-provided definition of team behaviors, which are to be selected and de-selected jointly through the use of communications or other means of synchronization. Team behaviors, typically at higher-levels of the hierarchy, serve to synchronize high-level tasks, while at lower-levels of the hierarchy agents select individual (and often different) behaviors which control their execution of their own individual role.

For instance, in the ModSAF example, the scout should fly to look for the enemy, while the attackers should wait at the way-point. All agents are executing in this case a team behavior called *wait-at-point*, in service of which the attackers are executing individual *just-wait* (land) behaviors, while the scout is executing a more complex *fly-ahead* behavior.

Disagreement between team-members is manifested by selection of different team behaviors, by different agents, at the same time. Since team behaviors are to be jointly selected (by designer specification), such a disagreement can be traced to a difference in the satisfaction of the relevant pre-conditions and termination conditions, e.g., agent A believes P , while agent B believes $\neg P$, causing them to select different behaviors. It is these contradictions which the diagnosis process seeks to discover. A set of contradictions in beliefs that accounts for the selection of different team behaviors by different agents is called a diagnosis.

3.1 Disambiguating Diagnosis Hypotheses

The first phase in the diagnosis process involves selection of the agent(s) that will carry out the diagnosis process. However, the algorithms used for selection may depend on the diagnosis process selected in the second phase (disambiguation), and so for clarity of presentation, we will begin by discussing the second phase. Assume for now that one or more

agents have been selected to carry out the diagnosis process. The agents must now identify the beliefs of their peers.

Perhaps the simplest algorithm for this is to have all team-members send their relevant beliefs to the diagnosing agent (the diagnosing agent can inform the team-members of the detection of a disagreement to trigger this communication). In order to prevent flooding the diagnosing agent with irrelevant information, each team-member sends only beliefs that are relevant to the diagnosis, i.e., only the beliefs that are associated with its currently selected behavior.

Upon receiving the relevant beliefs from all agents, the generation of the diagnosis proceeds simply by comparing all beliefs of team-members to find contradictions (e.g., agent A believes P , while agent B believes $\neg P$). Since the beliefs of the other agents are known with certainty (based on the communications), the resulting diagnosis must be the correct one. However, having all agents send their beliefs, may severely impact network load.

We thus propose a novel selective monitoring algorithm, in which the diagnosing agent controls the communications, by initiating targeted queries which are intended to minimize the amount of communications. To do this, the diagnosing agent builds hypotheses as to the possible beliefs held by each agent, and then queries the agents as necessary to disambiguate these hypotheses.

This process begins with *RESL*, a previously-published behavior recognition and belief inference algorithm [Kaminka and Tambe, 2000], which is only presented here briefly as a reminder. Under the assumption that each agent has knowledge of all the possible behaviors available to each team-member, i.e., their behavior library (an assumption commonly made in plan recognition), each observing agent creates a copy of the fully-expanded behavior hierarchy for each of its teammates. It then matches observed actions with the actions associated with each behavior. If a behavior matches, it is tagged. All tagged behaviors propagate their tags up the tree to their parents (and down to their children) such as to tag entire matching paths: These signify behavior recognition (plan recognition) hypotheses that are consistent with the observed actions of the team-member.

Once hypotheses for the selected behavior of an agent are known to the observer, it may infer the possible beliefs of the observed agent by examining the pre-conditions and the termination conditions of the selected behavior. To do this, the observer must keep track of the last known behavior(s) hypothesized to have been selected by the observed agent. As long as the hypotheses remain the same, the only general conclusion the observer can make is that the termination conditions for the selected behaviors have not been met. Thus it can infer that the observed agent currently believes the negation of the termination conditions of selected behaviors.

When the observer recognizes a transition from one behavior to another, it may conclude (for the instant in which the transition occurred) that the termination conditions of the previous behavior, and the pre-conditions of the new behavior are satisfied. In addition, again the termination conditions of the new behavior must not be satisfied; otherwise this new behavior would not have been selected. Therefore, the beliefs of the observed agent (at the moment of the transition) are:

$(\text{termination conditions of last behavior}) \wedge (\text{pre-conditions of current behavior}) \wedge \neg (\text{termination conditions of current behavior})$.

For instance, suppose an attacker is observed to have landed. The observer may conclude that either the *halt* or *wait-at-point* behavior has been selected. Furthermore, the observer can conclude that the termination conditions for *halt* do not hold, or that the termination conditions for *wait-at-point* do not hold. If the observed action indicates that the agent has just transitioned into the behavior associated with landing, then the preconditions of *halt* and *wait-at-point* would also be inferred as true.

Once the hypotheses are known, the agent can send targeted queries to specific agents in order to disambiguate the hypotheses. The queries are selected in a manner that minimizes the expected number of queries. Intuitively, the agent prefers to ask first about propositions whose value, when known with certainty, will approximately split the hypotheses space.

3.2 Selecting a Diagnosing Agent

Let us now turn to the preceding phase, in which the agents that will carry out the diagnosis are selected. Several techniques are available. First, a design-time selection of one of the agents is the most trivial approach. It requires a failure state to be declared in the team, such that the selected agents know to begin their task. Of course, one problem with this approach is that it requires all agents to be notified of the failure. A second technique that circumvents this need is to leave the diagnosis in the hands of those agents that have detected the failure, and allow them to proceed with the diagnosis without necessarily alerting the others unless absolutely necessary.

We present a novel third approach, in which selection of the diagnosing agent is based on its team-members' estimate of the number of queries that it will send out in order to arrive at a diagnosis, i.e., the number of queries that it will send out in the disambiguation phase of the diagnosis (previous section). The key to this approach is for each agent to essentially simulate its own reasoning in the second phase, as well as that of its teammates. Agents can then jointly select the agent with the best simulated results (i.e., the minimal number of queries).

Surprisingly, all agents can make the same selection without communicating, using a recursive modelling technique in which each agent models itself through its model of its teammates. This proceeds as follows. First, each agent uses the belief recognition algorithm to generate the hypotheses space for each team-member other than itself. To determine its own hypotheses space (as it appears to its peers), each agent uses recursive modelling, putting itself in the position of one of its teammates and running the belief recognition process described above with respect to itself. Under the assumptions that all agents utilize the same algorithm, and have access to the same observations, an agent's recursive model will yield the same results as the modelling process of its peers. At this point, each team-member can determine the agent with the minimal number of expected queries (the queries that split the space of the queries). In order to guarantee an agreement on the selected agent, each team-member has an ID number,

which is determined and known in advance. In case there are two agents with the same minimal number of expected queries, the agent with the minimal ID is selected. This entire process is carried out strictly based on team-members' observations of one another, with no communications other than an announcement of a disagreement.

For instance, suppose there are three agents A, B and C. To determine the diagnosing agent, A puts itself in B's position and considers the hypotheses B has about A and C, given A's observable actions. A also uses the belief recognition process described earlier to determine the number of hypotheses available about B's beliefs, C's beliefs, etc. It now simulates selecting queries by each agent, and selects the agent (say, C) with the minimal number of expected queries. B, and C also run the same process, and under the assumption that each agent's actions are equally observable to all, will arrive at the same conclusion.

Summary. We presented a space of social diagnosis algorithms: Each such algorithm operates in two phases, and we presented alternative techniques for each phase. For the selection of the diagnosing agent, we have the following methods: (i) rely on pre-selection by the designer; (ii) let the agents that detected the fault do the diagnosis; or (iii) choose the agent most likely to reduce communications (using the distributed recursive modelling technique described). In terms of computing the diagnosis, two choices are available: Either have all agents communicate their beliefs to the selected agents, or allow the diagnosing agents to actively query agents as to the state of their beliefs, while minimizing the number of queries as described above.

4 Evaluation and Discussion

The design alternatives define a space of diagnosis algorithms. This section evaluates four intuitive design decisions in this space, and draws lessons about the effects of specific design choices on overall computation and communication overhead.

Method 1. The first design choice corresponds to arguably the most intuitive diagnosis algorithm, in which all agents are pre-selected to carry out the diagnosis. When a failure is detected (and is made known to all agents) each agent communicates all its relevant beliefs to the others so that each and every team-member has a copy of all beliefs, and therefore can do the disambiguation itself.

Method 2. Method 1 uses redundant communications to achieve this distribution, while arguably only a single agent really needs to have the final diagnosis in order to begin a recovery process. Thus in method 2, the agents that detected the disagreement automatically take it upon themselves to carry out the diagnosis, unbeknownst to their teammates, and to each other. Because their team-mates do not know of the disagreement (or who has been selected to diagnose it), they cannot rely on their team-mates to communicate their beliefs without being queried (in phase 2). Instead, they use the querying algorithm discussed in the previous section. However, because the diagnosing agents also do not know of each other, in cases where more than one agent detects the disagreement, all such agents will query the other team-

members.

Method 3. The next design choice we examine corresponds to a diagnosis algorithm, in which the designer pre-selects a neutral agent. When a failure is discovered (and is made known to all agents), all team-members communicate all their relevant beliefs to the pre-selected agent.

Method 4. A final algorithm attempts to alleviate the communications overhead. It uses the recursive modelling technique to have all team-members agree on which agent is to carry out the diagnosis (this requires the detection of the disagreement to be made known). Once the agent is selected (with no communications), it queries its teammates as needed.

Table 1 summarizes the different methods. Each method is presented in a different row. The columns correspond to the different phases of the diagnosis process. The choice of algorithm is presented in each entry, along with a marking that signifies the number of agents that execute the selected technique for the phase in question.

Method	Selection	Disambiguation
1	pre-selected (N)	N (report)
2	detectors ($K \leq N$)	K (query)
3	pre-selected (1)	1 (report)
4	estimated optimal (1)	1 (query)

Table 1: Summary of evaluated diagnosis methods

For instance row 2 should be read as follows: In method 2, the agents selected to perform the disambiguation are those who detected the disagreement. K such agents exist (where K is smaller or equal than the total number of agents in the team, N), and they each execute a minimal-queries algorithm. In contrast, row 3 indicates that a single pre-selected agent executes the diagnosis, and it relies on reports from all agents to carry out the diagnosis.

Focusing on diagnosing teams of behavior-based agents in the ModSAF domain, we performed experiments in which the different diagnosis methods were systematically tested on *thousands* of failure cases, varying the number of agents, the behaviors selected by each agent, and the roles of the agents. The experiments were executed with teams of two to ten agents. For each n agents there are three sets of tests: (1) one attacker and $n-1$ scouts; (2) $n-1$ attackers and one scout; (3) $n/2$ attackers and $n/2$ scouts. For each set of x attackers and y scouts, we systematically checked all possible disagreement cases for all team behaviors (a total 2372 tests for each method). In each test we recorded the number of messages sent, and runtime of the diagnosis process.

In Table 2 we present the results of a single test, where one scout and two attackers *fly in formation*, when the scout transitions to the *wait-at-point* behavior while the attackers continue to fly. The diagnosis is that the scout detected the *way-point*, while the attackers did not.

The first column in Table 2 reports the method used. The second column presents the number of messages sent reporting on beliefs, or querying about their truth (one message per belief). The third column reports the number of messages sent informing agents of the existence of failures

(we assume point-to-point communications). The next three columns summarize the runtime of each agent.

Method	Messages		Run-time(msec)		
	Belief	Failure	A1	A2	Scout
1	16	2	12	12	12
2	8	0	0	0	192
3	8	2	3	3	9
4	3	2	87	80	99

Table 2: results of diagnosing a specific failure case

For instance, the number of messages reporting on beliefs for method 3 is 8, and 2 failure messages were sent (i.e., one of the agents detected the failure and informed the others). The runtime of all the teammates for method 3 is 3 milliseconds, except for the scout, which disambiguated the beliefs in this case, and therefore took an additional 6 milliseconds (for a total of 9 milliseconds).

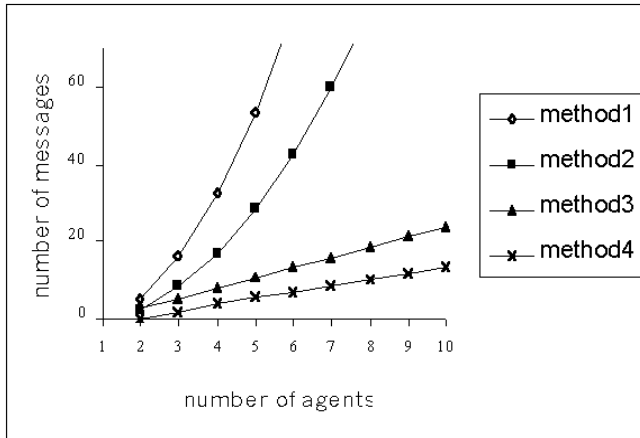


Figure 1: Average number of messages per failure case

Figures 1 and 2 summarize the results of the experiments. In both figures, the x axis shows the number of agents in the diagnosed team. Figure 1 presents the average number of belief messages for the different failure and role variations, for each team size (failure messages were ignored in the figure, since their effect is negligible). Figure 2 presents the average run-time (in milliseconds) of those same tests. The run-time of each test was taken as the maximum of any of the agents in the test.

Both figures show interesting grouping of the evaluated techniques. In Figure 1 (number of messages), methods 3 and 4 show a slow, approximately-linear growth, while methods 1 and 2 show a much faster non-linear growth. In Figure 2 (runtime), the grouping is different: Methods 1 and 3 grow significantly slower than methods 2 and 4.

The first conclusion we draw from these figures is that run-time is affected by the choice of a disambiguation method (Figure 2, ref. Table 1). Methods (here, methods 1 and 3) that rely on the agents to report their relevant beliefs do not reason about the hypothesized beliefs of others. Therefore,

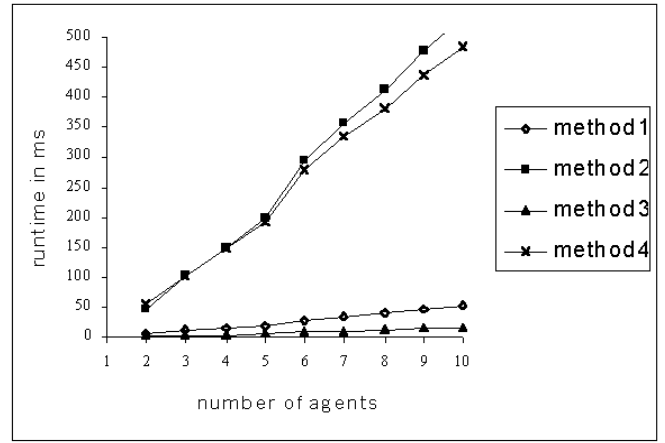


Figure 2: Average run-time per failure case

their run-time is much smaller than methods (here, methods 2 and 4) which hypothesize about the beliefs of others. However, as Figure 1 shows, the goal of reducing communications is actually achieved, as method 4 does indeed result in less communications than method 3. The question of whether the cost in run time is worth the reduction in communications is dependent on the domain.

Indeed, we draw a second conclusion from Figure 1. Despite the additional savings provided by the minimal query algorithm, the choice of a centralized diagnosing agent is the main factor in qualitatively reducing the number of messages sent, as well as in shaping the growth curve as the number of agents is scaled up. These results contrast sharply with previous work in disagreement detection, in which distributed algorithms reduce communications [Kaminka and Tambe, 2000].

5 Related Work

While diagnosis of a single-agent system is relatively well understood, and known to be computationally difficult, social diagnosis (diagnosis of social failures such as disagreements and coordination failures) remains an open area of research. In particular, to our best knowledge, an in-depth exploration of design choices in terms of communication and computation has not been done before. The most closely-related work to ours is reported in [Kaminka and Tambe, 2000]. This previous investigation provides guarantees on detection of disagreements, but only presented a heuristic approach to diagnosis, which indeed does not always succeed. The algorithms we present here succeed in the same examples where the previous heuristic approach fails. Parker [1998] reports on a behavior-based architecture which is very robust and is able to recover from failures by having robots take over tasks from failing teammates. This is done using continuous communications, but with an out explicit diagnosis process such as those described in this paper.

Dellarocas and Klein [2000] described failures handling services that use a knowledge base of generic exceptions and a decision tree of diagnoses; the diagnosis process is done by

a centralized agent traversing down the tree by asking questions about the relevant problem. Similarly, Horling [1999] uses a casual model of failures and diagnoses to detect and respond to multi-agent failures. Both investigations do not address communications overhead.

Frohlich [1997] suggested dividing a spatially distributed system into regions, each under the responsibility of a diagnosis agent. If the fault depends on two regions the agents that are responsible to those regions cooperate in making the diagnosis. This method is inappropriate for dynamic team settings, where agents cannot pre-select their communication partners. Roos [2001] expanded Frohlich's work to semantically distributed systems, where each agent looks at different aspects of the whole system. In this system each agent makes diagnosis separately and the correct diagnosis is found by exchanging the diagnoses between the agents. However, the communication links are fixed, such that each failure is diagnosed strictly by the agents that are associated with its communication link. In contrast, in teams disagreements can arise between any two agents, and thus fixed communication commitments cannot be made in design time.

Brodie [2002] tries to minimize communication costs in computer networks. He uses intelligence probes which are sent through the network in order to query remote nodes. By combining the results of different probes, failing nodes can be identified and isolated. Thus Brodie's work essentially determines the liveness status of agents, while our work focuses on fine-grain diagnosis of causes for disagreements, in terms of contradictory beliefs held by different agents.

6 Summary and Future Work

In this paper we present a novel design space for algorithms of social diagnosis, and evaluate specific design decisions in terms of their communications and computation overheads. We presented four methods of diagnosing a team of behavior-based agents, using familiar and novel algorithms, among these an algorithm that minimizes the number of diagnosis queries by using a recursive modelling technique to select the agent which will use the smallest number of queries. We then empirically and systematically evaluated the different combinations to draw general conclusions about the design of diagnosis algorithms. A first conclusion is that centralizing the diagnosis disambiguation task is critical in reducing communications. Furthermore, techniques where agents reason explicitly about the beliefs of their peers are computationally inferior (in run-time) to techniques where agents do not reason about others. However, such computation does result in a slight reduction in communications.

All methods find only the contradictions between agent beliefs, where the beliefs are derived directly from the hypothesized behaviors. But in complex behavior-based control systems, chains of inference may lead from one belief to the next. Our system is currently not able to back chain through such inference pathways, and thus is unable to draw conclusions beyond the beliefs that immediately tied to preconditions and termination conditions. We plan to tackle this challenge in the future.

Acknowledgments

We thank Meirav Hadad for help in the submission process. As always, thanks to K. Ushi and K. Raviti.

References

- [Brodie *et al.*, 2002] M. Brodie, I. Rich, and S. Ma. Intelligence probing: A cost-effective approach to fault diagnosis computer networks. *IBM Systems Journal*, 41-3, 2002.
- [Cohen and Levesque, 1991] Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 35, 1991.
- [Dellarocas and Klein, 2000] Chrysanthos Dellarocas and Mark Klein. An experimental evaluation of domain-independent fault-handling services in open multi-agent systems. pages 95–102, Boston, MA, 2000. IEEE Computer Society.
- [Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. 1987.
- [Frohlich *et al.*, 1997] P. Frohlich, I. Mora, W. Nejdl, and M. Schroeder. Diagnostic agents for distributed systems. *proceedings of ModelAge*, 97, January 1997.
- [Grosz and Kraus, 1996] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group actions. *JAIR*, 86:269–358, 1996.
- [Horling *et al.*, 1999] Bryan Horling, Victor R. Lesser, Regis Vincent, Ana Bazzan, and Ping Xuan. Diagnosis as an integral part of multi-agent adaptability. Technical Report CMPSCI Technical Report 1999-03, University of Massachusetts/Amherst, January 1999.
- [Jennings, 1995] Nicholas R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. 75(2):195–240, 1995.
- [Kaminka and Tambe, 2000] Gal A. Kaminka and Milind Tambe. Robust multi-agent teams via socially-attentive monitoring. 12:105–147, 2000.
- [Kraus *et al.*, 1998] Sarit Kraus, Sycara Katia, and Amir Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1–2):1–69, 1998.
- [Newell, 1990] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.
- [Parker, 1998] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.
- [Roos *et al.*, 2001] N. Roos, A. Teije, A. Bos, and C. Witeveen. Multi agent diagnosis with spatially distributed knowledge. *BNAIC*, 2001.
- [Tambe, 1997] Milind Tambe. Towards flexible teamwork. *JAIR*, 7:83–124, 1997.

Model-based Diagnosis of Hybrid Systems

Sriram Narasimhan

QSS Group
NASA Ames Research Center
Moffett Field, CA 94035
sriram@arc.nasa.gov

Gautam Biswas

Dept. of EECS & ISIS
Vanderbilt University
Nashville, TN 37235
gautam.biswas@vanderbilt.edu

Abstract

Recent years have seen a proliferation of embedded systems that combine a digital (discrete) supervisory controller with an analog (continuous) plant. Diagnosing faults in such hybrid systems, require techniques that are different from those used for discrete and continuous systems. In addition, these algorithms have to be deployed online to meet the real time requirements of embedded systems. This paper presents a methodology for online tracking and diagnosis of hybrid systems. We demonstrate the effectiveness of the approach with experiments conducted on the fuel transfer system of fighter aircraft.

1 Introduction

This paper addresses the problem of designing and implementing online monitoring and diagnosis systems for complex systems whose behavior is hybrid (discrete + continuous) in nature. Hybrid modeling covers naturally occurring systems, such as cell-cycle control systems in biology. They also capture the behavior of embedded systems that are common in the avionics, automotive, and robotics domains. This wide applicability of hybrid systems has inspired a great deal of research from both control theory and theoretical computer science.

We focus on a special class of embedded hybrid systems, characterized by continuous plant dynamics and a discrete supervisory controller. The plant dynamics are defined by continuous state variables associated with the components of the plant. The controller generates actuator signals at discrete time points that can change the plant configuration by turning components ON and OFF, and changing component parameter values and the set points of regulators. Therefore, hybrid system models have to seamlessly integrate discrete and continuous behavior analyses using multiple system models. As a result, tasks like monitoring, fault diagnosis, and control require appropriate model selection and switching to be performed online as system behavior evolves.

This paper discusses methodologies for the model-based diagnosis (MBD) of hybrid systems. Current techniques in MBD apply well to dynamic systems whose behavior is modeled with discrete event [9, 17], or con-

tinuous models [5, 14]. Discrete event approaches to hybrid system diagnosis are based on abstractions of nominal and faulty behavior system behavior into event trajectories. This process may result in loss of information critical for fault isolation and control. Our work in continuous diagnosis has demonstrated that behavior transients are the key to quick diagnosis of abrupt faults[10]. It may also be computationally expensive to pre-enumerate all possible nominal and faulty behavior trajectories. Traditional algorithms for continuous diagnosis use a single model that does not accommodate discrete changes. Therefore, discrete effects of mode changes have to be modeled by complex continuous non-linear functions that are hard to analyze online in real time.

Recent work on diagnosis of hybrid systems [3, 6, 8] has focused on discrete faults, and requires the pre-enumeration of the model in all modes to perform diagnosis. We present an online model-based diagnosis methodology for parametric faults in hybrid systems that is based on tracking hybrid behaviors (continuous behaviors interspersed with discrete changes), but unlike hybrid automata models [1] pre-enumeration of all system modes is avoided by generating models at runtime as mode switches occur.

The fault isolation task has to take into account possible mode changes during diagnostic analysis. The occurrence of a fault necessarily implies that one no longer has a correct model of system behavior; therefore, mode changes cannot be correctly predicted. To address this, the fault isolation task incorporates a search process, where mode changes may have to be hypothesized and incorporated into the consistency-based diagnosis scheme. The fault isolation process becomes even more complicated if fault detection is delayed, and the diagnosis algorithm has to roll back modes to identify the mode in which the fault occurred. We have developed a generic tracking, fault detection, and fault isolation scheme, which address all of the issues we have outlined above. The rest of the paper presents our hybrid modeling, tracking, and diagnosis scheme for solving the hybrid diagnosis problem.

2 Unified Modeling Framework

We use a unified modeling framework called hybrid bond graphs that extends continuous bond graph modeling [7]

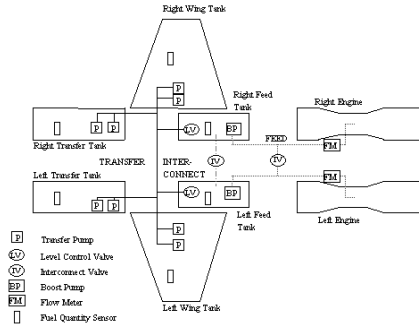


Figure 1: Fuel System Schematic

to provide compact representations for hybrid models. Its component based and hierarchical representation is expressed as topological structures that facilitate causal analysis of system dynamics. It also provides standard techniques for deriving state space and input output equation formulations that are suitable for tracking and estimation tasks [15].

2.1 Bond Graphs

A bond graph (BG) is a domain-independent topological representation that captures energy-based interactions among the different physical processes that make up the system. The vertices in the graph represent subsystems modeled as generic physical processes: capacities, inertias, and dissipators that can have linear and non-linear behaviors. Bonds are energy pathways by which subsystems/processes exchange energy in the system. Two additional types of vertices (0 and 1 junctions) represent domain independent generalizations of Kirchhoff's laws and are used as connection points between the sub-systems. There exist systematic techniques to construct the bond graph from the system description [2].

2.2 Hybrid Bond Graphs

Additional mechanisms are introduced into the continuous BG language to include discrete transitions and model switching. We use switched junctions proposed by Mosterman and Biswas [13], where each junction in the bond graph may be switched on (activated) and off (deactivated). An activated junction behaves like a conventional BG junction. All the bonds incident on a junction turned off are made inactive, and hence do not play any part in the system dynamics. Note that activating or deactivating junctions affect the behavior of adjoining junctions.

A Finite State Automaton (FSA) implements the junction switching function. The FSA may have several states, and each state maps to either the off mode (i.e., it causes the junction to turn off) or the on mode (i.e., the junction turns on) of the junction. Mode transitions defined solely by external controller signals define *controlled switching*, and those expressed by internal variables crossing boundary values define *autonomous switching*. For controlled switching the control signal is provided as input to the FSA. For autonomous switching, the function determining the transition condition is pro-

vided as input to the FSA. The overall mode of the system is determined by a parallel composition of modes of modes of the individual switched junctions.

Formally, hybrid bond graphs can be defined as a triple: $HBG = \{BG, M, \sigma\}$, where BG is the bond graph model, $M = \{M_1, M_2, \dots, M_k\}$ is a set of finite state of automata, and σ is the mapping between each M_i and a junction in the bond graph. Each M_i is a finite state automaton of the type described above, with an output function that maps each state of M_i to either on or off. A system mode change is defined by one or more junction automata changing state, and this results in a new bond graph model.

Figure 1 illustrates the fuel transfer system that we use for our experiments. The fuel system is designed to provide an uninterrupted supply of fuel at a constant rate to the aircraft engines, and at the same time to maintain the center of gravity of the aircraft. The system is symmetrically divided into the left and right parts (top and bottom in the schematic). The four supply tanks (Left Wing (LWT), Right Wing (RWT), Left Transfer (LTT), and Right Transfer (RTT)) are full initially. During engine operation, fuel is transferred from the supply tanks to the receiving tanks (Left Feed (LFT) and Right Feed (RFT)) based on a pre-defined sequence. The fuel transfer sequence is controlled by valves on pipes at the outlet of the supply tanks and the inlet to the feed tanks. The hybrid bond graph segment for the connection between a wing tank and a feed tank, shown in Figure 2, illustrates the component-oriented modeling approach for the HBG. An element called switching implements the finite state automata discussed earlier. The HBG framework also associates one or more parameters with system components. We exploit this in defining a component-based diagnosis methodology, where faults in components are represented as deviations in their parameter values. For example, there are six potential fault candidates in the fuel transfer subsystem in Figure 2. (Pump, Efficiency, Wing Tank, Pipe, Switched, and Feed). In earlier work, we have shown the equivalence between the HBG framework and the hybrid automata representation [16].

2.3 Alternate Model Representations

The bond graph can be used to systematically derive alternate model representations. Three representations are used to solve different sub-tasks in the diagnosis scheme: (i) state space equations, for tracking of continuous behavior, (ii) temporal causal graphs for qualitative fault isolation, and (iii) input output equations, for parameter estimation and refinement of the fault isolation results. A

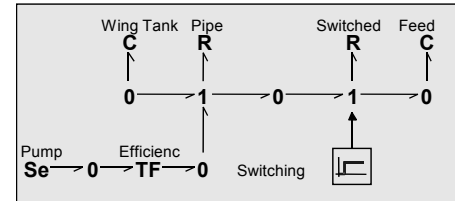


Figure 2: HBG of Fuel Transfer subsystem

detailed description of temporal causal graphs (TCG) can be found in [14, 15] and is not repeated here. Essentially the TCG captures causal and temporal relations between variables in the system. The vertices in the graph represent variables of the system and the edges (with labels) represent the types of relations between the variables.

3 MBD Architecture

Our diagnosis architecture implements a scheme to track the nominal system dynamics using an observer that is robust to model uncertainties and noise in the measurements. It uses a fault detection scheme to trigger the fault isolation scheme when discrepancies are detected between the observed and predicted measurements, tracks and analyzes the fault transients using fault signatures to isolate the fault, and then employs a quantitative parameter estimation scheme to determine the magnitude of the fault. Our work focuses on component parameter faults, which are multiplicative, i.e., faults directly affect the system dynamics models. As discussed earlier, the hybrid nature of the system complicates the tracking and diagnosis tasks, because mode transitions cause model switching, which has to be included in the online behavior tracking and fault isolation algorithms. For pragmatic reasons we simplify our algorithms by making the single fault assumption.

We have developed a novel approach that combines qualitative and quantitative algorithms for fault isolation. This extends our earlier work [12] on fault isolation in continuous systems. The qualitative approach overcomes limitations of quantitative schemes, such as convergence and accuracy problems in dealing with complex nonlinearities and lack of precision of parameter values in system models. It plays a significant role in cutting down on computational complexity, enabling online algorithms for fault isolation in the hybrid framework. The qualitative reasoning scheme is fast and effective, but it has limited discriminatory ability. To uniquely identify the true fault candidate, we employ a quantitative parameter estimation scheme, which also returns the magnitude of the deviated parameter.

3.1 Tracking and Fault Detection

Our hybrid observer is implemented as a combination of an extended Kalman filter (EKF) and a hybrid automaton to track continuous behavior in individual modes of operation, and discrete mode changes (controlled and autonomous), respectively. At mode changes, the new state space model and the initial state of the system are recomputed. Model uncertainty and measurement noise are implemented as white, uncorrelated Gaussian distributions with zero mean. The state space model in mode q is defined as:

$$\mathbf{x}_{k+1} = \mathbf{F}_q(\mathbf{x}_k)\mathbf{x}_k + \mathbf{G}_q(\mathbf{x}_k)\mathbf{u}_{k+1} + \mathbf{w}_k$$

$$\mathbf{y}_{k+1} = \mathbf{C}_q(\mathbf{x}_k)\mathbf{x}_{k+1} + \mathbf{D}_q(\mathbf{x}_k)\mathbf{u}_{k+1} + \mathbf{v}_{k+1}$$

where $\mathbf{w}$ is distributed $N(0, \mathbf{Q})$ and $\mathbf{v}$ is distributed $N(0, \mathbf{R})$, and $\mathbf{Q}$ and $\mathbf{R}$ are process and measurement noise covari-

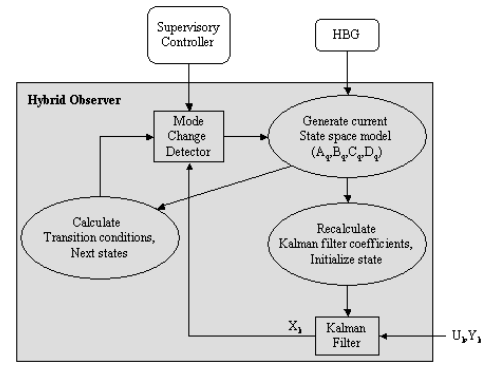


Figure 3: Hybrid Kalman Filter

ance matrices. It is assumed that $\mathbf{w}_k$ incorporates the $\Delta \mathbf{F}_q \cdot \mathbf{x}_k$ term that captures modeling errors in the system. In our work, the $\mathbf{Q}$ and $\mathbf{R}$ matrices were determined empirically. The extended Kalman filter algorithm follows the methodology presented in [4].

Mode change calculations are based on the system mode at time step k , q_k , and the continuous state of the system, $\mathbf{x}_k$. The discrete controller signals to the plant are assumed known. For controlled transitions, we assume such a signal is input at time step k , and the appropriate mode transition is made at time step $k+1$ to q_{k+1} . For autonomous transitions, the estimated state vector, $\mathbf{x}_k$ is used to compute the Boolean functions that signal mode transitions. Note that several transition functions may be triggered simultaneously. They are combined to derive the new system mode. A mode transition results in a new state equation model, i.e., the matrices $\mathbf{F}_q$, $\mathbf{G}_q$, $\mathbf{C}_q$, and $\mathbf{D}_q$ are recalculated online. We have developed an efficient symbolic solver that can construct state equation models from equation fragments. The equation fragments correspond to constituent equations defining component behavior, and the junction relations. When switching occurs, sets of equation fragments are de-activated, and others are activated. The new state equations are then derived incrementally. To simplify analysis, we assume that mode changes and faults occur only after the Kalman filter state estimate has converged to its optimal behavior. Further details of the observer implementation are presented in [15].

Fault detection is performed by first computing estimates of the output variables y_k from the state estimates $\mathbf{x}_k$. We then compute a smoothed estimate $\hat{y}_k$ using an FIR filter. Finally we compute the residual $(\hat{y}_k - o_k)$, where o_k are the observations at time step k . If this residual r_k is above a threshold ε for a pre-defined number of time steps, then a fault is signaled.

3.2 Fault Isolation and Identification

Once a fault has been detected, fault isolation and identification is performed to uniquely isolate the fault and determine its magnitude. Our fault isolation and identification architecture is presented in Figure 4 involves three steps: (i) *qualitative roll-back*, (ii) *qualitative roll-forward*, and (iii) *quantitative parameter estimation*.

For hybrid systems, discontinuous changes in measured variables can only occur at the point of failure or at points at which discrete mode changes occur in the plant behavior. At all other time points the plant behavior is continuously differentiable. We take advantage of this fact for qualitative analyses of all measured variables, y_k . The residual (r_k) for any variable is defined as the difference between the measure plant output and the nominal expected plant output. Since both of these are continuously differentiable after the fault occurrence, and after each mode change, the residual can be approximated by the Taylor series expansion:

$$r_t = r_{T_{j_0}} + r'_{T_{j_0}} \frac{(t - T_{j_0})}{1!} + r''_{T_{j_0}} \frac{(t - T_{j_0})^2}{2!} + \dots + r^{(k)}_{T_{j_0}} \frac{(t - T_{j_0})^k}{k!} + R_k$$

We can then represent the residual as the coefficients of the magnitude and higher order derivative terms of the residual. Instead of representing them in quantitative form, qualitative values (-, 0, and +) are used to indicate if the coefficient is below, at, or above zero.

The qualitative analyses that comprise of the roll-back and roll-forward steps work on these qualitative coefficients. After detection of the fault, the signal to symbol generator is responsible for converting the measured observations to symbolic form at each time step. The discussion of the computation of the residual and the converting it to symbolic form is presented in [11], and not repeated here.

The roll-back algorithm can be summarized as follows. Given the observer estimated mode trajectory $Q = \{q_1, q_2, \dots, q_k\}$, we first use the back propagation algorithm [14] to generate hypotheses in mode q_k . The deviated symbols at the time of fault detection (α) are back propagated through the temporal causal graph in mode q_k to identify causes for the deviations. Since the fault may have occurred in previous modes, we then go back in the mode trajectory and create hypotheses in each of the previous modes $q_{k-1}, q_{k-2}, \dots, q_{k-n+1}$, where n is a number determined externally by diagnosability studies. During the crossover from a mode to a previous mode, the symbols need to be propagated back across the mode change. This is done by using the inverse of the reset functions (γ^{-1}) associated with the mode transition. For example, the symbols to be propagated in mode q_{k-1} is obtained as $\gamma^{-1}(\alpha, q_k, q_{k-1})$. The hybrid hypotheses generation algorithm returns a hypotheses set, $H = \{h_1, h_2, \dots, h_m\}$, where each hypothesis h_i is a three-tuple $\{q, p, \lambda\}$, where q represents the mode in which the fault is hypothesized to have occurred, p is the parameter whose deviation corresponds to the fault, λ is the direction of deviation of parameter p .

The occurrence of the fault may change the parameters of the functions that determine autonomous transitions leading the observer to incorrectly predict (or not predict) an autonomous transition. Hence the current mode estimated by the hybrid observer may not be the actual mode of system under hypothesized fault conditions. We need to estimate the current mode of the system for each hypothesis. However, we still do not have the quantitative value for the faulty parameter, implying that we cannot accurately determine the current mode of the system. To

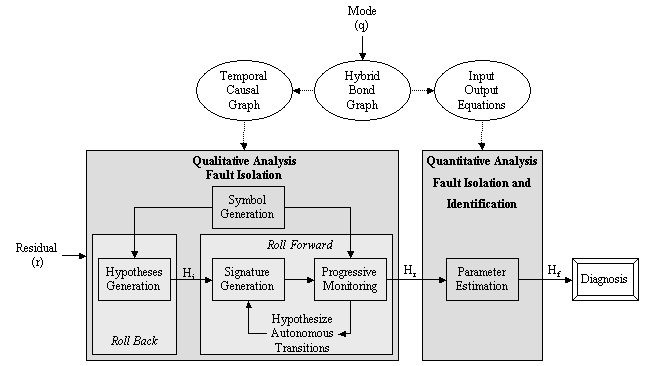


Fig. 4: Fault Isolation and Identification Architecture

solve this problem we take advantage of the fact that the same sequence of mode transitions in any order would lead the system to the same end state. This follows from the fact that each mode transition essentially changes the status of a switched junction in the hybrid bond graph representation. Based on this observation, for each hypothesis, we apply all controlled transitions (assuming that no autonomous transitions have occurred) that have occurred since the hypothesized fault mode to get an estimated current mode. This is known as the roll-forward process.

We can now use the model (TCG) to predict the expected qualitative values for the residuals (signatures) in the current estimated mode for each hypothesis. This is done through a forward propagation algorithm [14]. The fault hypotheses in the estimated current mode are compared with the symbols generated by the signal to symbol generator using a progressive monitoring scheme [14]. If there is a mismatch, the hypothesis cannot be dropped immediately, we assume that an autonomous mode transition may have caused this mismatch. We apply all possible autonomous transitions to the current estimated mode and derive m new estimated current modes for the hypothesis, where m is the number of possible autonomous transitions. For each of these m new modes, we can generate the qualitative signature using forward propagation and compare this against observations. In case of mismatch we hypothesize occurrence more autonomous transitions and repeat the process. Once the total number of transitions (controlled + hypothesized autonomous) exceeds the diagnosability limit, further mismatches in signatures and symbols eliminate hypotheses.

In other work [12], we have shown the limited discriminatory capabilities of the qualitative progressive monitoring scheme. This often leads to multiple fault hypotheses being reported as the diagnostic result. Even when we are left with only one hypothesis, determining the magnitude of the parameter associated with the hypothesis is essential to continue tracking the plant behavior in the faulty situation. We use a parameter estimation technique based on the least-squares estimation method for further fault isolation and identification. Applying a statistical hypothesis testing scheme to the error in the fault identification task leads to unique fault isolation,

and an estimate of the deviated parameter value. Consider the input output equation form:

$$Y = ((\sum_i \sum_j b_{ij} q^j) / (\sum_k a_k q^k)) U,$$

where U are the inputs, Y are the outputs, a 's and b 's are constant coefficients and q is the forward delay operator. The typical parameter estimation task is to estimate the a 's and b 's using measurements, u and y . The optimal estimate is given by: $\theta_{est} = (\Phi^T \Phi)^{-1} \Phi^T Z$, where θ is a vector of a 's and b 's, Φ is the regression vector, and Z is current output vector. The single fault assumption implies that only one parameter is unknown. So, for each remaining hypothesis, we rewrite θ in terms of the corresponding hypothesized fault parameter p . If we assume a first order polynomial relation (it is possible to extend this to arbitrary polynomial relations): $\theta = K_1 p + K_0$, where K_1 and K_0 are matrix constants. Now we can reformulate the estimation problem and obtain the optimal estimate for p as:

$$p = (K_1^T \Phi^T Z - K_1^T \Phi^T K_0 - K_1^T \Phi^T K_1 - Z^T \Phi K_1) / (2 K_1^T \Phi^T K_1)$$

**Table 1: Trace of Fault Isolation and Estimation
Left Wing Tank Pump Degradation**

Time	Measured Deviation	Fault Set
433	Transfer Manifold Pressure (XMP)	13 fault candidates
434	XMP: discontinuous	10 candidates
439	<i>Mode Change: Left Feed Tank On</i>	
465	<i>Mode Change: Left Feed Tank Off</i>	
469	LeftWing Tank Pressure ++	LWT_Pipe.R + LeftWingTank.TF - LeftWingTank.R + Leg26.R -
471	<i>Mode Change: Left Wing Tank - Second Pump On</i>	
489	Left Feed Tank Pressure --	LWT_Pipe.R + LeftWingTank.TF - LeftWingTank.R +
528	<i>Mode Change: Left Feed Tank On</i>	
528	Parameter estimation started	LeftWingTank.TF fault coefficient: 0.658041

For each remaining hypothesis, we compute the input output equations of the system in the estimated current mode from the bond graph. This can be achieved by computing the temporal causal graph from the bond graph. The TCG can then be used to compute the signal flow graph, which can be used to derive the input output equations. Please note that the signal flow graph still contains parameters in symbolic form as opposed to actual numeric values. This gives us a parameterized input output equation. We then calculate the K_1 and K_0 matrices. Finally we accumulate u and y values, and estimate p using the above expression. This may also be used for fault isolation by plugging back the estimated parameter in the state space equations and computing the predicted values for the outputs. Hypotheses whose predicted output values are statistically different from actual output values are eliminated. Table 1 illustrates an experimental run for a left wing tank pump degradation (33% loss) at time step 150. The initial change was observed in the transfer manifold pressure at time step = 433 (Fig. 5), but two mode changes occurred between the fault occurrence and its detection. The roll back process was employed, and this produced an initial list of 13 candidates. Detecting that the transfer manifold pressure was discontinuous

reduced the candidate set to 10 faults. As other measurements deviated over time (first left wing tank pressure, then left feed tank pressure), the candidate set was further refined. Mode changes required the re-derivation of the system models, and the computation of new signatures to track system behavior. The qualitative scheme reduced the candidate set to 4, and then the parameter estimation scheme was invoked. This resulted in generating the correct fault hypothesis, and a correct estimation of the faulty parameter.

4 Experimental Results

We demonstrate the effectiveness of our diagnosis scheme on a real world example, the fuel system of fighter aircraft (Fig. 1). The pumps are modeled as a source of effort (pressure) with a transformation factor that defines its efficiency. The pipes are modeled as nonlinear resistances.

The diagnosis experiments used a controller sequence provided by Boeing. The performance of the hybrid observer in tracking the nominal data (with 3% noise) through mode transfers is illustrated in Figure 5 for the transfer manifold and left wing tank pressure measurements. The dots represent the measured data and the black line shows the observer estimates. A number of diagnosis experiments were run for different noise values and fault magnitudes (see [15] for details).

Table 2: Fuel System Diagnosability

	WTP	WTR	TTP	TTR	SPR	FTP	FTR
WTP	-	x	√	√	√	√	√
WTR	x	-	√	√	√	√	√
TTP	√	√	-	x	√	√	√
TTR	√	√	x	-	√	√	√
SPR	√	√	√	√	-	√	√
FTP	√	√	√	√	√	-	x
FTR	√	√	√	√	√	x	-

Table 2 summarizes the different fault classes that can be distinguished by our diagnosis algorithms. The fault classes are as follows: (i) Wing Tank Pump (WTP), (ii) Wing Tank Resistance (WTR), (iii) Fuselage Tank Pump (TTP), (iv) Fuselage Tank Resistance (TTR), (v) Switched Pipe Resistance (SPR), (vi) Feed Tank Pump (FTP), and (vii) Feed Tank Resistance (FTR). The √ mark in row i and column j indicates that fault class i can be distinguished from fault class j . The x mark indicates that the current controller sequence and set of measurements are not sufficient to distinguish between the pair in question. From the table, we see that we cannot distinguish between tank pump faults and tank outlet pipe resistance faults. However, this is true only for a pump efficiency (T_F -) decrease and pipe resistance increase (R+). Since the pump efficiency cannot increase (no T_F + fault), pipe resistance decreases, R-, (i.e., leaks) can be uniquely identified. All other classes of faults can be distinguished

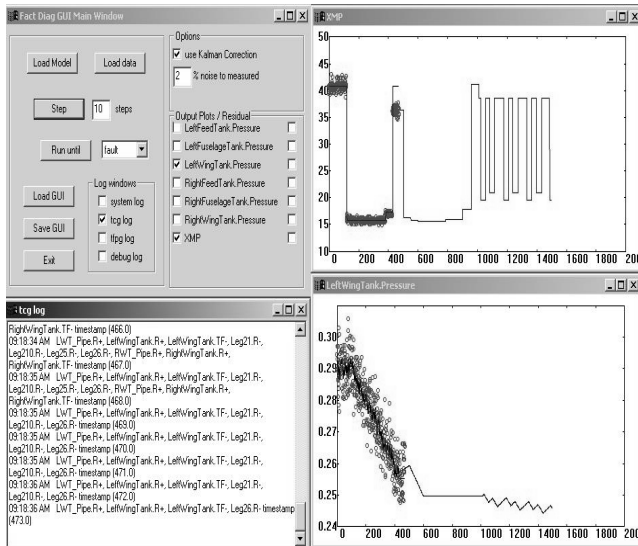


Figure 5: Transfer manifold and Left Wing Tank Pressure

from one another. In some cases, the isolation may be achieved only after mode changes occur.

5 Summary

In this paper we have presented an integrated approach to solving the tracking, fault detection, isolation, and identification tasks for hybrid systems. The novel contribution of the presented work is the extension of continuous system model-based diagnosis techniques to hybrid systems. These include the hybrid observer that combines an extended Kalman filter and hybrid automaton, hybrid fault isolation through roll back and roll forward using qualitative analysis, and the single parameter estimation for further fault isolation and identification. Our work is motivated by the requirements of the fault accommodation task, where diagnosis has to be performed online for embedded systems during their operation. Hybrid diagnosis techniques directly apply to embedded systems and else where [15]. We have also demonstrated through time and space complexity analysis that our algorithms can be applied to online analysis in resource constrained environments.

Acknowledgments

The DARPA/IXO SEC program (F30602-96-2-0227), NASA IS NCC 2-1238, and The Boeing Company (Kirby Keller and Tim Bowman) have supported the activities described in this paper. We would like to thank Dr. Gabor Karsai, Tivadar Szemethy, Nagabhushan Mahadevan, and Eric Manders for their help.

References

1. Alur, R., et al., *Hybrid Automata - An Algorithmic Approach to Specification and Verification of Hybrid Systems*, *Lecture Notes in Computer Science: Hybrid Systems I*. 1994, Springer Verlag. p. 209-229.

2. Biswas, G. and X. Yu. *A Formal Modeling Scheme for Continuous Systems: Focus on Diagnosis*. in *Proc. IJCAI 1993*. Chambery, France, p 1474-1479.
3. Dearden, R. and D. Clancy. *Particle Filters for Real-Time Fault Detection in Planetary Rovers*. in *Thirteenth Intl Wkshp. on Principles of Diagnosis*, 2002. Semmering, Austria.
4. Gelb, A., *Applied Optimal Estimation*. 1979, Cambridge, MA, USA: MIT Press.
5. Gertler, J., *Fault detection and Diagnosis in Engineering Systems*. 1998, New York: Marcel Dekker.
6. Hofbaur, M.W. and B.C. Williams. *Mode Estimation of Probabilistic Hybrid Systems*. in *Fifth International Workshop on Hybrid Systems: Computation and Control (HSCC '02)*. 2002. Stanford, CA, USA: Springer.
7. Karnopp, D., D.L. Margolis, and R.C. Rosenberg, *System dynamics : modeling and simulation of mechatronic systems*. 3rd ed. 2000, New York: Wiley.
8. Koutsoukos, X., et al. *Fault Modeling for Monitoring and Diagnosis of Sensor-Rich Hybrid Systems*. in *IEEE Conference on Decision and Control*. 2001. Orlando, FL.
9. Lunze, J. and J. Schroder, *Sensor and Actuator Fault Diagnosis of Systems with Discrete Inputs and Outputs*. in press, *IEEE Transactions on Systems, Man, and Cybernetics*, March, 2002.
10. Manders, E. and G. Biswas. *Transient Detection and Analysis for Diagnosis of Abrupt Faults in Continuous Dynamic Systems*. in *Intl Wkshp. on Intelligent Signal Processing (WISP '01)*. 2001. Budapest, Hungary.
11. Manders, E., P.J. Mosterman, and G. Biswas. *Signal to Symbol Transformation techniques for Robust Diagnosis in TRANSCEND*. in *Intl Wkshp on Principles of Diagnosis*. 1999. Loch Awe, Scotland.
12. Manders, E., et al. *A combined Qualitative/Quantitative approach to efficient Fault Isolation in complex dynamic systems*. in *Intl. SafeProcess Symposium*, 2000. Budapest, Hungary, p.512-517.
13. Mosterman, P.J. and G. Biswas, *A Theory of discontinuities in Physical System Models*. *Journal of Franklin Institute*, 1998. **335**(B): p. 401-439.
14. Mosterman, P.J. and G. Biswas, *Diagnosis of Continuous Valued Systems in Transient Operating Regions*. *IEEE Transactions on Systems, Man, and Cybernetics*, 1999. **29**: p. 554-565.
15. Narasimhan, S., *Model-based Diagnosis of Hybrid Systems*, in *Computer Science*. 2002, Vanderbilt University: Nashville.
16. Narasimhan, S. and G. Biswas. *An Approach to Model-Based Diagnosis of Hybrid Systems*. in *Hybrid Systems: Computation and Control (HSCC '02)*. 2002. Stanford, CA: Springer Verlag.
17. Sampath, M., et al., *Failure Diagnosis using Discrete-Event Models*. *IEEE Transactions on Control Systems Technology*, 1996. **4**: p. 105-124.

Automated Qualitative Domain Abstraction

Martin Sachenbacher

MIT Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
sachenba@mit.edu

Peter Struss

TU München, Fakultät für Informatik
85748 Garching, Germany
struss@in.tum.de

Abstract

Automated problem-solving for engineered devices is based on models that capture the essential aspects of their behavior. In this paper, we deal with the problem of automatically abstracting behavior models such that their level of granularity is as coarse as possible, but still sufficiently detailed to carry out a given behavioral prediction or diagnostic task. A task is described by a behavior model, as composed from a library, a specified granularity of the possible observations, and a specified granularity of the desired results. The goal is to determine partitions for the domains of the variables (termed qualitative values) that are both necessary and sufficient for the task at hand. We present a formalization of the problem within a common relational (constraint-based) framework, present results regarding solutions to task-dependent qualitative domain abstraction, and devise methods for automatically determining qualitative values for a device model.

1 Introduction

Model-based systems ([Hamscher *et al.*, 1992; Weld and de Kleer, 1990]) represent knowledge about the structure and behavior of a physical system in terms of a behavior model, and use it to support engineering tasks such as behavior prediction, diagnosis, planning and testing. When constructing model-based systems, one of the most difficult parts is modeling the device. A fundamental idea to support and facilitate modeling is to compose models from model fragments, that is, re-usable elements of knowledge about a device that can be organized in a library ([Falkenhainer and Forbus, 1991]). This requires that model fragments have to be formulated, as far as possible, in a generic way and independent of their specific application context. However, it also means that information about the task a model will be used for cannot be anticipated in the model fragments.

But a model needs to be suited for the problem-solving task at hand in order to provide an effective and efficient solution to it. Using always the most accurate and most detailed model available may make the respective problem-solving task intractable, or at least unnecessarily complex and

resource-consuming. For instance, for the task of diagnosing a device in an on-board environment, it is crucial to have a model that focuses only on those aspects that are essential to the goal of discriminating between its normal and faulty behavior. Any unnecessary details that are not relevant to this task impair its ability to meet the stringent time and space requirements of this application. In general, models straightforwardly composed from a library tend to be either inefficient, because they are overly detailed (that is, too fine-grained), or ineffective, because they are not detailed enough (that is, too coarse-grained) for the task they will be used for.

The approach pursued in this paper is therefore to automatically re-formulate a behavior model, after it has been composed, to a level of abstraction that is adequate for the specified task. We focus on the abstraction of the domains of variables, that is, the problem of deriving sets of meaningful qualitative values. Although some work has been carried out on finding qualitative values within a specific context, such as simulation ([Kuipers, 1986]), the general problem of characterizing and systematically deriving qualitative values for an arbitrary relational (constraint-based) behavior model is a relatively unexplored area. However, much of the work in qualitative reasoning about physical systems ([Weld and de Kleer, 1990]) relies on this type of abstraction. The resolution of a behavior model's domains has a strong effect on the size of the model, the efficiency of reasoning with the model, and the size of the solutions. Furthermore, within an on-board or real-time setting, the number of qualitative values determines how many of the observations will be qualitatively different, and therefore it influences the frequency at which reasoning has to be initiated at all.

1.1 Example

Consider, for example, the system depicted in figure 1. The device is a simplified version of a pedal position sensor used in a passenger car. Its purpose is to deliver information about the position of the accelerator pedal to the electronic control unit (ECU) of the engine management system. The ECU uses this information to calculate the amount of fuel that will be delivered to the car engine.

The pedal position is sensed in two ways, via the potentiometer as an analogue signal, v_{pot} , and via the idle switch as a binary signal, v_{switch} . The idle switch changes its state at a particular value $pos_{switching}$ of the mechanically transferred

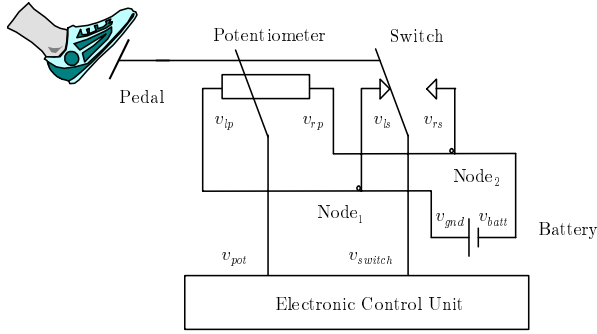


Figure 1: The Pedal Position Sensor

pedal position. The reason for the redundant sensing of the pedal position is that the signals v_{pot} and v_{switch} are cross-checked against each other by the on-board control software of the ECU. This plausibility check is a safety feature of the system, in order to avoid cases where a wrong amount of fuel injected evokes dangerous driving situations.

Assume we want to perform the plausibility check between the electrical signals v_{pot} and v_{switch} automatically by the means of a behavior model of the system. For the potentiometer model fragment, this requires a distinction in the domain of v_{pot} that corresponds to the switching point $pos_{switching}$ of the switch. This is the only distinction in this domain that is required for the purpose at hand.

The problem is that this particular distinction cannot be anticipated in a generic model fragment of the potentiometer component, because it would not make any sense in a different structure. It is only the specific combination of the potentiometer and the switch together with the pursued task that requires this distinction. In contrast, other tasks such as control or design might require more detailed domains that would allow to relate the position of the switch to particular potentiometer voltages.

The problem is important, because it impairs the idea of using a model of the pedal position sensor as a common basis for different tasks. For engineered systems, it is typical that several tasks along the product's life cycle — such as failure modes and effects analysis (FMEA), on-board diagnostics development, generation of repair manuals or workshop diagnosis — share a significant amount of common knowledge about the behavior of the system under consideration. It would be unacceptable having to *manually* create models from scratch that are tailored to each of these tasks.

2 Task-dependent Distinctions

The example above has confronted us with the problem that simply picking model fragments from a library and composing the model is not enough. It is infeasible, in general, to anticipate the required granularity in the domains of variables. Therefore, the ability to *transform* the domains to the right level of abstraction *after* composing the constraints of the model is a highly practical requirement. It means grouping

together those domain values whose distinction is irrelevant for the task at hand.

The core idea of distinctions between domain values being redundant is captured by the concept of interchangeability, first proposed by Freuder ([Freuder, 1991]). For a constraint satisfaction problem that consists of a set of variables, domains and constraints on these variables, two values val_1 , val_2 of a variable v are said to be fully interchangeable, if for any solution where $v = val_1$, substituting $v = val_2$ produces another solution, and vice versa. That is, solutions involving val_1 (val_2) are identical to solutions involving val_2 (val_1) except for the value val_1 (val_2) itself.

Interchangeable values define equivalence classes on the domains of the variables, and grouping them together corresponds to an abstraction of the constraint satisfaction problem that exactly preserves the set of its solutions. Freuder and Sabin ([Freuder, 1991], [Freuder and Sabin, 1995]) already observe that interchangeability is related to abstraction and the formation of “semantic groupings” within the domains of variables. However, it is also known that in practice, interchangeability of domain values does not occur very frequently.

A model-based problem solving task, such as behavioral prediction or diagnosis, can be cast as an instance of a constraint satisfaction problem. However, it is particular in two respects:

- (1) the *input* consists not only of the model, but also of the observations that it is confronted with, such as measurements, hypothetical situations, etc. Typically, observations are restricted because not all of the variables in the device model are observable, or because domain values cannot be observed beyond a certain granularity.
- (2) the *output* involves not all feasible assignments of domain values to variables, but only certain aspects of the solutions are required. Typically, we might be interested in knowing whether values remain below or exceed a certain threshold, or what the values of mode variables of components are (for a diagnostic task), etc.

The idea pursued in this paper is to generalize the basic principle of interchangeability, leveraging on this specific context of a model-based problem solving task. The notion of a task is captured as (1) *observable distinctions* that express what inputs to the problem solving process (for example, observations) can occur, and (2) *target distinctions* that express what aspects of the outcome we are after. They can be exploited to obtain so-called *induced abstractions* — domain abstractions that go beyond the low, generic level of interchangeability, but are still adequate for the given task.

We pursue the approach in the context of general, relational models that are not limited to restricted cases such as linear relationships or monotonic functions. A relational (or constraint-based) behavior model is a subset

$$R \subseteq DOM(v) = DOM(v_1) \times DOM(v_2) \times \dots \times DOM(v_n)$$

that restricts the possible combinations of values for the variables $v = (v_1, v_2, \dots, v_n)$, where $DOM(v_i)$ denotes the domain of a variable v_i . We use join ($\bowtie$), projection (Π), and selection (σ) as operators on relations.

Consider again the device shown in figure 1. Assume that the domain is

$$\{[0V,2V), [2V,4V), [4V,6V), [6V,8V), [8V,10V)\}$$

for variables involving voltage and

$$\{0\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$$

for variables involving position, and that the only parameter in the system, $pos_{switching}$, equals 40%. Then the device can be modeled by a relation R that consists of 10 tuples:

v_{pot}	$position$	v_{switch}	...
[0V,2V)	0%, 20%	[0V,2V)	...
[2V,4V)	20%, 40%	[0V,2V)	...
[4V,6V)	40%	[0V,2V)	...
[4V,6V)	60%	[8V,10V)	...
[6V,8V)	60%, 80%	[8V,10V)	...
[8V,10V)	80%, 100%	[8V,10V)	...

Observable distinctions reflect the measurement granularity or incomplete observability of variables. An observable distinction for a variable is expressed as a partition of its domain:

Definition 1 (Observable Distinction) An observable distinction for a variable v_i , denoted $\pi_{obs,i}$, is a partition of its domain $DOM(v_i)$.

A variable v_i is not observable at all if $\pi_{obs,i}$ is equal to the trivial domain partition $\pi_{triv,i} := \{DOM(v_i)\}$. For instance, the fact that for the pedal position sensor, the control unit observes the signal v_{pot} and the signal v_{switch} can be expressed by the observable distinction

$$\begin{aligned} \pi_{obs,v_{pot}} &= \{\{[0V,2V)\}, \dots, \{[8V,10V)\}\}, \\ \pi_{obs,v_{switch}} &= \{\{[0V,2V)\}, \dots, \{[8V,10V)\}\}, \end{aligned}$$

whereas all other variables receive the trivial domain partition.

Target distinctions reflect the granularity of solutions we are after. Analogously to observable distinctions, target distinctions are expressed as domain partitions:

Definition 2 (Target Distinction) A target distinction for a variable v_i , denoted $\pi_{tar,i}$, is a partition of its domain $DOM(v_i)$.

A variable v_i is said to have no target partition, if $\pi_{tar,i}$ is equal to the trivial partition. For instance, the target distinctions for the pedal position sensor are determined by the goal to distinguish between the domain values for the variable v_{switch} (the plausibility check itself is not represented in the model):

$$\pi_{tar,v_{switch}} = \{\{[0V,2V)\}, \dots, \{[8V,10V)\}\}.$$

2.1 Domain Abstractions

A domain partition π_i can also be understood as a domain abstraction

$$\tau_i : DOM(v_i) \rightarrow DOM'(v_i) \subseteq 2^{DOM(v_i)}$$

that maps elements val from a base domain $DOM(v_i)$ to a transformed domain $DOM'(v_i)$ that consists of sets of values from the base domain such that $val \in \tau_i(val)$. Abstractions

can be straightforwardly extended from single values to sets of values, by taking the union of the resulting sets. For two abstractions τ_i and τ'_i , τ_i is a *refinement* of τ'_i , if

$$\forall val \in DOM(v_i) : \tau_i(val) \subseteq \tau'_i(val).$$

Two abstractions τ_i, τ'_i are *comparable*, if τ_i is a refinement of τ'_i or τ'_i is a refinement of τ_i . We apply the notion of refinement and comparability equally to abstractions and domains. A vector of domain abstractions is denoted

$$\tau = (\tau_1, \tau_2, \dots, \tau_n).$$

A domain abstraction τ is a refinement of τ' , if every τ_i is a refinement of τ'_i . Two abstractions τ, τ' are *piecewise comparable*, if every τ_i is comparable with τ'_i . If a join operation combines two relations that are defined on different, but piecewise comparable domains, we define, for convenience, that the result is a relation over the finer domains.

3 Qualitative Abstraction Problems

Given this representational apparatus, we can now formally define the problem of task-dependent qualitative abstraction.

Definition 3 (Qualitative Abstraction Problem) Let R be a relational behavior model, OBS a set of external restrictions, τ_{obs} a domain abstraction defined by observable distinctions, and τ_{tar} a domain abstraction defined by target distinctions. The qualitative abstraction problem consists of finding an induced domain abstraction τ_{ind} such that:

- (1) (Adequacy) For all external restrictions $R_{obs} \in OBS$, $R_{obs} \subseteq DOM(v)$,
$$\begin{aligned} &\tau_{tar}(R \bowtie \tau_{obs}(R_{obs})) = \\ &\tau_{tar}(\tau_{ind}(R) \bowtie \tau_{ind}(\tau_{obs}(R_{obs}))). \end{aligned}$$
- (2) (Simplicity) If τ_{ind} is a refinement of a domain abstraction τ_{ind}' and τ_{ind}' fulfills (1), then $\tau_{ind} = \tau_{ind}'$.

The first condition (adequacy) states that the abstracted model $\tau_{ind}(R)$ derives a solution on the level of target distinctions, if and only if the original model R derives the same solution on the level of target distinctions. We require this to hold for all the possible external restrictions (actual observations, design specifications, etc.) on the level of observable distinctions. This guarantees that for any external restriction that can occur, the abstracted model will yield the same results as the original model. That is, if we apply τ_{ind} before carrying out our problem-solving task, it won't affect the result because this abstraction incorporates all the distinctions that are necessary for this task. As a consequence, we can substitute the abstracted model $\tau_{ind}(R)$ for the original model R in problem solving.

In general, there may be many domain abstractions that fulfill the adequacy criterion. In particular, the identical domain abstraction τ_{id} that retains all the distinctions in the model is an adequate abstraction according to the definition. However, among all adequate abstractions, we prefer those that are the "simplest" ones. Simplicity of an adequate abstraction is defined in the second condition of definition 3. The approach taken is to select the abstractions that are coarsest in the sense that there exists no other adequate abstraction of

which they would be a strict refinement (an abstraction that would further merge at least two of the qualitative values). A domain abstraction that is both adequate and simple incorporates only distinctions that are both necessary and sufficient according to the target and observable distinctions. It represents a level of abstraction that neither makes any unnecessary distinctions, nor abstracts away any distinctions that are crucial to solve the problem. Definition 3 thus formalizes the problem of finding qualitative values for the domains of variables. Compared to interchangeability, which is concerned with possibilities for abstraction within a single problem instance only, a qualitative abstraction problem (QAP) describes a whole class of instances defined by a model relation, a set of external restrictions, and the task-dependent distinctions. Interchangeability enforces that the set of solutions remains the same as for the original model. A QAP generalizes (or relaxes) this basic principle and demands that the set of solutions remains the same but only on the level of target distinctions, and only for inputs on the level of observable distinctions.

Definition 4 (Properties of QAP) A QAP with a set of external restrictions OBS is said to be *obs-complete*, if $\{\tau_{obs}(R_{obs}) \mid R_{obs} \in OBS\} = 2^{\tau_{obs}(DOM(v))}$. It is said to be *sol-complete*, if for all v_i : $\tau_{target,i}(DOM(v_i)) \subseteq \Pi_i(\{\tau_{target}(R \bowtie \tau_{obs}(R_{obs})) \mid R_{obs} \in OBS\})$.

Obs-completeness means that all the possible observations on the level of observable distinctions can actually occur or have to be considered during problem-solving (consequently, induced abstractions can be derived without knowing the exact set OBS). Sol-completeness means that all the possible solutions defined by the target distinctions can indeed be distinguished based on the model and the external restrictions. In addition, we demand that τ_{obs} and τ_{target} are piecewise comparable. Note that this is not actually a restriction, because it can be established for any QAP by possibly introducing additional variables that separate the target and observable distinctions.

Intuitively, under these conditions, we expect that we have to keep all the target distinctions, because we need them to distinguish the solutions, but we can eliminate the distinctions between observations that would lead to the same set of solutions. If QAP is obs-complete, the $\tau_{obs}(R_{obs})$ are given by the possible subsets of $\tau_{obs}(DOM(v))$. For each tuple $\mathbf{val}_{obs,j} \in \tau_{obs}(DOM(v))$, define $R_{sol,obs,j}$ to be the solution it derives on the level of target distinctions:

$$R_{sol,obs,j} := \tau_{target}(R \bowtie \mathbf{val}_{obs,j}).$$

Let $R_{obs,k}$ denote the sets of $\mathbf{val}_{obs,j}$ that obtain the same solution, i.e. those for which $R_{sol,obs,j}$ is equal:

$$R_{obs,k} := \bigcup_{j: R_{sol,obs,j} = R_{sol,obs,k}} \mathbf{val}_{obs,j}.$$

Then the $R_{obs,k}$ form the elements of a partition of $\tau_{obs}(DOM(v))$:

$$\Sigma(R, \tau_{obs}, \tau_{target}) := \bigcup_k \{R_{obs,k}\}.$$

Figure 2 shows the resulting partition $\Sigma(R, \tau_{obs}, \tau_{target})$ for the pedal position sensor example, given the observable and target distinctions stated in Sec. 2 (all variables except v_{pot} and v_{switch} have no distinction and have been omitted from the figure; values $[2V, 4V]$, $[4V, 6V]$ and $[6V, 8V]$ for v_{switch} do not appear in R and have been omitted from its domain).

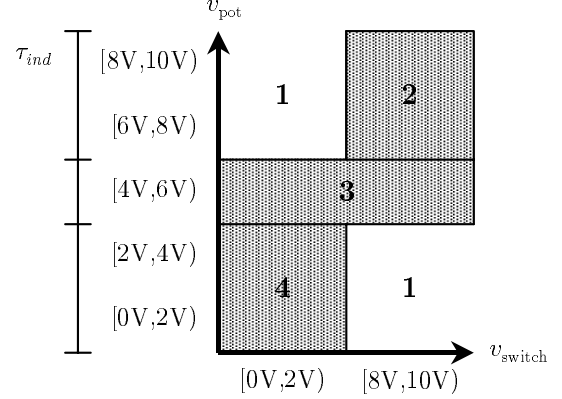


Figure 2: Partition $\Sigma(R, \tau_{obs}, \tau_{target})$ for the pedal position sensor example. The partition consists of four elements numbered 1, 2, ..., 4. Applying theorem 1 yields the distinctions for v_{pot} shown on the left-hand side of the figure.

Theorem 1 (Solution to QAP) Let QAP be a qualitative abstraction problem that is obs-complete and sol-complete. Let $\tau_{FI,\Lambda,i}$ be the domain abstraction that aggregates the interchangeable values of a relation Λ , that is, two values $val_1, val_2 \in DOM(v_i)$ are combined if

$$\Pi_{v \setminus \{v_i\}}(\sigma_{v_i=val_1}(\Lambda)) = \Pi_{v \setminus \{v_i\}}(\sigma_{v_i=val_2}(\Lambda)).$$

Then the simplest domain abstraction that is a refinement of $\tau_{target,i}$ and every domain abstraction

$$\tau_{FI,\Lambda,i} \text{ where } \Lambda \in \Sigma(R, \tau_{obs}, \tau_{target})$$

is an induced abstraction for QAP.

For the pedal position sensor example, Theorem 1 derives three qualitative values for v_{pot} (see Fig. 2):

$$\{\{[0V, 2V), [2V, 4V)\}, \{[4V, 6V)\}, \{[6V, 8V), [8V, 10V)\}\}.$$

The first qualitative value $\{[0V, 2V), [2V, 4V)\}$ corresponds to situations where v_{switch} equals ground voltage, the third qualitative value $\{[6V, 8V), [8V, 10V)\}$ corresponds to situations where v_{switch} equals battery voltage, and the second qualitative value $\{[4V, 6V)\}$ corresponds to situations where the position of the switch and, hence, the voltage of v_{switch} , is ambiguous.

Theorem 1 shows that the basic concept of interchangeability plays a central role in the determination of solutions to a qualitative abstraction problem. In particular, the problem of finding interchangeable values in a relation can be recast as a special case of a QAP, where one distinguishes only empty from non-empty solutions:

Corollary 1 (Interchangeability as QAP) Let $QAP = (R, \tau_{obs}, \tau_{targ})$ be an obs-complete qualitative abstraction problem such that $\tau_{obs} = \tau_{id}$, $\tau_{targ} = \tau_{triv}$. Then $\tau_{FI, R, i}$ is an induced abstraction for QAP.

In general, however, the granularity of induced abstractions is different from the granularity of interchangeable values. τ_{ind} can be either more coarse or more fine-grained than $\tau_{FI, R}$. The latter case occurs if target distinctions are specified between domain values that are interchangeable with respect to the model relation.

Theorem 1 constitutes also a starting point for finding useful approximations of qualitative values. One approach is to use only necessary conditions for interchangeability. In order for two domain values to be aggregated by $\tau_{FI, \Lambda}$, it is necessary that they appear in the same elements of $\Sigma(R, \tau_{obs}, \tau_{targ})$:

Proposition 1 (Approximate Solution to QAP) Let QAP be a qualitative abstraction problem that is obs-complete and sol-complete. Let $\tau_{app, i}$ be the simplest domain abstraction that is a refinement of $\tau_{targ, i}$ and every domain abstraction

$$\tau_{FI, \Lambda', i} \text{ where } \Lambda' := \Pi_{v_i}(\Lambda), \Lambda \in \Sigma(R, \tau_{obs}, \tau_{targ}).$$

Then the induced abstractions are a refinement of $\tau_{app, i}$.

The approximation τ_{app} considers only the projection of each Λ on the individual variables. This yields a granularity that is simpler than or equal to the induced abstractions. For the pedal position sensor example, the distinctions derived by $\tau_{app, i}$ are identical to those derived by Theorem 1. Obtaining $\tau_{app, i}$ is easier than determining $\tau_{ind, i}$, because it involves only the projection and intersection of sets and does not require to determine the interchangeable values in Λ . However, because it considers only restrictions for individual variables, the approximation τ_{app} is not adequate. In general, a restriction might lead to a different solution only if it is combined with additional restrictions for the other variables.

4 AQUA: A Prototypic System for Task-dependent Domain Abstraction

The computation of induced abstractions for a QAP involves, based on the results above, the subproblems of constructing the model relation R , computing the partition $\Sigma(R, \tau_{obs}, \tau_{targ})$, and determining interchangeable values within the elements of this partition.

Our prototypic system AQUA (Automated Qualitative Abstraction) [Sachenbacher, 2001] determines the relation R through structural decomposition of the constraint network that is defined by the set of model fragments it is composed of. The basic principle of structural decomposition ([Gottlob *et al.*, 2000]) is to transform a constraint network into an equivalent acyclic (tree-structured) instance. AQUA then iterates over the partition elements of the observable and target distinctions and labels the tuples of the relation R that are consistent with the respective partition elements. Since directional arc consistency is sufficient for establishing consistency in a tree-structured network ([Dechter and Pearl, 1988]), this step can be performed efficiently by local constraint propagation. This step yields the partition

$\Sigma(R, \tau_{obs}, \tau_{targ})$. Interchangeable values in the partition elements of $\Sigma(R, \tau_{obs}, \tau_{targ})$ can then be found using the basic algorithm described in [Freuder, 1991]. Alternatively, the partition elements can be projected on the individual variables to obtain the approximate solution. AQUA also performs further optimizations in that it can automatically remove redundant values (domain values that do not appear in any constraint) and eliminate variables that have no distinction at all. The decomposition step is independent of the particular task in terms of observable and target distinctions; hence, the resulting tree can be re-used for different combinations of observable or target distinctions.

AQUA builds on components of an existing model-based reasoning framework that consists of a development system for composing a device model from of a library of model fragments, and a runtime system for performing behavioral prediction and diagnosis based on actual measurements for the device. Using AQUA, several tasks can be supported in the context of building model-based systems that are often carried out manually or solved on an ad hoc-basis. The common theoretical basis is to find suitable domains for the variables in a model. However, in different contexts this basic task can have different interpretations, depending on what the terms variable and domain refer to, including magnitudes, modes of components, and deviations from reference behaviors ([Sachenbacher, 2001]).

In a larger, real-world application taken from the automotive domain [Sachenbacher *et al.*, 2000], AQUA was used to obtain abstractions of behavior models of a turbo control subsystem that involved several hundred (initially, real-valued) variables. The qualitative level of abstraction of the derived model was instrumental to meet real-time requirements, and allowed for keeping up with the rate of the measurements in an on-board diagnostic context. In this context, task-dependent abstraction proved useful as a precompilation method that guaranteed adequacy of the results.

5 Discussion

While several pieces of work have addressed the problem of automatically deriving appropriate models ([Nayak, 1994; Levy *et al.*, 1997; Rickel and Porter, 1997]), the work presented here is distinctive in that it focuses specifically on the granularity (resolution) of the domain values, and uses relations and partitions as a common and concise representational framework. This has several important implications.

The narrowed focus allows us to represent the space of possible candidate models implicitly and compactly as the space of possible domain partitions. In contrast, [Nayak, 1994; Rickel and Porter, 1997] define this space as the possible combinations of model fragments. Candidate models are then defined by different choices (selections) of possible fragments. Because in our framework, the difference between different candidate models is well-defined (by means of operators τ_i), we can in contrast take on the view of transforming (re-formulating) models instead of selecting them. The general relational representation subsumes both infinite and finite constraints, and is not limited to specific types of constraints and special-purpose reasoning methods. It allows

us to capture the conditions for a solution in a single, concise formula (Definition 3) and, more importantly, allows to determine the solutions analytically and in closed form (Theorem 1). In contrast, [Nayak, 1994; Levy *et al.*, 1997; Rickel and Porter, 1997] all devise search procedures that start from an initial model and backtrack until they find a solution. In general, it can be said that our limited scope allows us to assume a less knowledge-based, more mathematical view of automated modeling.

QSIM ([Kuipers, 1986]), a system for performing qualitative simulation of device behavior over time, incorporates methods for refining the domains of variables by deriving new distinctions (“landmarks”) during the simulation process. However, except for signs, the mapping of qualitative values to their base domain is only partially known, since only information on the ordinal relationship and knowledge about values that must be assumed simultaneously (“corresponding values”) is provided. Therefore, the derived distinctions can in general not be used to simplify the model. Extensions of QSIM that deal with semi-quantitative reasoning [Berleant and Kuipers, 1997] allow to further constrain the landmark values to numeric intervals, but are specific to the context of simulation, and the constraints are limited to algebraic relationships and monotonic functions. For the specific task of diagnosis, Torasso and Torta [Torasso and Torta, 2002] recently presented an approach for merging together behavior modes that are indistinguishable, given the granularity of the observations. However, the method does not incorporate a notion of target distinctions.

While the theory of task-dependent domain abstraction is applicable both to finite and infinite domains, an important area for future work is to efficiently derive induced abstractions for real-valued base models. [Sachenbacher, 2001] outlines a method for iterative refinement of qualitative values given a real-valued base model. [Struss, 2002] investigates cases where distinctions can be obtained for monotonic regions of real-valued functions.

6 Conclusion

The increasing complexity of engineered devices has lead to an increased demand for computer-supported behavior prediction, diagnosis, and testing. Given the maturity and scale of model-based systems applications, the question of how to re-use behavior models is of growing interest. It has been shown that a model composed from a library cannot be expected to have a level of granularity suitable for different tasks right away. Instead, the ability to re-formulate the model after composing it is a crucial requirement. We identified, within a common relational framework, fundamental properties of re-formulation that is based on abstraction of domain values. Observable distinctions and target distinctions allow to capture important aspects of a task. They are the starting point for deriving qualitative values that are both adequate for the task and as simple as possible. Our analysis reveals that the degree of domain abstraction that can be achieved is strongly dependent on the characteristics of the task. Task-dependent qualitative domain abstraction is a contribution to further bridging the gap between quantitative and

qualitative modeling, as it allows to express knowledge about component behavior without being committed early to a specific abstraction level of the domains. It can help to make model-based system more efficient and more cost-effective due to automating steps that are currently done by hand.

References

- [Berleant and Kuipers, 1997] D. Berleant and B. Kuipers. Qualitative and quantitative simulation: Bridging the gap. *Artificial Intelligence*, 95(2):215–255, 1997.
- [Dechter and Pearl, 1988] Rina Dechter and Judea Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1988.
- [Falkenhainer and Forbus, 1991] B. Falkenhainer and K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.
- [Freuder and Sabin, 1995] E. C. Freuder and D. Sabin. Interchangeability supports abstraction and reformulation for constraint satisfaction. In *Proceedings of SARA'95*, 1995.
- [Freuder, 1991] E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. of AAAI-91*, pages 227–233, Anaheim, CA, 1991.
- [Gottlob *et al.*, 2000] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, 2000.
- [Hamscher *et al.*, 1992] W. Hamscher, L. Console, and J. de Kleer, editors. *Readings in Model-based Diagnosis*. Morgan Kaufman Publishers, San Mateo, CA, 1992.
- [Kuipers, 1986] B. J. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.
- [Levy *et al.*, 1997] A. Levy, Y. Iwasaki, and R. Fikes. Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence*, 96:351–394, 1997.
- [Nayak, 1994] P. Pandurang Nayak. Causal approximations. *Artificial Intelligence*, 70:277–334, 1994.
- [Rickel and Porter, 1997] J. Rickel and B. Porter. Automated modeling of complex systems to answer prediction questions. *Artificial Intelligence*, 93:201–260, 1997.
- [Sachenbacher *et al.*, 2000] M. Sachenbacher, P. Struss, and C. Carlen. A prototype for model-based on-board diagnosis of automotive systems. *AI Comm.*, 13(2):83–97, 2000.
- [Sachenbacher, 2001] M. Sachenbacher. *Automated Qualitative Abstraction and its Application to Automotive Systems*. PhD thesis, TU Munich, 2001.
- [Struss, 2002] P. Struss. Automated abstraction of numerical simulation models: Theory and practical experience. *Proc. QR-02, Spain*, 2002.
- [Torasso and Torta, 2002] P. Torasso and G. Torta. Merging indiscriminable diagnoses: An approach based on automatic domains abstraction. *Proc. DX-02, Austria*, 2002.
- [Weld and de Kleer, 1990] D. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, CA, 1990.

Coupling CSP Decomposition Methods and Diagnosis Algorithms for Tree-Structured Systems*

Markus Stumptner

University of South Australia,
Advanced Computing Research Centre
5095 Mawson Lakes SA, Adelaide, Australia
mst@cs.unisa.edu.au

Franz Wotawa[†]

Technische Universität Graz
Institute for Software Technology (IST)
8010 Graz, Inffeldgasse 16b/2, Austria
wotawa@ist.tugraz.at

Abstract

Decomposition methods are used to convert general constraint satisfaction problems into an equivalent tree-structured problem that can be solved more effectively. Recently, diagnosis algorithms for tree-structured systems have been introduced, but the prerequisites of coupling these algorithms to the outcome of decomposition methods have not been analyzed in detail, thus limiting their diagnostic applicability. In this paper we generalize the TREE* algorithm and show how to use hypertree decomposition outcomes as input to the algorithm to compute the diagnoses of a general diagnosis problem.

1 Introduction

The development of effective algorithms for diagnosing large and complex systems remains one of the key issues in model-based reasoning. Nonetheless, apart from various additional optimizations and control strategies (e.g., [de Kleer, 1991]) the main architectures for consistency-based diagnosis systems, TMS [de Kleer and Williams, 1987] and hitting sets computation [Reiter, 1987], have remained remarkably stable since the late 1980s. Over the last years though, a number of algorithms were published that exploited advantageous structural properties of the systems to be diagnosed for significant computational speedups [Fattah and Dechter, 1995; Stumptner and Wotawa, 2001], enabling the very fast diagnosis of large tree-structured systems.

It was recognized early on that the combination of tree-oriented algorithms could in principle be combined with a second class of algorithms aiming at the decomposition of cyclic problems to equivalent tree-structured problems so that the faster solution algorithms could be brought to bear [Fattah and Dechter, 1995; Darwiche, 1998]. Again, this direction profits from recent new results relating the different decomposition algorithms [Gottlob *et al.*, 2000] and analyzing their performance [Gottlob *et al.*, 2002]. This paper joins the two strands by tying the link between the recent work on problem decomposition and the TREE* algorithm [Stumptner and Wotawa, 2001].

*This work was partially supported by the Austrian Science Fund (FWF) under project grants P15163-INF and P15265-INF.

[†]Authors are listed in alphabetical order.

This combination is aided by the fact that both subareas tend to use Constraint Satisfaction Problems (CSPs) as their representation of choice. The diagnosis computation work [Fattah and Dechter, 1995; Stumptner and Wotawa, 2001; Mauss and Tatar, 2002] focuses on the relational combination of the different constraint relations, while the problem reformulation work views CSPs as (often cyclic) hypergraphs that are broken down into tree structures.

A CSP (V, D, C) comprises a set of variables V , their domains D , and a set of constraints C . A constraint C_i is tuple (S_i, T_i) where $S_i \subseteq V$ is its scope and T_i is a set of tuples of values for the variables in S_i . We assume two functions tpl and $scope$ on constraints that return the set of tuples and the scope respectively. Each constraint C_i restricts the possible values of the variables with respect to values of other variables in the same scope. For example, the digital circuit from Figure 1 can be represented by the following constraints¹:

N_1			N_2		
a	f	g	f	b	h
1	1	1	0	0	0
0	0	0	0	1	1
			1	0	1
			1	1	1

N_3			N_4		
g	c	i	h	i	j
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1

A solution of a CSP is an assignment of values to all variables that satisfy the given constraints. All values must be elements of their variable's domain. The variable assignment $b = 1, a = 0, c = 0, f = 0, g = 0, h = 1, i = 0, j = 1$ is a solution for the above CSP whereas $b = 1, a = 0, c = 0, f = 0, g = 0, h = 1, i = 0, j = 0$ is not because constraint N_4 is not satisfied.

A standard approach for computing solutions for CSPs are backtracking algorithms, which are in the worst case exponential in the number of variables. However, for a specific

¹Note that not all constraints are components, N_1 is the tabular representation of an equality constraint between connections).

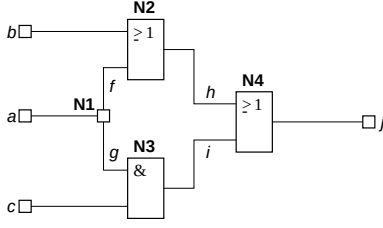


Figure 1: A famous small combinational circuit

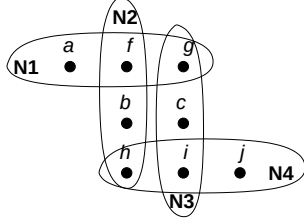


Figure 2: A hypergraph representing the constraints

class of CSPs a solution can be computed in polynomial time. This class comprises all CSPs that have an acyclic corresponding hypergraph². A hypergraph of a CSP can be easily constructed by mapping all variables of the CSP to vertices and the constraint scopes to hyperedges. A CSP with an acyclic hypergraph can be solved effectively in a backtrack-free manner by first traversing the graph from the leaves to the root and computing possible value tuples and secondly, traversing the graph from the root to the leaves and selecting one tuple of a node as a solution. The hypergraph corresponding to Fig. 1 is cyclic (see Figure 2).

In the rest of the paper, we recapitulate decomposition methods, present a version of TREE* that fits these methods, show the interaction of decomposition and TREE*, and present an extension to the algorithm that can be used with extended domains as presented in [Mauss and Tatar, 2002].

2 Decomposition methods

Several different decomposition methods have been published, most recently the hypertree decomposition method [Gottlob *et al.*, 1999a]. This and other structure-based decomposition methods make compute a tree-structured systems from general CSPs by elimination of cycles from the CSP. A cycle can be eliminated by applying the relational algebra join operation to the constraints on the cycle. However, blindly joining all constraints in the cycle can result in exponential costs for computing the tuples of the joined constraint, and a key property of the different decomposition methods is the different techniques they use to select the constraints to be joined. Gottlob *et al.* [2000] compared different decomposition methods with respect to their width, i.e., the maximum number of constraints to be joined, and found that hypertree decomposition is superior with respect to the width. Since hypertree decomposition can be seen as a generalization of the

other techniques, we restrict our examination here to hypertree decomposition. Note that the result of a decomposition method is always a hypertree.

In [Gottlob *et al.*, 2000] hypertree decomposition is characterized as follows. A hypertree of a hypergraph H is a triple (T, χ, λ) , where $T = (N, E)$ is a rooted tree with vertices N and edges E , χ and λ are labeling functions which associate to each vertex $p \in N$ a set of variables $\lambda(p) \subseteq \text{var}(H)$ and a set of constraints $\chi(p) \subseteq \text{edges}(H)$. We further define χ for a subtree $T' = (N', E')$ of T as $\chi(T') = \bigcup_{p \in N'} \chi(p)$, and for any $p \in N$, T_p denotes the subtree of T rooted at p . We denote the root of a hypertree by $\text{root}(T)$.

Based on the above definitions, the hypertree decomposition of a hypergraph H is defined as a hypertree $HD = (T, \chi, \lambda)$ where $T = (N, E)$ which satisfies the following conditions:

1. For each edge $h \in \text{edges}(H)$, there exists a $p \in N$ such that $\text{var}(h) \subseteq \chi(p)$.
2. For each variable $Y \in \text{var}(H)$, the set $\{p \in N \mid Y \in \chi(p)\}$ induces a connected subtree of T .
3. For each $p \in N$, $\chi(p) \subseteq \text{var}(\lambda(p))$.
4. For each $p \in N$, $\text{var}(\lambda(p)) \cap \chi(T_p) \subseteq \chi(p)$.

Moreover, we say that $h \in \text{edges}(H)$ is strongly covered in HD if there exists a vertex $p \in N$ such that $\text{var}(h) \subseteq \chi(p)$ and $h \in \lambda(p)$. A hypertree decomposition HD of a hypergraph H is a complete decomposition of H if every edge of H is strongly covered in HD . Note that it is always possible to make an incomplete decomposition complete by adding new vertices to the decomposition. Gottlob *et al.* [1999b] gave an algorithm for computing (complete) hypertree decompositions.

Note that unlike other decomposition methods such as bi-connected component decomposition and hinge decomposition, there is in general no unique hypertree decomposition of a given CSP. Figure 3 shows four possible hypertree decompositions for the hypergraph depicted in Figure 2.

3 Diagnosis with TREE*

To be self contained we briefly recapitulate the TREE* algorithm. Stumptner and Wotawa [2001] introduced the TREE* algorithm as an extension of the TREE algorithm. Both algorithms work on tree-structured constraint systems. As opposed to TREE which requires the constraints to be mathematical functions, TREE* imposes no limitations on the constraints. TREE* uses the following auxiliary functions associated to constraints: *constr* denotes all tuples of the constraint. *val* denotes the tuples remaining after the application of TREE*. *diags* denotes the diagnoses that correspond to a given tuple. Accordingly, a CSP that is to be used for diagnosis purpose has to represent not only the tuples for each constraint but also the diagnoses that correspond to each tuple. For example, the small circuit from Figure 1 can be represented by the following CSP:

²A hypergraph (N, E) consists of a set of vertices N and a set of hyperedges E such that $e \subseteq N$ for each $e \in E$

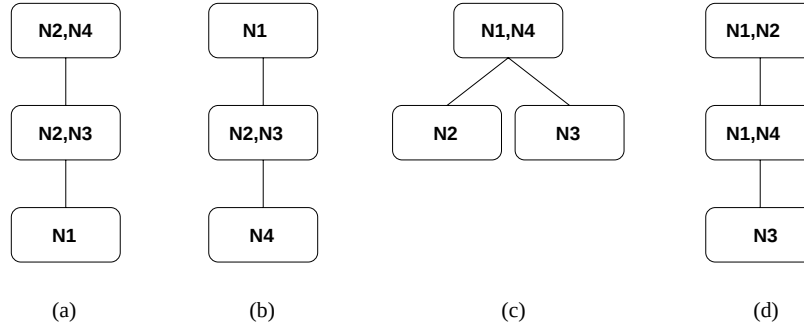


Figure 3: Alternate hypertree decompositions of Figure 2

N_1				N_2			
a	f	g	$diags$	f	b	h	$diags$
1	1	1	$\{\{\}\}$	0	0	0	$\{\{\}\}$
0	0	0	$\{\{\}\}$	0	1	1	$\{\{\}\}$
			$\{\{\}\}$	1	0	1	$\{\{\}\}$
			$\{\{\}\}$	1	1	1	$\{\{\}\}$
			$\{\{\}\}$	x	x	x	$\{\{N_2\}\}$

N_3				N_4			
g	c	i	$diags$	h	i	j	$diags$
0	0	0	$\{\{\}\}$	0	0	0	$\{\{\}\}$
0	1	0	$\{\{\}\}$	0	1	1	$\{\{\}\}$
1	0	0	$\{\{\}\}$	1	0	1	$\{\{\}\}$
1	1	1	$\{\{\}\}$	1	1	1	$\{\{\}\}$
x	x	x	$\{\{N_3\}\}$	x	x	x	$\{\{N_4\}\}$

The tuples of form $\langle x \ x \dots x \rangle$ in the above tables are intended to match all tuples that are not explicitly given in the table. Note that the constraint N_1 represents a connection and is therefore assumed to be always correct. Hence, it always returns the empty set as a diagnosis.

The original description of the TREE* algorithm was based on the underlying assumption that the leaf vertices of the tree correspond only to one diagnosis component. This assumption has an impact on the description of the algorithm, but not on the empirical results. If the constraints given to the algorithm do not satisfy this requirement, diagnoses of a size (i.e., number of faulty components) larger than specified might be returned because there is no way to remove these diagnoses from the set of tuples. The requirement did not constitute a hindrance since [Stumptner and Wotawa, 2001] did not deal with CSPs resulting from decomposition. To explicitly generalize TREE* to interact with decomposition algorithms, we here present a modified version that removes this requirement.

The following operations are used by TREE*: semi-join ($\bowtie$) for constraints, Cartesian product ($\times$) for combining two sets of diagnoses, i.e., $DS_1 \times DS_2 = \{d_1 \cup d_2 \mid d_1 \in DS_1 \wedge d_2 \in DS_2\}$, and cardinality restriction ($|$) for removing diagnoses from a set of diagnoses with a size greater than the given value, i.e., $DS|_s = \{d \mid d \in DS \wedge |d| \leq s\}$.

Algorithm TREE* ($HD, p, diagSize$)

Computes all diagnoses up to a pre-specified size for a given tree-structured diagnosis system.

Input: A decomposition HD with edges E , the root vertex p , and the pre-specified diagnosis size $diagSize$.

Output: The diagnoses for each value tuple.

- (1) $V \leftarrow constr(p) \bowtie observed(p)$
- (2) **for** $s \in V$ **do**
- (3) $diags(s) \leftarrow diags(s)|diagSize$
- (4) **if** $diags(s) = \emptyset$ **then**
- (5) $V \leftarrow V \setminus \{s\}$
- (6) **end if**
- (7) **end for**
- (8) **for** $(p, q) \in E$ **do**
- (9) TREE*(q)
- (10) $V \leftarrow V \bowtie val(q)$
- (11) **for** $s \in V, t \in val(q)$ such that s, t join **do**
- (12) $diags(s) \leftarrow (diags(s) \times diags(t))|diagSize$
- (13) **if** $diags(s) = \emptyset$ **then**
- (14) $V \leftarrow V \setminus \{s\}$
- (15) **end if**
- (16) **end for**
- (17) **end for**
- (18) $val(p) \leftarrow V$

The TREE* algorithm is called using the root p of the hypertree, i.e., the result of a decomposition, as argument. After execution, the computed diagnoses can be found in the $diag$ column of the tuples associated with the root p .

The TREE* algorithm correctly computes all diagnoses up to the given size if the following requirements are fulfilled:

1. Every constraint is at least used in one of tree vertices.
2. The induced subtree for every variable is connected.

Theorem 1 (Correctness) *The TREE* algorithm correctly computes diagnoses up to the pre-specified size.*

Proof. The proof is by induction over the size of the tree.

Base step For each leaf of the tree only lines (1)-(7) and (18) of TREE* are executed. In these lines, all tuples that contradict the given observations and all diagnoses larger than the specified size are removed. Tuples with no corresponding diagnoses are removed. These steps do not prevent TREE* from computing a diagnosis. Hence, all diagnoses (up to the pre-specified size) are computed for leaves.

Hypothesis We assume that TREE* correctly computes diagnoses for trees of size smaller than n .

Induction step We now prove that TREE* correctly computes the diagnoses for trees of size n . The steps (1)-(7) only reduce the number of diagnoses. Tuples that contradict the observations are removed. Diagnoses that are larger than the pre-specified values are removed and tuples with a corresponding empty diagnosis set are also removed because they have no influence on the result. In step (9) the TREE* algorithm is called recursively. Because of our induction hypothesis, the algorithm returns correct diagnoses for each tuple. It remains to prove that the combination of these tuples with the tuple of the current node is done correctly. Since every tuple that joins (line (11)) is considered and since that every diagnosis is combined with every diagnosis of the child vertices (line (12)), this process must lead to a correct result. Hence, TREE* correctly computes the diagnoses in step n as well. $\square$

Using the same arguments as above we can show that TREE* allows for computing all diagnoses providing that all the diagnoses for each vertex of the hypertree are computed. We omit the actual proof here.

Theorem 2 (Completeness) *The TREE* algorithm computes all diagnoses up to the pre-specified size.*

The remaining issue now is to determine the conditions that a decomposition method must satisfy in order to be used with TREE*. Moreover, we examine the computation of the join relation for those tree vertices that comprise more than one constraint.

4 Decomposition and Joining of Constraints

As an example we take the hypertree decomposition result from Figure 3(b). In order to apply TREE*, we first have to compute the tuples for the constraints that occur in one vertex, e.g., $\{N_2, N_3\}$. The tuples can be computed by first joining the constraints, and second computing the diagnoses *diags* for each tuple of the joined constraint by combining the diagnoses associated with the corresponding tuples of the original constraints. The following algorithm computes the join relation for the constraint of a hypertree vertex p .

Algorithm JoinRelation (p)

Computes the join for a hypertree vertex

Input: A hypertree vertex p .

Output: The *constr* and *diags* function of the vertex p .

- (1) $constr(p) \leftarrow c_1 \bowtie \dots \bowtie c_k$ where $\{c_1, \dots, c_k\} = \lambda(p)$.
- (2) **for** $t \in constr(p)$ **do**
- (3) $diags_p(t) \leftarrow \{\{\}\}$
- (4) **for** $c \in \lambda(p)$ **do**
- (5) **if** $scope(c) \subseteq \chi(p)$ **then**
- (6) $diags_p(t) \leftarrow diags_p(t) \times diags_c(\pi_{scope(c)} t)$
- (7) **end if**
- (8) **end for**
- (9) **end for**

In the above algorithm the *diags* function is indexed with the corresponding vertex of the hypertree or the corresponding constraint. The function $\bowtie$ stands for relational join and π for relational projection. Diagnoses are only combined if a constraint is fully captured by the given vertex of the hypertree. Otherwise, it is not considered. Since every constraint must be fully captured by at least one hypertree vertex, no information is lost by this procedure.

For example, the join of constraints N_2 and N_3 would lead to the following constraint $[N_2, N_3]$ with 25 tuples (when retaining the 'x' constants, and of course 64 otherwise):

f	b	h	g	c	i	diags
0	0	0	0	0	0	$\{\{\}\}$
0	0	0	0	1	0	$\{\{\}\}$
0	0	0	1	0	0	$\{\{\}\}$
0	0	0	1	1	1	$\{\{\}\}$
0	0	0	x	x	x	$\{N_3\}$
0	1	1	0	0	0	$\{\{\}\}$
0	1	1	0	1	0	$\{\{\}\}$
0	1	1	1	0	0	$\{\{\}\}$
0	1	1	1	1	1	$\{\{\}\}$
0	1	1	x	x	x	$\{N_3\}$
1	0	1	0	0	0	$\{\{\}\}$
1	0	1	0	1	0	$\{\{\}\}$
1	0	1	1	0	0	$\{\{\}\}$
1	0	1	1	1	1	$\{\{\}\}$
1	0	1	x	x	x	$\{N_3\}$
1	1	1	0	0	0	$\{\{\}\}$
1	1	1	0	1	0	$\{\{\}\}$
1	1	1	1	0	0	$\{\{\}\}$
1	1	1	1	1	1	$\{\{\}\}$
1	1	1	x	x	x	$\{N_3\}$
x	x	x	0	0	0	$\{N_2\}$
x	x	x	0	1	0	$\{N_2\}$
x	x	x	1	0	0	$\{N_2\}$
x	x	x	1	1	1	$\{N_2\}$
x	x	x	x	x	x	$\{N_2, N_3\}$

If, as in this example, the join operation is a Cartesian Product, the resulting relation is of course very large. However, during the computation of diagnoses using TREE*, many tuples can generally be eliminated because of the given observations and the pre-specified maximum diagnosis size.

To illustrate this, assume now that we have a set of observations $OBS = \{a = 1, b = 0, c = 1, j = 0\}$ and that we are searching only for single diagnoses, i.e., $diagSize = 1$. Using the decomposition from Figure 3(b) TREE* is first called with the vertex (constraint) $[N_1]$. After executing lines (1)–(7) (including the semijoin that results in the removal of tuples that do not fit the operations) the constraint of $[N_1]$ is given by:

a	f	g	diags
1	1	1	$\{\{\}\}$

TREE* is then recursively called on vertex $[N_2, N_3]$, resulting in the following relation, which is substantially smaller than the computed join relation for this vertex, even after all x entries of the original tables have been replaced by either 0 or 1, while avoiding duplicated entries (first table), then we again recursively call TREE* which leads to the computation of the following relation for vertex $[N_4]$ (second table):

f	b	h	g	c	i	diags
0	0	0	0	1	0	$\{\{\}\}$
0	0	0	1	1	1	$\{\{\}\}$
0	0	0	0	1	1	$\{\{N_3\}\}$
0	0	0	1	1	0	$\{\{N_3\}\}$
1	0	1	0	1	0	$\{\{\}\}$
1	0	1	1	1	1	$\{\{\}\}$
1	0	1	0	1	1	$\{\{N_3\}\}$
1	0	1	1	1	0	$\{\{N_3\}\}$
0	0	1	0	1	0	$\{\{N_2\}\}$
1	0	0	0	1	0	$\{\{N_2\}\}$
0	0	1	1	1	1	$\{\{N_2\}\}$
1	0	0	1	1	1	$\{\{N_2\}\}$

h	i	j	diags
0	0	0	$\{\{\}\}$
0	1	0	$\{\{N_4\}\}$
1	0	0	$\{\{N_4\}\}$
1	1	0	$\{\{N_4\}\}$

In the next step TREE* continues the computation at line (10) for vertex $[N_2, N_3]$. After combining the diagnoses and removing the tuples with diagnoses larger than $diagSize$ we get the following relation for $[N_2, N_3]$:

f	b	h	g	c	i	diags
0	0	0	0	1	0	$\{\{\}\}$
0	0	0	1	1	1	$\{\{N_4\}\}$
0	0	0	1	1	0	$\{\{N_3\}\}$
1	0	1	0	1	0	$\{\{N_4\}\}$
1	0	1	1	1	1	$\{\{N_4\}\}$
1	0	0	0	1	0	$\{\{N_2\}\}$

And finally for the vertex $[N_1]$:

a	f	g	diags
1	1	1	$\{\{N_4\}\}$

Hence, only the diagnosis $\{N_4\}$ is a single diagnosis for our example.

5 Putting it all together

In order to make use of TREE* for general CSPs, we first apply a decomposition method and then apply the algorithm to the resulting acyclic problem. In this section we summarize the requirements placed on the decomposition method.

Theorem 3 *TREE* computes all consistent diagnoses for a given (possibly cyclic) CSP (V, D, C) , if it is applied to the decomposition HD of (V, D, C) that was produced by a decomposition method h with the following properties:*

1. *The decomposition result, i.e., a hypertree HD , must be complete.*
2. *The vertices that use a given variable $Y \in V$ must form a connected subtree of the hypertree HD .*

Proof (sketch): We consider each condition in turn. As discussed in the previous section, the decomposition method must produce a complete decomposition, i.e., every constraint must be strongly covered, as this is a prerequisite for the correct working of the join algorithm. Concerning condition 2, assume that this condition were not satisfied, i.e., that for some Y , $\{p \in N | Y \in \chi(p)\}$ does not induce a connected subtree. This implies either that (if we have no sub-tree of the

resulting hypertree) the decomposition is not cyclic, or that there exists a variable that is used in both sub-trees but not in the parent. In the latter case, the process that computes a solution cannot view the sub-trees as independent problems any more, and TREE* will fail to compute a correct outcome. The other conditions of the hypertree decomposition can be relaxed without affecting the result. If condition 3 is not obeyed, that means some nodes contain variables that are not constrained. This may affect the efficiency of the algorithm (because the node relations include a cartesian product with the values of those variables) but not the correctness. If condition 4 is not obeyed, this means that some variable z such that $z \in \chi(p)$ for a vertex $p \in N$ is not contained in the constraint associated with p , i.e. in relational terms z has been projected away. This means that some subtree of T is going to be less restrictive in execution, leading to excess tuples. However, since the decomposition is required to be complete, all constraints that contain z must exist in unprojected form somewhere in T . $\square$

These two conditions are true for hypertree decomposition [Gottlob *et al.*, 1999a], biconnected components [Freuder, 1985], hinge decomposition [Gyssens *et al.*, 1994], and tree clustering [Dechter and Pearl, 1989].

The following algorithm summarizes the combined use of TREE* and a decomposition method to compute the diagnoses for any CSP $\mathcal{C}$ of the form described in Section 3. Given a CSP $\mathcal{C}$, using any decomposition method that fulfills the properties of Theorem 3.

- (1) apply decomposition to $\mathcal{C}$, i.e., compute a hypertree (N, E) for the corresponding hypergraph of $\mathcal{C}$
- (2) **for** $v \in N$ **do** JoinRelation(v) **end for**;
- (3) TREE*($root(T)$);
- (4) **return** diags($root(T)$);

The pre-compilation performance of the overall algorithm is that of the decomposition algorithm (examined in [Gottlob *et al.*, 2002]) together with the required costs for joining constraints using the JoinRelation algorithm. The diagnosis time is the running time of TREE*. Performance and scalability TREE* have been studied in [Stumptner and Wotawa, 2001]. Note that the TREE* run time depends on the size of the relations that are stored in the vertices of the resulting hypertree. This size depends on the number of constraints that must be joined, and this in turn corresponds to the width of the decomposed system. Hence, using a decomposition method that provides a smaller width leads to hypertrees where TREE* performs better.

6 Extension to Intensional Relations

In [Stumptner and Wotawa, 2001], we mentioned the possibility to use other than extensional relations for constraint specifications for the TREE and TREE* algorithm, e.g., computed functions or equations with infinite domains. Such domains require a different interpretation of the operators that are used for joining the relations associated with nodes in the tree. Conceptually, however, nothing is changed since, as we will show, the definition of the TREE* algorithm fits the requirements. We show this by adopting the notation used for the basic computational operations of the *aggregation paradigm* described in [Mauss and Tatar, 2002].

The Rich Constraint Languages approach described in [Mauss and Tatar, 2002] consists of three inference procedures, which are applied to a set of constraints R . The procedure `isConsistent` produces a proof tree (called *aggregation tree*) that derives consistency or inconsistency. If consistent, then, applied to the root of this tree, the procedure `solve` computes (nondeterministically) a solution that assigns a value to every variable in R . If not, the procedure `XC1` (and its extension `XE1`) computes the set of minimal conflicts for A .

The aggregation operation that gives the tree its name consists of joining two constraints A and B (expressed as relations or equations) and then projecting out all variables except the set of variables X needed for joining to other constraints: $C \leftarrow \pi(A \wedge B, X)$. If $X = \text{var}(A)$, this can be written using the semijoin operator that we have used above: $C \leftarrow A \bowtie B$. Thus, the `TREE*` algorithm can be changed to approximate the aggregation paradigm purely by letting s and t refer to tuples (for extensional constraints) or equations, and changing line (1) to $V \leftarrow \pi(\text{constr}(p) \wedge \text{observed}(p), \text{var}(\text{constr}(p)))$, and line (10) to $V \leftarrow \pi(V \wedge \text{val}(q), \text{var}(\text{val}(q)))$.

The `JoinRelation` Algorithm is changed by replacing the first line by $\text{constr}(p) \leftarrow (\wedge c_i, \text{var}(p))$.

In terms of algorithmic structure, the approach of [Mauss and Tatar, 2002] bears many resemblances to the decompose-and-diagnose paradigm used in [Fattah and Dechter, 1995; Darwiche, 1998; Stumptner and Wotawa, 2001]. Computationally, there are however significant differences. The outcome of `isConsistent` is a proof tree whose leaves are the base constraints of the original CSP, not a hypertree whose nodes are the constraints of a backtrack-free CSP equivalent to the original CSP. The search for a diagnosis consists of running `isConsistent`, computing minimal conflicts, and the hitting sets. In our case, we compute the decomposition hypertree, then apply the `JoinRelation` algorithm and finally apply `TREE*` to compute the diagnoses directly.

While the nondeterministic selection of arbitrary constraints from the given CSP to produce a proof tree is quite effective in general, as indicated by the authors, there are cases where the “width” of the generated problem leads to a drastic growth in intermediate relations. It is this situation where an approach based on hypertree decomposition (which is generally the method with lowest widths) fits best to the strengths of the `TREE*` algorithm.

7 Conclusion

In this paper we introduced a framework that allows for combining various structure decomposition methods and algorithms for solving tree-structured diagnosis problems. The framework comprises two parts. In the first part, we show how to construct a tree-shaped constraint system that can be used directly by `TREE*` for computing diagnoses. For this purpose we introduced the join relation of constraints. In the second part, we state the properties of a decomposition method so that it can be combined with `TREE*`. Finally, we show the suitability of the `TREE*` framework for the extension to infinite domains and intensional constraints (e.g., equations), by adopting a basic operation from another, not hypertree based framework.

References

- [Darwiche, 1998] Adnan Darwiche. Compiling Devices: A Structure-Based Approach. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 156–166, 1998.
- [de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.
- [de Kleer, 1991] Johan de Kleer. Focusing on probable diagnoses. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 842–848, Anaheim, July 1991.
- [Dechter and Pearl, 1989] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.
- [Fattah and Dechter, 1995] Yousri El Fattah and Rina Dechter. Diagnosing tree-decomposable circuits. In *Proceedings 14th International Joint Conf. on Artificial Intelligence*, pages 1742 – 1748, 1995.
- [Freuder, 1985] Eugene C. Freuder. A Sufficient Condition for Backtrack-Bounded Search. *Artificial Intelligence*, 32(4):755–761, 1985.
- [Gottlob et al., 1999a] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree Decomposition and Tractable Queries. In *Proc. 18th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-99)*, pages 21–32, Philadelphia, PA, 1999.
- [Gottlob et al., 1999b] Georg Gottlob, Nicola Leone, and Francesco Scarcello. On Tractable Queries and Constraints. In *Proc. 12th International Conference on Database and Expert Systems Applications DEXA 2001*, Florence, Italy, 1999.
- [Gottlob et al., 2000] Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124(2):243–282, December 2000.
- [Gottlob et al., 2002] Georg Gottlob, Martin Hüttele, and Franz Wotawa. Combining hypertree, bcomp, and hinge decomposition. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02)*, Lyon, July 2002. IOS Press.
- [Gyssens et al., 1994] Marc Gyssens, Peter G. Jeavons, and David A. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66:57–89, 1994.
- [Mauss and Tatar, 2002] Jakob Mauss and Mugur Tatar. Computing minimal conflicts for rich constraint languages. In F. van Harmelen, editor, *Proc. ECAI*, Amsterdam, 2002. IOS Press.
- [Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
- [Stumptner and Wotawa, 2001] Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127(1):1–29, 2001.

Automatic Abstraction in Component-Based Diagnosis Driven by System Observability

Gianluca Torta, Pietro Torasso

Dipartimento di Informatica, Università di Torino (Italy)

e-mail: {torta,torasso}@di.unito.it

Abstract

The paper addresses the problem of automatic abstraction of component variables in the context of Model Based Diagnosis, in order to produce models capable of deriving fewer and more general diagnoses when the current observability of the system is reduced. The notion of indiscriminability among faults of a set of components is introduced and constitutes the basis for a formal definition of admissible abstractions which preserve all the distinctions that are relevant for diagnosis given the current observability of the system. The automatic synthesis of abstract models further restricts abstractions such that the behavior of abstract components is expressed in terms of a simple and intuitive combination of the behavior of their subcomponents. As a validation of our proposal, we present experimental results which show the reduction in the number of diagnoses returned by a diagnostic agent for a space robotic arm.

1 Introduction

System model abstraction has been successfully exploited in many approaches to model-based diagnosis (MBD). The pioneer work of [Mozetič, 1991] and recent improvements proposed e.g. by [Provan, 2001] and [Chittaro and Ranon, 2001] mostly use abstraction in order to focus the diagnostic process and thus improve its efficiency. However, (flexible) abstraction has another main benefit, namely to provide fewer and more concise abstract diagnoses when it is not possible to discriminate among detailed diagnoses. The works by [Console and Theseider Dupré, 1994] and [Friedrich, 1993] accomplish this goal by including abstraction axioms in the Domain Theory and using preference criteria based on the abstraction level of diagnoses.

Recently, some authors have aimed at the same goal in a different way, namely automatic abstraction of the system model ([Sachenbacher and Struss, 2001], [Torasso and Torta, 2002])¹. If the available observables and/or

their granularity are too coarse to distinguish among two or more behavioral modes of a component, or the distinction is not important for the considered system, a system model is automatically generated where such behavioral modes are merged into an abstracted behavioral mode. By using the abstracted model for diagnosis there's no loss of (important) information, while the number of returned diagnoses is reduced, and such diagnoses, by being "as abstract as possible", are more understandable for a human.

The work presented in this paper aims at extending previous works by introducing automatic abstraction of variables (i.e. components) in the presence of a reduced availability of the number and/or granularity of observables. Abstractions based on system observability are particularly relevant in the context of on-board diagnosis. Indeed, it is likely that when a system is operated on-board the only available measures are provided by sensors (which can themselves fail) and taking further measures manually is out of question. Moreover, on-board diagnosis is usually constrained by strict time and resources requirements: using an abstracted system model should yield savings in both the time and the space requirements of the diagnostic process.

Our proposal requires that abstractions do not cause any loss of diagnostic information (e.g. as in the incomplete abstractions discussed in [Autio and Reiter, 1998]) or loss of efficiency (e.g. due to increased fan-in as pointed out in [Provan, 2001]); moreover, we restrict the mapping from abstract components to their subcomponents to be enough simple and intuitive. In order to exclude all the undesired abstractions we introduce a precise definition of *admissible* abstraction, and further restrict the computation of abstractions through *cutoff* criteria which forbid admissible abstractions that may lead to computational inefficiencies.

As a running example to illustrate our definitions and algorithms, we will use throughout the paper the fragment of hydraulic circuit adapted from [Chittaro and Ranon, 2001] depicted in figure 1 (a); table 1 reports its domain theory (valve modes *so* and *sc* abbreviate

components had been proposed in [Out *et al.*, 1994] for simpler models describing only normal behavior of the system

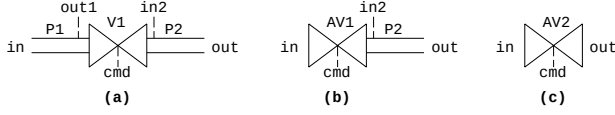


Figure 1: A fragment of an hydraulic circuit at three levels of abstraction

$valve(x) \wedge cmd_x(open) \wedge x(ok) \Rightarrow out_x = in_x$ $valve(x) \wedge cmd_x(open) \wedge x(leak) \Rightarrow out_x < in_x$ $valve(x) \wedge cmd_x(open) \wedge x(so) \Rightarrow out_x = in_x$ $valve(x) \wedge cmd_x(open) \wedge x(sc) \Rightarrow out_x = 0$
$valve(x) \wedge cmd_x(close) \wedge x(ok) \Rightarrow out_x = 0$ $valve(x) \wedge cmd_x(close) \wedge x(leak) \Rightarrow out_x = 0$ $valve(x) \wedge cmd_x(close) \wedge x(so) \Rightarrow out_x = in_x$ $valve(x) \wedge cmd_x(close) \wedge x(sc) \Rightarrow out_x = 0$
$pipe(x) \wedge x(ok) \Rightarrow out_x = in_x$ $pipe(x) \wedge x(br) \Rightarrow out_x = 0$

Table 1: Model of hydraulic components

stuck open and *stuck closed* respectively). Two possible abstractions of such system are shown in figures 1 (b) (where pipe $P1$ and valve $V1$ have been abstracted into valve $AV1$) and 1 (c) (where abstract valve $AV1$ and pipe $P2$ have been abstracted into valve $AV2$). As for experimental validation of our techniques, however, we will present results collected in a larger domain, namely the model of the SPIDER space robotic arm used in [Portinale and Torasso, 1999].

The paper is structured as follows. In section 2 we give a formal definition of what we consider an admissible abstract component. In section 3 we describe how the declarative notions introduced in 2 can be implemented computationally. In section 4 we present experimental results collected in the SPIDER robotic arm domain. Finally, in section 5 we compare our work to related papers and make some concluding remarks.

2 Abstractions Defined

We first report formal definitions of system model, diagnostic problem and diagnosis.

Definition 2.1 A System Description (SD) is a 3-tuple $\langle \mathcal{V}, DT, \mathcal{G} \rangle$ where:

- $\mathcal{V}$ is a set of discrete variables partitioned in the following sorts: CXT (inputs), $COMPS$ (components), $STATES$ (endogenous variables), OBS (observables); $DOM(v)$ is the finite domain of variable $v \in \mathcal{V}$
- DT (Domain Theory) is an acyclic set of Horn clauses defined over $\mathcal{V}$ representing the behavior of the system (under normal and abnormal conditions); we require that, given an instantiation of $COMPS$ and CXT , the DT derives exactly one value for each other variable
- $\mathcal{G}$ (Causal Graph) is a DAG whose nodes are in $\mathcal{V}$ representing the causal structure of the system; whenever a formula $N_1(bm_1) \wedge \dots \wedge N_k(bm_k) \Rightarrow M(bm_i)$ appears in DT , nodes N_1 through N_k are parents of M in the graph

It is worth noting that the fact that $\mathcal{G}$ is restricted to be a DAG does not mean that the system model is a tree in the sense e.g. of [Darwiche, 1998]; since we allow the existence of multiple directed paths between two nodes the associated jointree may well be cyclic. The DAG restriction just forbids feedback loops in the causal graph, which is a common assumption in approaches which deal with dynamic systems only if they are amenable to state-based diagnosis ([Struss, 1997]).

Example 2.1 Note that the System Description for the running sample system given in table 1 is expressed in a slightly different formalism than the one described in our definition, due to the use of equations and disequations on the right-hand side of formulas. However, it can be easily mapped to our formalism by modeling in_x and out_x as discrete *STATES* variables by using qualitative deviations ([Struss et al., 1996]); thus, for example, formula:

$$valve(x) \wedge cmd_x(open) \wedge x(ok) \Rightarrow out_x = in_x$$

is mapped to a set of three formulas:

$$in_x(\langle +, 0 \rangle) \wedge cmd_x(open) \wedge x(ok) \Rightarrow out_x(\langle +, 0 \rangle)$$

$$in_x(\langle +, - \rangle) \wedge cmd_x(open) \wedge x(ok) \Rightarrow out_x(\langle +, - \rangle)$$

$$in_x(\langle 0, - \rangle) \wedge cmd_x(open) \wedge x(ok) \Rightarrow out_x(\langle 0, - \rangle)$$

where $v(\langle [val], [\Delta val] \rangle)$ means that variable v has a value with sign $[val]$ and its deviation from the nominal value has sign $[\Delta val]$. In the rest of the examples we continue to use the notation of [Chittaro and Ranon, 2001] simply because it is more compact. $\square$

Definition 2.2 A diagnostic problem is a 3-tuple $DP = \langle SD, OBS, CXT \rangle$ where SD is the System Description, OBS is an instantiation of OBS and CXT is an instantiation of CXT

Definition 2.3 Given a diagnostic problem $DP = \langle SD, OBS, CXT \rangle$ an instantiation $D = \{C_1(bm_1), \dots, C_n(bm_n)\}$ of $COMPS$ is a diagnosis for DP iff:

$$\forall M(val) \in OBS \quad DT \cup CXT \cup D \vdash M(val)^2$$

As noted in the introduction, there are real scenarios where not all the observables OBS are available and/or their granularity is reduced. We identify the available observables with a set $OBS_{AV} \subseteq OBS$; we also assume that a *granularity mapping* Π is given s.t. $\Pi(M(val))$ maps an instantiation of $M \in OBS_{AV}$ to a possibly more abstract instantiation $M(aval)$. If $\forall M \in OBS_{AV}, \forall val \in DOM(M) : \Pi(M(val)) = M(aval)$ then there is no loss of granularity at all; in this particular case we denote Π with Π_{id} . For example, in figure 1 (a), $OBS_{AV} = OBS = \{out1, in2, out\}$; in 1 (b), $OBS_{AV} = \{in2, out\}$ and in 1 (c), $OBS_{AV} = \{out\}$; as for Π we may consider a situation where $out_x = in_x$ and $out_x < in_x$ have been mapped to a coarser value $out_x \neq 0$. The important point about OBS_{AV} and Π is that these reductions of the observability of the system can cause the model to become less discriminant

²This definition of diagnosis is fully abductive; however results presented in this paper apply equally well to consistency-based diagnosis

and thus different faults of the same component and/or of different components to become indiscriminable. The following definition introduces the notion of indiscriminability among instantiations of subsets of *COMPS*.

Definition 2.4 Let *SCOMPS* be a subset of *COMPS*, *OBS_{AV}* the set of available observables and Π a granularity mapping. We say that two instantiations **SCOMPS1**, **SCOMPS2** of *SCOMPS* are $\langle OBS_{AV}, \Pi \rangle$ -indiscriminable iff for any instantiation **CXT** of *CXT* and any instantiation **OTHERS** of *COMPS* \ *SCOMPS* the following holds ³:

$$\begin{aligned} & \Pi(tclosure_{OBS_{AV}}(\mathbf{OTHERS} \cup \mathbf{SCOMPS1})) = \\ & \Pi(tclosure_{OBS_{AV}}(\mathbf{OTHERS} \cup \mathbf{SCOMPS2})) \end{aligned}$$

Note that the $\langle OBS_{AV}, \Pi \rangle$ -indiscriminability relation induces a partition into $\langle OBS_{AV}, \Pi \rangle$ -indiscriminability classes of the set of possible instantiations of *SCOMPS*. Also note that, when $|SCOMPS| = 1$, indiscriminability coincides with the indistinguishability among behavioral modes introduced in [Torasso and Torta, 2002].

Example 2.2 In the abstraction example of figure 1 (b), under context *cmd* = *open* indiscriminable instances of *P1*, *V1* are grouped in the following sets: $C1 = \{\langle ok, ok \rangle, \langle ok, so \rangle\}$ (which derive $in2 = in$); $C2 = \{\langle ok, leak \rangle\}$ (which derives $in2 < in$); $C3 = \{\langle ok, sc \rangle, \langle br, ok \rangle, \langle br, leak \rangle, \langle br, so \rangle, \langle br, sc \rangle\}$ (which derive $in2 = 0$). By also considering context *cmd* = *close*, set *C1* is further split in two distinct sets: $C11 = \{\langle ok, ok \rangle\}$ (which derives $in2 = 0$) and $C12 = \{\langle ok, so \rangle\}$ (which derives $in2 = in$). If in a specific diagnostic problem we observe, for instance, $in2 = 0$ given *cmd* = *open*, the set of diagnoses is given by *C3* ⁴. $\square$

For now, just note how this notion of indiscriminability is the basis for any potential abstraction. If, indeed, two instantiations **SCOMPS1** and **SCOMPS2** of *SCOMPS* $\subseteq$ *COMPS* are indiscriminable, this means that whenever **SCOMPS1** $\cup$ **OTHERS** is a diagnosis for a given *DP*, **SCOMPS2** $\cup$ **OTHERS** is another, indiscriminable, diagnosis for *DP*.

We now introduce a weak notion of abstraction where, as it is common in structural abstractions, abstract components are recursively built bottom-up starting with the primitive components.

Definition 2.5 Given a set $COMPS = \{C_1, \dots, C_n\}$ of component variables, a components abstraction mapping *AM* of *COMPS* defines a set $COMPS_A = \{AC_1, \dots, AC_m\}$ of discrete variables (abstract components) and associates to each $AC_i \in COMPS_A$ one or more $C_j \in COMPS$ (subcomponents of AC_i) s.t. each component in *COMPS* is the subcomponent of exactly one abstract component. Moreover, *AM* associates, to

³Given an instantiation **COMPS** of *COMPS* we denote with $tclosure_{OBS_{AV}}(\mathbf{COMPS})$ the set of instantiations of variables in *OBS_{AV}* derived from $(DT \cup \mathbf{CXT} \cup \mathbf{COMPS})$

⁴There are $|C3| = 5$ indiscriminable diagnoses; 2 of them, namely $\langle ok, sc \rangle$ and $\langle br, ok \rangle$, can be regarded as preferred diagnoses since they involve only one fault

each abstract component *AC* a definition def_{AC} , which is a characterization of the behavioral modes of *AC* in terms of the behavioral modes of its subcomponents. More precisely, an abstract component and its definition are built hierarchically as follows:

- if $C \in COMPS$, *AC* is a simple abstract component if its definition def_{AC} associates to each $abm \in DOM(AC)$ a formula $def_{abm} = C(bm_1) \vee \dots \vee C(bm_k)$ s.t. $bm_1, \dots, bm_k \in DOM(C)$ ⁵; in the trivial case, *AC* has the same domain as *C* and $\forall bm \in DOM(C) : def_{bm} = C(bm)$
- if AC', AC'' are abstract components with disjoint sets of subcomponents *SCOMPS'*, *SCOMPS''* then *AC* is an abstract component with subcomponents $SCOMPS' \cup SCOMPS''$ if def_{AC} associates to each $abm \in DOM(AC)$ a definition def_{abm} which is a logical formula built by connecting definitions $def_{bm'}, bm' \in DOM(AC')$ with definitions $def_{bm''}, bm'' \in DOM(AC'')$ through $\vee, \wedge$ and $\sim$

The definition def_{AC} of *AC* thus specifies a relation between instantiations of the subcomponents of *AC* and instantiations (i.e. behavioral modes) of *AC* itself. However we need to put some restrictions on these relations in order to match our intuitions about what is an “admissible” abstraction.

Definition 2.6 Given a simple abstract component the definitions associated to its behavioral modes are said to be admissible. For a non-simple abstract component *AC* which is the composition of abstract components AC', AC'' , an admissible definition for $abm \in DOM(AC)$ is defined as follows:

1. conjunction: $def_{abm} = def_{abm'} \wedge def_{abm''}, abm' \in DOM(AC'), abm'' \in DOM(AC'')$ is admissible
2. generalized OR: if $def'_1, \dots, def'_l \in DOM(AC')$ and $def''_1, \dots, def''_m \in DOM(AC'')$ then:
 $def_{abm} = (def'_1 \wedge \sim def'_2 \wedge \dots \wedge \sim def'_l) \vee (def''_1 \wedge \sim def''_2 \wedge \dots \wedge \sim def''_m)$
is admissible. As an important special case, when $l = m = 1$, def_{abm} is a canonical OR

Admissible definitions capture common abstractions, such as the case where the abstract component is *OK* if all its subcomponents are *OK* (conjunction) and faulty if at least one of its subcomponents is faulty (canonical OR). Moreover, since we address the case of components with multiple behavioral modes, we extend canonical OR with exceptions (generalized OR); clearly, the maximum number of exceptions allowed in a generalized OR should be a configurable parameter of the abstraction algorithm.

It is worth noting that the proposed operators, although chosen according to the rationale just exposed, are meant by no means to be the only possible choice in order to

⁵This case has the purpose of including behavioral modes abstraction as described in [Torasso and Torta, 2002] in our framework: $bm_1, \dots, bm_k$ are abstracted in the single behavioral mode *abm*

make our approach to abstraction applicable; if more operators or a different set of operators would fit better particular systems or families of systems, new operators could simply be plugged-in and replace the ones we have defined.

Example 2.3 In the abstraction example of figure 1 (b), the behavioral modes of the abstract component AV1 can be expressed as admissible definitions over primitive components $P1, V1$:

$$\begin{aligned} def_{abm1} &= P1(ok) \wedge V1(ok) \text{ (conjunction)} \\ def_{abm2} &= P1(ok) \wedge V1(leak) \text{ (conjunction)} \\ def_{abm3} &= P1(ok) \wedge V1(so) \text{ (conjunction)} \\ def_{abm4} &= P1(br) \vee V1(sc) \text{ (canonical OR)} \quad \square \end{aligned}$$

Armed with the *admissible definitions* for behavioral modes we can now formally identify the abstraction mappings we are interested in.

Definition 2.7 Given a system model SD , a components abstraction mapping AM of $COMPS$, a set $OBS_{AV} \subseteq OBS$ and a granularity mapping Π , we say that AM is admissible w.r.t. SD, OBS_{AV}, Π iff for each abstract component AC with subcomponents $SCOMPS$:

1. admissible behavioral modes: each $def_k \in def_{AC}$ is admissible in the sense of definition 2.6
2. mutual exclusion: for any two distinct $def_k, def_l \in def_{AC}$, and any instantiation $\mathbf{COMPS}$ of $COMPS$: $\mathbf{COMPS} \cup \{def_k \wedge def_l\} \vdash \perp$
3. completeness: for any instantiation $\mathbf{COMPS}$ of $COMPS$: $\mathbf{COMPS} \vdash \bigvee_{k=1, \dots, |DOM(AC)|} def_k$
4. correctness: given $def_k \in def_{AC}$, the set of instantiations of $SCOMPS$ which satisfy def_k is a $\langle OBS_{AV}, \Pi \rangle$ -indiscriminability class⁶

Example 2.4 The abstraction example of figure 1 (b), is admissible w.r.t. $OBS_{AV} = \{in2, out\}$ and $\Pi \equiv \Pi_{id}$, if the behavioral modes definitions of example 2.3 are used. Such definitions are admissible, moreover there is a 1 : 1 mapping between them and the $\langle OBS_{AV}, \Pi \rangle$ -indiscriminability classes shown in example 2.2; in particular def_{abm1} corresponds to $C11$, def_{abm2} to $C2$, def_{abm3} to $C12$ and def_{abm4} to $C3$. It follows that the mutual exclusion, completeness and correctness conditions are also satisfied $\square$

Note that given an admissible components abstraction mapping AM , to each instantiation $\mathbf{COMPS}$ of $COMPS$ corresponds exactly one instantiation $\mathbf{COMPS}_A$ of $COMPS_A$ consistent with $COMPS$ given the definitions of elements in $COMPS_A$. We say that $\mathbf{COMPS}_A$ is the *abstraction* of $\mathbf{COMPS}$ according to AM .

3 Computing Abstractions

The hierarchical way abstract components are defined in section 2 suggests that, after an initial step of behavioral modes abstraction, the computational process

⁶Note that this guarantees that the behavioral modes of the abstract component are all distinguishable in the sense of [Torasso and Torta, 2002]

can produce new abstract components incrementally, by merging only two components at each iteration. After some finite number of iterations, arbitrarily complex abstract components can be produced.

As already mentioned in section 1, however, the admissibility of a component abstraction is not enough in order to produce useful and meaningful abstractions. We thus introduce some *cutoff* criteria on abstractions over two components (i.e. single iterations), to be enforced by the computational process.

First, we don't want to have a different behavioral mode of the abstract component for each combination of the behavioral modes of its subcomponents (limited-domain criterion); a proliferation of behavioral modes in the abstract component has negative effects on both the efficiency of diagnosis and the understandability of abstract diagnoses. We chose to impose the not too-restrictive limit $|DOM(AC)| \leq |DOM(AC')| + |DOM(AC'')|$. Second, we must be able to control the fan-in of the abstract components; indeed, if a structure-based diagnostic algorithm is used ([Darwiche, 1998]), introducing an abstract component that has a fan-in (much) larger than that of all its subcomponents leads to computational inefficiencies as pointed out in [Provan, 2001]. The limit imposed on the fan-in of the abstract component (fan-in criterion) can vary from the maximum among the fan-ins of its subcomponents to the sum of such fan-ins; the choice should be driven by the type of diagnostic algorithm to be used with the abstracted model as well as by specific characteristics of the model under consideration. In the experiments reported in section 4 we have obtained significant results by restricting the fan-in of abstract components to be at most the maximum fan-in among their subcomponents.

The following is an high-level description of the computational process:

```
Function Abstract( $SD, OBS_{AV}, \Pi$ )
   $SD := BMAbstract(SD, OBS_{AV}, \Pi)$ 
  While ( $Oracle(SD) = \langle C_i, C_j \rangle$ )
     $LocalDT = CutDT(DT, C_i, C_j)$ 
    If ( $MergeComps(LocalDT, C_i, C_j) = \langle AC, defAC \rangle$ )
       $Comps = Comps \setminus \{C_i, C_j\} \cup \{AC\}$ 
       $DT = ReviseDT(\langle AC, defAC \rangle, C_i, C_j)$ 
       $G = RecomputeGraph(DT)$ 
  Return  $SD$ 
EndFunction
```

The invocation of function $BMAbstract()$ builds simple abstract components by abstracting behavioral modes that result indistinguishable given the current observability expressed by OBS_{AV}, Π ; then a while loop is entered.

Function $Oracle()$ selects the next two candidates C_i, C_j for abstraction according to the strategies outlined in the next paragraph. Then $LocalDT$, i.e. the portion of DT relevant for computing the influence of C_i, C_j over the values of observables in OBS_{AV} , is isolated by calling $CutDT()$. The causal graph $\mathcal{G}_{loc}$ associated to $LocalDT$ essentially contains the paths from C_i, C_j to nodes in OBS_{AV} ; all the parents of the nodes on such paths are

also added. It is easy to see that, given the nodes in $\mathcal{G}_{loc}$, the remaining nodes in the original $\mathcal{G}$ do not play any role in the way C_i, C_j influence the variables in OBS_{AV} . Function *MergeComps()* (see paragraph 3.2) then tries to compute the definition of the new abstract component given *LocalDT* and in case it succeeds, *SD* is updated accordingly by function *ReviseDT()*. Intuitively, the formulas in *LocalDT* are replaced with a set of formulas whose antecedents mention the new abstract component *AC*. Given the antecedent of a formula mentioning *AC* the consequent is computed according to the definition *def_{AC}*. For example, considering again figure 1 (b), we introduce the formula $(cmd = open) \wedge AV1(abm1) \Rightarrow (in2 = in)$ since $(cmd = open) \wedge def_{abm1} \vdash (in2 = in)$ (recall that $def_{abm1} = P1(ok) \wedge V1(ok)$); similarly, we introduce formulas $(cmd = open) \wedge AV1(abm2) \Rightarrow (in2 < in)$, $(cmd = open) \wedge AV1(abm3) \Rightarrow (in2 = in)$, etc.⁷. The whole process terminates when function *Oracle()* has no new suggestions for further abstractions.

3.1 The Oracle

Function *Oracle()* must choose, at each iteration, a pair of candidate components for abstraction. Since a search over the entire space of potential abstractions would be prohibitive, the function is based on a greedy heuristic that tries to achieve a good overall abstraction without doing any backtracking. It is worth noting that the greedy approach reduces the worst-case search performed by *Oracle()* to the evaluation of all the possible pairs of components before returning the chosen pair; this can happen at most $|COMPS|$ times since every time an abstraction takes place the number of components is reduced by one and thus the search space size is clearly polynomial in $|COMPS|$.

In our experience it turned out that the heuristic H_A based on the following two principles could achieve significant results. First, H_A follows a *locality principle* for choosing C_i, C_j . Manually written structural abstractions, indeed, tend to consist in hierarchies such that at each level of the hierarchy the new abstractions involve components that are structurally close to one another. This usually has the advantage of building abstract components that have a fan-in comparable to that of their subcomponents and have a limited and meaningful set of behavioral modes. Two good examples of structural patterns that follow this principle are components connected in sequence and in parallel⁸.

Second, the heuristic prefers pairs of components that are connected to the same observables in OBS_{AV} ; this follows from the fact that if at least one of the two components is observable separately from the other, it is more unlikely to find indiscriminable instantiations of the two components.

⁷It is easy to see that, by interpreting *abm1*, *abm2*, *abm3*, *abm4* as *ok*, *leak*, *so*, *sc* respectively, *AV1* behavior is exactly that of a valve

⁸Note that the structural notions of vicinity, sequentiality and parallelism can be naturally transposed in terms of relationships in the causal graph $\mathcal{G}$

While evaluating a pair of candidates, *Oracle()* immediately enforces the fan-in criterion (this check can be easily performed at this stage). As for the limited-domain cutoff criterion and the actual feasibility of an admissible abstraction, *Oracle()* just “trusts” the heuristic H_A , and defers to *MergeComps()* the actual enforcement. There’s thus no warranty that the components selected by *Oracle()* will end up being merged but just a good chance of it to happen.

In the current implementation, *Oracle()* terminates when it can’t find any pair of components which meet the limited fan-in criterion, are connected in sequence or parallel and influence the same set of observables.

Example 3.1 Given the system in figure 1 (a) and $OBS_{AV} = \{in2, out\}$, *Oracle()* selects *P1*, *V1* as candidates according to heuristic H_A : *P1*, *P2*, e.g., are further away and there’s also an observable point between them, namely *in2*. Since the inputs to the potential abstract component *AV1* would be *in*, *cmd* and the inputs to *V1* are *out1*, *cmd* there is no increase in the fan-in of the abstract component. Thus *P1*, *V1* are returned $\square$

3.2 Abstraction of Two Components

Once two candidate components C_i, C_j have been selected, function *MergeComps()* tries to merge them into a single abstract component. The following is a sketch of the function:

```
Function MergeComps(LocalDT,  $C_i, C_j$ )
   $\mathcal{P} = \text{FindIndiscriminable}(\text{LocalDT}, C_i, C_j)$ 
  If  $(|\mathcal{P}| > |DOM(C_i)| + |DOM(C_j)|)$  Then Return NULL
  AC = NewCompName();  $def_{AC} = \emptyset$ 
  ForEach  $p \in \mathcal{P}$ 
    abm = NewBMName()
     $def_{abm} = \text{MakeABMDefinition}(p)$ 
    If  $(def_{abm} = \text{NULL})$  Then Return NULL
    Else  $def_{AC} = def_{AC} \cup \{(abm, def_{abm})\}$ 
  Return  $\langle AC, def_{AC} \rangle$ 
EndFunction
```

First, the set $DOM(C_i) \times DOM(C_j)$ is partitioned into indiscriminability classes by function *FindIndiscriminable()*. Such function considers in turn each observable M reachable from C_i, C_j and computes the set $SN(M)$ of source nodes (i.e. nodes without parents) in $\mathcal{G}_{loc}$ connected to M (excluding C_i, C_j); instantiations of the source nodes represent the contexts under which C_i, C_j influence the value of M . For each instantiation of $SN(M)$ then *FindIndiscriminable()* computes the transitive closure of each pair of behavioral modes in $DOM(C_i) \times DOM(C_j)$ and gradually refines the partition by putting into separate classes pairs that cause different values for M . It is easy to see that after *FindIndiscriminable()* has looped over each $M \in OBS_{AV}$ and each instantiation of $SN(M)$, the resulting partition $\mathcal{P}$ of $DOM(C_i) \times DOM(C_j)$ consists exactly in the indiscriminability classes of definition 2.4. Indiscriminability classes of $\mathcal{P}$ form the basis for building abstract behavioral modes definitions: associating exactly one abstract behavioral mode to each of them, guarantees that the mutual exclusion, completeness and

correctness conditions given in definition 2.7 are automatically satisfied. Since the number of classes in the partition $\mathcal{P}$ corresponds to the number of abstract behavioral modes to be eventually generated, the limited-domain cutoff criterion is applied to $|\mathcal{P}|$.

If the check is passed successfully, the generation of the definitions for the abstract behavioral modes starts, by considering an indiscriminability class p at a time and calling function *MakeABMDefinition()*. Such function tries to build an admissible behavioral mode definition by considering the admissible forms in the same order as given in definition 2.6; if it does not succeed, it returns *NULL* and the abstraction of C_i, C_j fails.

Example 3.2 Given that *Oracle()* has selected $P1, V1$ as candidates for abstraction, the admissible abstraction described in example 2.4 is computed by *MergeComps()* in the following way: function *FindIndiscriminable()* computes a partition $\mathcal{P}$ whose indiscriminability classes are the ones mentioned in example 2.2; since $|\mathcal{P}| = 4 < 6 = |DOM(P1)| + |DOM(V1)|$ the cutoff check is passed; then, function *MakeABMDefinition()* builds, for each indiscriminability class, the corresponding admissible definition shown in example 2.3 $\square$

3.3 Correctness

We now state two correctness results concerning algorithm *Abstract()*. Their validity follows intuitively by the description of the algorithm given in the previous paragraphs; because of lack of space we omit the formal proofs. The first property just states that the algorithm builds abstractions according to definitions in section 2.

Property 3.1 *Function Abstract() builds an admissible components abstraction mapping*

The second property makes explicit the correspondence between detailed and abstract diagnoses.

Property 3.2 *Let SD_A be the system description obtained from SD, OBS_{AV}, Π by applying *Abstract()* and AM the associated admissible abstraction mapping. Given a diagnostic problem $DP = \langle SD, OBS, CXT \rangle$ and the corresponding abstract diagnostic problem $DP_A = \langle SD_A, OBS_{AV}, CXT \rangle$, D is a diagnosis for DP iff its abstraction D_A according to AM is a diagnosis for DP_A*

4 Experimental Results

For testing the approach described above we have used the model of the space robotic arm SPIDER ([Portinale and Torasso, 1999]), consisting in 35 components with an average 3.43 behavioral modes each, 45 observables and 1143 formulas. All the tests have been performed using a Java implementation running on a Sun Sparc Ultra 5 equipped with SunOS 5.8.

Please note that the SPIDER system model expresses causal relationships among the variables: for example, a failure in an electrical component can influence the temperature measured at some other physically neighboring

model	testset 1	testset 2	testset 3
detailed	5.1 ± 0.3	22.0 ± 2.5	123.3 ± 23.1
abstract1	3.7 ± 0.2	11.7 ± 1.3	47.4 ± 8.4
abstract2	1.6 ± 0.1	2.1 ± 0.2	3.5 ± 0.5

Table 2: Average number of preferred diag. (conf. 95%)

components.

We have applied the *Abstract()* algorithm by considering as available observables the set of 29 sensorized observables at their maximum granularity. The resulting model, computed in about 1sec, was reduced to have 21 components with an average 2.76 behavioral modes and 548 formulas. Table 2 reports the average number of preferred (i.e. minimum fault cardinality) diagnoses produced by the diagnostic agent when using respectively the original model (*detailed*), the model obtained by performing only behavioral modes abstraction (*abstract1*) and the model created by *Abstract()* (*abstract2*). The three columns report results for three testsets of 250 cases each. The difference between the testsets lies in the number of faults (1, 2 and 3 respectively) injected in each test case.

The average times employed to solve a test case using the *detailed* model were, with a 95% confidence, 72 ± 4 msec (*testset1*), 135 ± 9 msec (*testset2*) and 333 ± 54 msec (*testset3*); by using the *abstract2* model the average times dropped to 35 ± 3 msec (*testset1*), 41 ± 3 msec (*testset2*) and 45 ± 8 msec (*testset3*).

5 Related Work and Conclusions

As noted in the introduction only a few methods have been proposed for automatic model abstraction in the context of MBD. In particular, [Sachenbacher and Struss, 2001] and [Torasso and Torta, 2002] aim at the abstraction of the values of variables in the model while the present work aims also at automatically abstracting component variables. In [Out et al., 1994] the authors introduce the notion of *ID-hierarchies* of abstract components which, as our admissible abstraction mappings, preserve a strict correspondence between an abstract diagnosis and the set of detailed diagnoses consistent with it. However, their work deals with models which represent only normal behavior of the system, which results in a significant simplification of the synthesis of behavioral modes definitions for the abstract components. In particular, the partition of the instantiations of the sub-components is always reduced to two indiscriminability classes: a singleton *OK-class* (containing the unique instance where all subcomponents are *OK*) and an *AB-class* (containing all the remaining instances, where at least one subcomponent is *AB*). Since there's no fault model, the only instance which predicts values for the observables is the one in the *OK-class*; instances in the *AB-class* do not predict anything about observable values, and thus can be safely put in the same indiscriminability class without computing any transitive closure.

Among the methods based on manual abstraction,

[Friedrich, 1993] proposes a notion of diagnosis (theory diagnoses) which exploits *ab*-clauses explicitly added to the domain theory in order to compute abstract diagnoses. Similarly, in [Console and Theseider Dupré, 1994] the model is augmented with abstraction axioms which model *is-a* relationships; in such model also observables can be expressed at different levels of abstraction. In both works the level of abstraction of diagnoses is flexible (i.e. a diagnosis can mix elements at different levels of abstraction) and is driven by the specific diagnostic case at hand (i.e. values of the observables and contexts). Our abstraction algorithm automatically synthesizes a single abstract level given the knowledge of which variables are observable at which granularity; we assume that the changes in the availability of observables are rare, so that the produced models are reused for many diagnostic cases. Moreover, the kinds of relationships that we allow among abstract component behavior and its subcomponents behavior are less restrictive than *ab*-clauses and *is-a* relationships.

The definition of admissible components abstractions given in this paper aims at building abstract models that can completely replace the detailed models without any loss of discriminability. This is different from the goal of [Mozetič, 1991] and its improvements, where the abstract model is viewed as a focusing mechanism, and is used in conjunction with the detailed model in order to improve efficiency. Our notion of indiscriminability enforces abstractions where a diagnosis D_a at the abstract level corresponds exactly to the set of detailed diagnoses whose abstraction is D_a (property 3.2). As shown by [Autio and Reiter, 1998] this is not the case for some common abstractions: in the example of the abstract *NOR* gate consisting of an *OR* and a *NOT* gates, the *ab*-clause $AB(NOR) \equiv AB(OR) \vee AB(NOT)$ does not correspond to an indiscriminability class, since the case where both *OR* and *NOT* are faulty has a different behavior than the cases where only one fault is present. Our algorithm does not synthesize such *ab*-clause, thus avoiding the consequent loss of diagnostic information.

Most of the computational effort of our approach is devoted to the computation of the indiscriminability classes; once they have been identified, the generation of behavioral modes definitions from each of them is cheap and straightforward. An alternative approach that could be worth exploring is constructive inductive learning, where techniques have been developed for the automatic synthesis of complex attributes (see e.g. [Pagallo and Haussler, 1990]).

Another interesting point to explore would be to study more sophisticated versions of the *Oracle()* function, able to recognize a wider variety of patterns in the causal graph and possibly to look ahead one or more steps in order to select the candidate pair of components; in order to keep the complexity of the search manageable we believe that such extended versions of *Oracle()* should be supported by additional domain-specific knowledge (for instance, whether certain kinds of components are likely to be connected in ways that form admissible abstract

components).

In conclusion, performing components abstraction led to dramatic reductions in the number of returned diagnoses in the experimental domain, outperforming the application of behavioral modes abstraction alone (see section 4). The reduction in the model size also resulted in significant time savings for the diagnostic process.

References

- [Autio and Reiter, 1998] K. Autio and R. Reiter. Structural abstraction in model-based diagnosis. In *Proc. ECAI98*, pages 269–273, 1998.
- [Chittaro and Ranon, 2001] L. Chittaro and R. Ranon. Hierarchical diagnosis guided by observations. In *Proc. IJCAI01*, pages 573–578, 2001.
- [Console and Theseider Dupré, 1994] L. Console and D. Theseider Dupré. Abductive reasoning with abstraction axioms. *LNCS*, 810:98–112, 1994.
- [Darwiche, 1998] A. Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.
- [Friedrich, 1993] G. Friedrich. Theory diagnoses: A concise characterization of faulty systems. In *Proc. IJCAI93*, pages 1466–1471, 1993.
- [Mozetič, 1991] I. Mozetič. Hierarchical model-based diagnosis. *Int. Journal of Man-Machine Studies*, 35(3):329–362, 1991.
- [Out et al., 1994] D.J. Out, R. van Rikxoort, and R. Bakker. On the construction of hierarchic models. *Annals of Mathematics and AI*, 11:283–296, 1994.
- [Pagallo and Haussler, 1990] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.
- [Portinale and Torasso, 1999] L. Portinale and P. Torasso. Diagnosis as a variable assignment problem: a case study in a space robot fault diagnosis. In *Proc. IJCAI99*, pages 1087–1093, 1999.
- [Provan, 2001] G. Provan. Hierarchical model-based diagnosis. In *Proc. DX01*, pages 329–362, 2001.
- [Sachenbacher and Struss, 2001] M. Sachenbacher and P. Struss. Aqua: A framework for automated qualitative abstraction. In *Proc. QR01*, 2001.
- [Struss et al., 1996] P. Struss, A. Malik, and M. Sachenbacher. Qualitative modeling is the key to automated diagnosis. In *Proc. IFAC96*, 1996.
- [Struss, 1997] P. Struss. Fundamentals of model-based diagnosis of dynamic systems. In *Proc. IJCAI97*, 1997.
- [Torasso and Torta, 2002] P. Torasso and G. Torta. Merging indiscriminable diagnoses: an approach based on automatic domains abstraction. In *Proc. DX02*, pages 43–50, 2002.

INFORMATION EXTRACTION

Information Extraction from Web Documents Based on Local Unranked Tree Automaton Inference

Raymond Kosala¹, Maurice Bruynooghe¹, Jan Van den Bussche², Hendrik Blockeel¹

¹ K.U.Leuven, Dept. of Computer Science, Celestijnenlaan 200A, B-3001 Leuven
{raymond,maurice,hendrik}@cs.kuleuven.ac.be

² University of Limburg, Dept. WNI, Universitaire Campus, B-3590 Diepenbeek
jan.vandenbussche@luc.ac.be

Abstract

Information extraction (IE) aims at extracting specific information from a collection of documents. A lot of previous work on IE from semi-structured documents (in XML or HTML) uses learning techniques based on strings. Some recent work converts the document to a ranked tree and uses tree automaton induction. This paper introduces an algorithm that uses unranked trees to induce an automaton. Experiments show that this gives the best results obtained so far for IE from semi-structured documents based on learning.

1 Introduction

Information extraction aims at extracting specific information from a collection of documents. One can distinguish between IE from unstructured and from (semi-) structured texts [Muslea, 1999]. Extracting information from web documents belongs to the latter category and gains importance [Levy *et al.*, 1998]. These documents are not written in natural language, but rather involve explicit annotations such as HTML/XML tags to convey the structure of the information, making the methods tuned towards natural language unusable.

While special query languages exist [Bry and Schaffert, 2002; XQL, 2002], their use is time consuming and requires nontrivial skill. As argued in [Muslea *et al.*, 2001; Kushmerick, 2000], there is a need for systems that learn to extract information from a few annotated examples (*wrapper induction*). Several machine learning techniques for inducing wrappers have been proposed. Examples are multi-strategy approaches [Freitag, 2000] and various grammatical inference techniques that induce a kind of delimiter-based patterns [Muslea *et al.*, 2001; Freitag and McCallum, 1999; Freitag and Kushmerick, 2000; Soderland, 1999; Freitag, 1997; Hsu and Dung, 1998; Chidlovskii *et al.*, 2000]. All these methods treat the document as a string of characters.

Structured documents such as HTML and XML documents, however, have an explicit tree structure. In [Kosala *et al.*, 2002b; 2002a], it is argued that one can better exploit this tree structure and use tree automata [Comon *et al.*, 1999]. The document tree is converted in a ranked binary tree and k -testable tree automata [Rico-Juan *et al.*, 2000] are induced and then used for the extraction task. Typically in a IE task

from structured documents, there is some structural context close to the target. After linearisation in a string, this context can be arbitrarily far away, making the learning task very difficult for string based methods. While binarisation may also increase the distance between the context and the target, they remain closer, and the learning task should be easier. This is confirmed by the experiments in [Kosala *et al.*, 2002b; 2002a]. If distance between the relevant context and the target is indeed a main factor determining the ability to learn an appropriate automaton, then an algorithm inducing a wrapper directly from the unranked tree should perform even better. This path is pursued in the current paper. As in [Kosala *et al.*, 2002b; 2002a] user intervention is limited to annotating the field to be extracted in a few representative examples. String based methods require substantially more user intervention, such as splitting the document into small fragments, and selecting some of them for use as a training example, *e.g.* [Soderland, 1999]; the manual specification of the length of a window for the prefix, suffix and target fragments [Freitag and McCallum, 1999; Freitag and Kushmerick, 2000], and of the special tokens or landmarks such as “>” or “;” [Freitag and Kushmerick, 2000; Muslea *et al.*, 2001].

The rest of the paper is organized as follows. Section 2 provides some background on unranked tree automata and their use for IE. Section 3 describes our methodology and introduces our unranked tree inference algorithm. Results are described in Section 4, related work in Section 5. Section 6 concludes.

2 Preliminaries

Grammatical inference and information extraction.

Grammatical inference (also called automata induction, grammar induction, or automatic language acquisition) refers to the process of learning rules from a set of labeled examples. The target domain is a formal language (a set of strings over some alphabet Σ) and the hypothesis space is a family of grammars. The inference process aims at finding a minimum automaton (the *canonical automaton*) that is *compatible* with the examples. There is a large body of work on grammar induction, for a survey see *e.g.* [Murphy, 1996; Parekh and Honavar, 1998].

In grammar induction, we have a finite alphabet Σ and a formal language $L \subseteq \Sigma^*$. Given a set of examples in L (S^+)

and a (possibly empty) set of examples not in $L(S^-)$, the task is to infer a deterministic finite automaton (DFA) that accepts the examples in S^+ and rejects those in S^- . In [Freitag, 1997], an IE task is mapped into a grammar induction task. A document is converted into a sequence of tokens (from Σ). Examples are transformed by replacing the token to be extracted by the special token x . Then a DFA is inferred that accepts all the transformed examples. In our case, a document is converted into a tree and the token to be extracted (at a leaf) is replaced by the special token x . Then a tree automaton is inferred that accepts all the transformed examples. When using the learned automaton, a similar transformation is done. Each token that is a candidate for extraction is in turn replaced by x . The token replaced by x is extracted iff the transformed document is accepted by the automaton.

Unranked tree automata. Some existing algorithms for string automaton induction have been upgraded to ranked tree automaton induction (e.g. [Rico-Juan *et al.*, 2000; Abe and Mamitsuka, 1997])¹. By converting documents (which are unranked) into binary trees (which are ranked), tree automaton induction can be used for IE as shown in [Kosala *et al.*, 2002b]. The present work avoids binarisation and uses unranked trees. Unranked tree automata have been studied since the late 60's, see e.g. [Brüggemann-Klein *et al.*, 2001] for a survey. To our knowledge, algorithms for inducing them do not yet exist. This paper is a first step in this direction.

An unranked label is a label with a variable rank (arity). Thus the number of children is not fixed by the label. Given a set V of labels in an unranked alphabet, we can define V^T , the set of all (unranked) trees, as follows:

- a is a tree where $a \in V$.
- $a(u_1, \dots, u_n)$ is a tree, where $a \in V$ and each u_i is a tree.

An unranked tree automaton (UTA) is a quadruple (V, Q, Δ, FS) , where V is a set of unranked labels, Q is a finite set of states, $FS \subseteq Q$ is a set of final (accepting) states, and Δ is a set of transitions where each transition is of the form $v(e) \rightarrow q$, where $v \in V$, $q \in Q$, and e is a regular expression over Q .

A bottom up UTA processes trees bottom up. When a leaf node is labeled v and there is a transition $v(e) \rightarrow q$ such that e matches the empty string, then the node is assigned state q . When an internal node is labeled v , its children have been assigned states $q_1, \dots, q_n$, and there is a transition $v(e) \rightarrow q$ such that the string $q_1, \dots, q_n$ matches the regular expression e , then the node is assigned state q . A tree is accepted if the state of its root is assigned an accepting state $q \in FS$.

3 Approach and algorithm

Preprocessing. Fig. 1 shows a representative task. For dealing with text nodes, we follow the approach described in [Kosala *et al.*, 2002b]: we replace most text nodes by CDATA², making an exception for so-called *distinguishing context*, specific text that is useful for the identification of the

field of interest. E.g., the text "Organization:" may be relevant for the extraction, hence this text should not be changed into CDATA. In each data set, at most one text field is identified as distinguishing context. It is found automatically using the approach described in [Kosala *et al.*, 2002b].

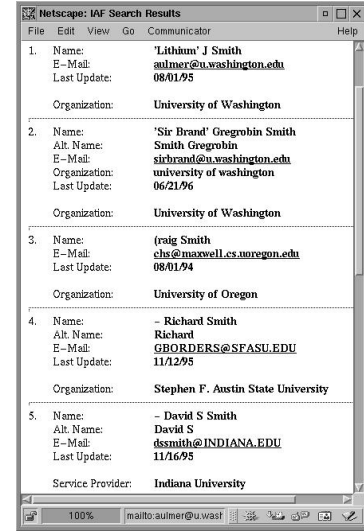


Figure 1: The fields to be extracted are the fields following the **Alt. Name** and **Organization** fields. A document consists of a variable number of records. The number of occurrences of the fields to be extracted is variable (from zero to many). Also the position is not fixed.

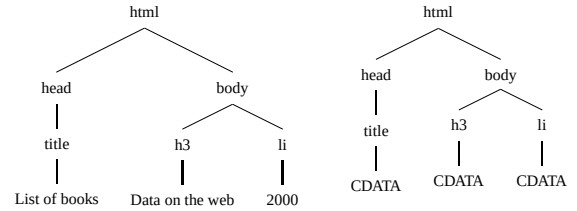


Figure 2: The left figure is an HTML tree. The right one is the same tree after abstracting the text nodes

Approach. Our approach for information extraction has the following characteristics:

- Strings stored at the text nodes are treated as single labels. If extracted, the whole string is returned.
- One automaton is learned for one type of field to be extracted, e.g., the field following "Organization".
- In examples used during learning, one target field (a text node) is replaced by x . A document gives rise to several examples when several targets occur.

The learning phase proceeds as follows:

- Replace in the examples the target by " x ", the distinguishing context(s) (if present) by " ctx " and all other text fields by CDATA.
- Map examples to trees and learn a tree automaton.

The extraction phase repeats for all candidate targets:

- Map the document to a tree and replace the candidate target by " x ", the distinguishing context(s) (if present)

¹Ranked: the label determines the number of children.

²See Fig. 2 for an example.

by “*ctx*” and all other text fields by CDATA.

- Run the tree automaton; if the tree is accepted, then output the original text of the node labeled with x .

3.1 Local unranked tree automaton inference algorithm

Preliminaries. A partition of a set S is a set of disjoint nonempty subsets of S such that the union of the subsets is S . The height of a tree is defined as: $height(v) = 1$ and $height(v(t_1, \dots, t_n)) = 1 + \max(height(t_1), \dots, height(t_n))$. If $t = l(t_1, \dots, t_n)$ then the t_i are called subtrees of t , and subtrees of t_i are also subtrees of t . When a tree t is “cut off” at level k , this means all subtrees at level k become leaves. Thus the height of the cut-off tree can be at most k . Given a tree t , the set of roots $r_k(t)$ is the singleton containing t cut off at k ; the set of forks $f_k(t)$ contains all subtrees of t of height at least k , cut off at k ; and the set of subtrees $s_k(t)$ contains all subtrees of t of height at most k . Roughly, these sets respectively collect all subtrees of height k at the top, in the middle, and at the bottom of t . See [Rico-Juan et al., 2000] for a formal definition.

Example 1 Let $t = a(b(a(b, x)), c)$ then $r_1(t) = \{a\}$, $f_2(t) = \{a(b, c), b(a), a(b, x)\}$, and $s_1(t) = \{b, x, c\}$.

Algorithm 1 Local unranked tree inference algorithm

Input: Positive examples (set T) and an integer $k_c > 1$.

Output: An unranked tree automaton (V, Q, Δ, FS) .

```

1:  $FS = \mathcal{F} = \mathcal{S} = \emptyset$ ;
2: for each  $t \in T$  do
3:    $t' = convert\_labels(t)$ 
4:    $\mathcal{F} = \mathcal{F} \cup f_2(t')$  % 2-forks
5:    $\mathcal{S} = \mathcal{S} \cup s_1(t')$  % 1-subtrees
6:    $FS = FS \cup r_1(t')$  % 1-root
7: end for
8:  $Q = FS \cup \{r_1(f) | f \in \mathcal{F}\} \cup \mathcal{S}$ 
9:  $\Delta = \{v(\epsilon) \rightarrow v | v \in \mathcal{S}\}$ 
10:  $P = partition(\mathcal{F})$ 
11: for each  $(v, Str) \in P$  do
12:   if  $v = v_x$  and the length of the shortest string in  $Str \geq k_c$  then
13:      $e = k\_contextual(k_c, Str)$  % a DFA
14:   else if  $v = v_x$  then
15:      $e = k\_contextual(2, Str)$  % a DFA
16:   else
17:      $e = *$  % accepts any sequence
18:   end if
19:    $\Delta = \Delta \cup \{v(e) \rightarrow v\}$ 
20: end for
```

The algorithm. We have designed a procedure, which is shown in Algorithm 1, to learn an unranked tree automaton from a set T of positive examples. The inferred automaton is “local” in the sense that it identifies a tree with its 2-forks as defined above. Note that DTDs for XML do the same [Murata et al., 2001]. In addition to the input T , it takes as additional parameter a positive integer $k_c > 1$, which is the parameter for the $k_contextual$ subroutine that it calls.

In a first for-loop, our algorithm collects all 2-forks, 1-subtrees and 1-roots. The latter become final states. However,

before these steps, the node labels of each input tree t are rewritten using the function $convert_labels(t)$. This function (Algorithm 2) rewrites node labels. The label v of the root of a subtree is changed into v_x if that subtree contains a “ x ” and into v_{ctx} if that subtree contains a “*ctx*”. The effect is that the special labels are remembered up to the root³.

Algorithm 2 convert_labels(t)

Input: A tree $t = v(t_1, \dots, t_n)$.

Output: A tree t'

```

1:  $t'_1 = convert\_labels(t_1), \dots, t'_n = convert\_labels(t_n)$ 
2: if  $\exists i : t'_i$  has label  $l_x$  or  $x$  then
3:    $t' = v_x(t'_1, \dots, t'_n)$ 
4: else if  $\exists i : t'_i$  has label  $l_{ctx}$  or ctx then
5:    $t' = v_{ctx}(t'_1, \dots, t'_n)$ 
6: else
7:    $t' = t$ 
8: end if
```

Algorithm 3 $k_contextual(k, Str)$

Input: An integer $k > 1$ and set of strings Str over Σ

Output: A k -contextual DFA (Σ, Q, Δ, FS) .

```

1:  $Q = FS = \Delta = \emptyset$ ;
2:  $U = \{\#^{k-1}s\# | s \in Str, \# \notin \Sigma\}$ 
3:  $Grams = \{s | s \text{ is a substring of } u, u \in U, |s| = k\}$ 
4: for each  $g \in Grams$  do
5:   Let  $g = uv$  with  $|u| = 1$  and  $|v| = k - 1$ 
6:    $Q = Q \cup \{v\}$ 
7:   if  $v = v'\#$  then
8:      $FS = FS \cup \{v\}$ 
9:   end if
10:   $\Delta = \Delta \cup \{uv \rightarrow v\}$ 
11: end for
```

Next, the states are collected (the 1-root, the 1-subtrees, and the 1-roots of the forks) in Q , the transitions are initialized with one transition for each 1-subtree and the 2-forks are partitioned according to the label of the forks’ root. The latter results in a set of pairs (v, Str) with Str a set of sequences, each sequence being the children of a fork. E.g., $\{a, \{(b, c), (b, c, c)\}\}$ represents two forks with root label a .

In the second for-loop, the $k_contextual$ algorithm (Algorithm 3) [Muggleton, 1990; Ahonen, 1996] is used to learn a deterministic finite automaton (DFA) that can be used as the representation of the regular expressions e to be used in the transitions $v(e) \rightarrow q$ of the UTA. As an illustration, consider an input string ab for $k = 3$. The value of U is $\#\#ab\#$ and one obtains $FS = \{b\#$ and $\Delta = \{\#\#a \rightarrow \#a, \#ab \rightarrow ab, ab\# \rightarrow b\#$, where $\#$ is a distinguished label that is not in Σ . It is capable of identifying in the limit any k -contextual string automaton, a subset of the finite automata, from positive examples only. In principle, we could use any string automaton inference algorithm for this purpose (see e.g. [Murphy, 1996; Parekh and Honavar, 1998] for others). We choose the $k_contextual$ algorithm because it is efficient, simple, and works well in practice.

³As a result, the tree automaton is not purely local.

An element (v, Str) of the partition gives all positive examples for a particular label v . The regular expression e captures the regularity in these examples (the document content model in XML terminology). To obtain sufficient generalization, we decided, after some experimentation, to distinguish three cases. If all children of a v_x node are long enough, we construct a DFA using the k -contextual algorithm with k -value k_c , otherwise with value 2. For other labels (either a v_{ctx} label or an original label), we ignore the content of the children and accept any sequence (the regular expression “*”).

Using a result of [Muggleton, 1990], it is quite straightforward to make an incremental version of Algorithm 1. We omit it here due to lack of space. Using a basic result of [Angluin, 1980], we can prove:

Theorem 1 *Every unranked tree language that is definable by a local unranked tree automaton with k -contextual regular expressions is identifiable in the limit, from positive examples only, by our algorithm.*

Example 2 *Applying the Algorithm 1 on the tree of Example 1 for $k_c = 2$, we obtain:*

- $t' = a_x(b_x(a_x(b, x)), c)$
- $r_1(t') = \{a_x\}$
- $\mathcal{F} = f_2(t') = \{a_x(b_x, c), b_x(a_x), a_x(b, x)\}$
- $\mathcal{S} = s_1(t') = \{b, x, c\}$.
- $FS = \{a_x\}$
- $Q = \{a_x, b_x, b, x, c\}$
- $P1 = \{a_x, \{(b_x, c), (b, x)\}\}$
- $P2 = \{b_x, \{(a_x)\}\}$
- $DFA1 : \Delta = \{\#, b_x \rightarrow b_x; b_x, c \rightarrow c; c, \# \rightarrow \#; \#, b \rightarrow b; b, x \rightarrow x; x, \# \rightarrow \#\}, Q = \{b_x, c, \#, b, x\}, FS = \{\#\}$.
- $DFA2 : \Delta = \{\#, a_x \rightarrow a_x; a_x, \# \rightarrow \#\}, Q = \{a_x, \#\}, FS = \{\#\}$.
- *transitions:*
 - $b \in \mathcal{S} : b(\epsilon) \rightarrow b$
 - $x \in \mathcal{S} : x(\epsilon) \rightarrow x$
 - $c \in \mathcal{S} : c(\epsilon) \rightarrow c$
 - $P1 : a_x(DFA1) \rightarrow a_x$
 - $P2 : b_x(DFA2) \rightarrow b_x$

4 Experimental results

We evaluated our method on two semi-structured data sets commonly used in IE research (available online from <http://www.isi.edu/~muslea/RISE/>): a collection of web pages containing people’s contact addresses (the Internet Address Finder (IAF) database) and a collection of web pages about stock quotes (the Quote Server (QS) database). For each dataset, there are two tasks; they are the extraction of alternative and organization fields in the IAF dataset and of the date and volume fields in the QS dataset. Each dataset consists of 10 documents. The number of fields to be extracted is respectively 94 (IAF-organization), 12 (IAF-alt.name), 24 (QS-date), and 25 (QS-vol). We choose these datasets because they are benchmark datasets that are commonly used for research in IE; hence they allow us to compare results. In order to provide a close comparison, we use the same train and test splits as in [Freitag and Kushmerick, 2000]. In addition, they require the extraction of a whole leaf node (our algorithms are designed for that task).

Moreover, the results obtained so far [Muslea *et al.*, 1999; Hsu and Chang, 1999] indicate that they are difficult tasks. In fact one of the authors in [Muslea *et al.*, 1999] has tried to build a hand-crafted extractor given all available documents from the QS dataset and achieved only 88% accuracy (or recall in our criteria below). We also test our method on a significantly reduced Shakespeare XML dataset (available online from <http://www.ibiblio.org/bosak/>). We use the same training and test set as in [Kosala *et al.*, 2002b]. The task on this dataset is to extract the second scene from one act in a particular play.

We apply the commonly used criteria of IE research for evaluating our method. Precision P is the number of correctly extracted objects divided by the total number of extractions, while recall R is the number of correct extractions divided by the total number of objects present in the answer template. The F1 score is defined as $2PR/(P + R)$, the harmonic mean of P and R . Table 1 shows the results we obtained as well as those obtained by some current state-of-the-art string-based methods: an algorithm based on Hidden Markov Models (HMMs) [Freitag and McCallum, 1999], the Stalker wrapper induction algorithm [Muslea *et al.*, 2001] and BWI [Freitag and Kushmerick, 2000]. We also include the results of the k -testable algorithm in [Kosala *et al.*, 2002b] which works on ranked trees. The results of HMM, Stalker and BWI are taken from [Freitag and Kushmerick, 2000]. All tests are performed with 10-fold cross validation following the splits used in [Freitag and Kushmerick, 2000], except in the small Shakespeare dataset which uses 2-fold cross validation. Each split has 5 documents for training and 5 for testing. We refer to Section 5 for a description of these methods.

Table 1 shows the best results of the unranked method with a certain k_c (cross-validation on one fold of 50% random training and test examples.) As can be seen, our method is the only one giving optimal results.

Table 2 shows the value of k and k_c used respectively by the k -testable and the unranked algorithm. It is well-known that when learning from positive examples only, there is a problem of over-generalization. Our algorithm requires a cross-validation on the value of k_c to avoid over-generalization.

Algorithm 1 runs in time $O(cn)$, where n is the total number of nodes in the training examples and c is a constant. It takes an average time between 19 and 26 ms in a 1.7 Ghz Pentium 4 PC for Algorithm 1 to learn an example in the IAF and QS datasets.

5 Related work

The IE work for (semi-) structured texts can be divided into systems built manually using a knowledge engineering approach, *e.g.* [Hammer *et al.*, 1997] and systems built (semi-) automatically using machine learning techniques or other algorithms. The latter are called wrapper induction methods. We briefly survey them.

The three systems referred to in Table 1 learn wrappers based on regular expressions. BWI [Freitag and Kushmerick, 2000] uses a boosting approach in which the weak learner learns a simple regular expression with high precision but low

Table 1: Comparison of the results

	IAF - alt.name			IAF - organization			QS - date			QS - volume			Small Shakespeare		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
HMM	1.7	90	3.4	16.8	89.7	28.4	36.3	100	53.3	18.4	96.2	30.9	-	-	-
Stalker	100	-	-	48.0	-	-	0	-	-	0	-	-	-	-	-
BWI	90.9	43.5	58.8	77.5	45.9	57.7	100	100	100	100	61.9	76.5	-	-	-
k -testable	100	73.9	85	100	57.9	73.3	100	60.5	75.4	100	73.6	84.8	56.2	90	69.2
unranked	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Table 2: Parameters used for the experiments

	IAF - alt.name	IAF - org.	QS - date	QS - volume	Small Shakespeare
k -testable (k)	4	4	2	5	3
unranked (k_c)	2	2	2	7	3

recall. The HMM approach [Freitag and McCallum, 1999] learns a hidden Markov model; it solves the problem of estimating probabilities from sparse data by using a statistical technique called shrinkage. This model has been shown to achieve state-of-the-art performance on a range of IE tasks. Stalker [Muslea *et al.*, 2001] induces extraction rules that are expressed as simple landmark grammars, a class of finite automata; it performs hierarchical extraction guided by a manually built *embedded catalog* tree that describes the structure of the fields to be extracted.

Several techniques based on naive-Bayes, two regular language inference algorithms, and their combinations for IE from unstructured texts are described in [Freitag, 1997]. WHISK [Soderland, 1999] learns extraction rules based on a form of regular expression patterns with a top-down rule induction technique. [Chidlovskii *et al.*, 2000] describe an incremental grammar induction approach; they use a subclass of deterministic finite automata that do not contain cyclic patterns. The SoftMealy system [Hsu and Dung, 1998] learns separators that identify the boundaries of the fields of interest. [Hsu and Chang, 1999] propose two classes of SoftMealy extractors: single pass, which is biased for tabular documents such as QS data (they reach up to 97% recall), and multi pass, which is biased for tagged-list document such as IAF data (they reach up to 57% recall). We cannot really compare results because the experimental setting is different.

All above methods use algorithms for learning string languages and require some manual intervention. HMMs and BWI require to specify a window length for the prefix, suffix and the target fragments. Stalker and BWI require to specify special tokens or landmarks such as “>” or “;”. SoftMealy extractors in [Hsu and Chang, 1999] requires to choose between single and multi pass bias.

In [Kosala *et al.*, 2002b], the document is converted in a ranked (binary) tree and an algorithm is used that induces a k -testable tree automaton. However, as binarisation increases the distance between target and distinguishing context, large k are needed and the resulting automaton is precise but does not generalize enough (Table 1). In [Kosala *et al.*, 2002a], the same authors generalize the obtained automaton by selectively introducing wild-card labels. This gives some modest improvement in recall but does not solve the problem. Our unranked tree automaton induction algorithm does.

The most apparent limitation of our method is that it can only output a whole text node. To overcome this, it could be extended with a second step where string based methods are used to extract part of the text node. For example, to extract the substring “the web” from the whole string “Data on the web” (Fig. 2). Another limitation is that our method only output a single field (slot) in one run.

Finally, a disadvantage that is not apparent from the results reported above, is that when the identification of target fields does not require dependencies between nodes in the tree but can rely on a local pattern (e.g., the field to be extracted is always surrounded by specific delimiters), our tree based method needs more examples to learn the same extraction rule as methods that automatically focus on local patterns. Intuitively, more variations in further-away nodes need to be observed before these variations are considered irrelevant. This is simply an instance of the well-known trade-off between the generality of a hypothesis space and the efficiency with which the correct hypothesis can be extracted from it.

Some other approaches that exploit the structure of the documents are: WL² [Cohen *et al.*, 2002], a logic-based wrapper learner that uses multiple (string, tree, visual, and geometric) representations of the HTML documents. In fact, WL² is able to extract all four tasks in the IAF and QS datasets with 100% recall; and wrappers [Sakamoto *et al.*, 2002] that identify a field with a path from root to leaf, imposing conditions on each node in the path.

6 Conclusion

We have presented an algorithm for the inference of a local unranked tree automaton with k -contextual regular expressions and have shown that it can be used for IE from structured documents. Our results confirm the claim of [Kosala *et al.*, 2002b] that utilizing the tree structure of the documents is worthwhile for structured IE tasks. Whereas the latter work transforms the positive examples into binary ranked trees, we use them directly as unranked trees. Our results are optimal for the previously considered benchmarks, substantially improving upon the published results of other string and tree based methods and are a strong indication that unranked tree automata are much better suited than ranked ones for structured IE tasks.

Possible future work includes experiments with larger and more difficult datasets, adapting tree automata for multi slot extraction in one run, and a more formal analysis of the algorithm.

Acknowledgements

We thank Nicholas Kushmerick for providing us with the datasets used for BWI experiments and anonymous reviewers for their very valuable comments. This work is supported by the FWO project query languages for data mining. Hendrik Blockeel is a post-doctoral fellow of the Fund for Scientific Research of Flanders.

References

- [Abe and Mamitsuka, 1997] N. Abe and H. Mamitsuka. Predicting protein secondary structure using stochastic tree grammars. *Machine Learning*, 29:275–301, 1997.
- [Ahonen, 1996] H. Ahonen. *Generating grammars for structured documents using grammatical inference methods*. PhD thesis, University of Helsinki, Department of Computer Science, 1996.
- [Angluin, 1980] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.
- [Brüggemann-Klein et al., 2001] A. Brüggemann-Klein, M. Murata, and D. Wood. newblock Regular tree and regular hedge languages over unranked alphabets, TR 2001-05, HKUST-TCSC, 2001.
- [Bry and Schaffert, 2002] F. Bry and S. Schaffert. Towards a declarative query and transformation language for XML and semistructured data: Simulation unification. In *ICLP 2002*, pages 255–270, 2002.
- [Chidlovskii et al., 2000] B. Chidlovskii, J. Ragetli, and M. de Rijke. Wrapper generation via grammar induction. In *ECML 2000*, pages 96–108, 2000.
- [Cohen et al., 2002] W. Cohen, M. Hurst, and L. S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *The Eleventh International World Wide Web Conference (WWW2002)*, 2002.
- [Comon et al., 1999] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Available on: <http://www.grappa.univ-lille3.fr/tata>, 1999.
- [Freitag and Kushmerick, 2000] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *AAAI/IAAI 2000*, pages 577–583. AAAI Press, 2000.
- [Freitag and McCallum, 1999] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
- [Freitag, 1997] D. Freitag. Using grammatical inference to improve precision in information extraction. In *ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, 1997.
- [Freitag, 2000] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000.
- [Hammer et al., 1997] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha. Extracting semistructured information from the Web. In *Workshop on Management of Semistructured Data*, pages 18–25, 1997.
- [Hsu and Chang, 1999] C-N. Hsu and C-C. Chang. Finite-state transducers for semi-structured text mining. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, 1999.
- [Hsu and Dung, 1998] C-N. Hsu and M-T. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Information Systems*, 23(8):521–538, 1998.
- [Kosala et al., 2002a] R. Kosala, M. Bruynooghe, H. Blockeel, and J. Van den Bussche. Information extraction by means of a generalized k -testable tree automata inference algorithm. In *Information Integration and Web-based Applications & Services, IIWAS, Proc.*, pages 105–109, 2002.
- [Kosala et al., 2002b] R. Kosala, J. Van den Bussche, M. Bruynooghe, and H. Blockeel. Information extraction in structured documents using tree automata induction. In *PKDD 2002*, pages 299–310, 2002.
- [Kushmerick, 2000] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.
- [Levy et al., 1998] A. Levy, C. Knoblock, S. Minton, and W. Cohen. Trends and controversies: Information integration. *IEEE Intelligent Systems*, 13(5), 1998.
- [Muggleton, 1990] S. Muggleton. *Inductive Acquisition of Expert Knowledge*. Addison-Wesley, 1990.
- [Murata et al., 2001] M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema languages using formal language theory. In *Extreme Markup Languages*, 2001.
- [Murphy, 1996] K. Murphy. Learning finite automata. TR 96-04-017, Santa Fe Institute, 1996.
- [Muslea et al., 1999] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Int. Conf. on Autonomous Agents*, 1999.
- [Muslea et al., 2001] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.
- [Muslea, 1999] I. Muslea. Extraction patterns for information extraction tasks: A survey. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
- [Parekh and Honavar, 1998] R. Parekh and V. Honavar. *Automata Induction, Grammar Inference, and Language Acquisition*. Handbook of Natural Language Processing. New York: Marcel Dekker, 1998.
- [Rico-Juan et al., 2000] J.R. Rico-Juan, J. Calera-Rubio, and R.C. Carrasco. Probabilistic k -testable tree-languages. In *Grammatical Inference: Algorithms and Applications ICGI 2000*, pages 221–228, 2000.
- [Sakamoto et al., 2002] H. Sakamoto, H. Arimura, and S. Arikawa. Knowledge discovery from semistructured texts. In *Progress in Discovery Science - Final Report of the Japanese Discovery Science Project*, pages 586–599. Springer, 2002.
- [Soderland, 1999] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.
- [XQL, 2002] Xquery 1.0: An xml query language. W3C Working Draft 16 August 2002. www.w3.org/TR/xquery/.

Intelligent Multimedia Indexing and Retrieval through Multi-source Information Extraction and Merging*

Jan Kuper[§], Horacio Saggion[†], Hamish Cunningham[†], Thierry Declerck[‡],
Franciska de Jong[§], Dennis Reidsma[§], Yorick Wilks[†], Peter Wittenburg[#]

[§]Department of Computer Science, University of Twente, The Netherlands; [†]Department of Computer Science, University of Sheffield, Sheffield, UK; [‡]DFKI GmbH, Saarbruecken, Germany; [#]Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands email: jankuper@cs.utwente.nl, saggion@dcs.shef.ac.uk

Abstract

This paper reports work on automated meta-data creation for multimedia content. The approach results in the generation of a conceptual index of the content which may then be searched via semantic categories instead of keywords. The novelty of the work is to exploit multiple sources of information relating to video content (in this case the rich range of sources covering important sports events). News, commentaries and web reports covering international football games in multiple languages and multiple modalities is analysed and the resultant data merged. This merging process leads to increased accuracy relative to individual sources.

1 Introduction

Multimedia repositories of moving images, texts, and speech are becoming increasingly available. This together with the needs for ‘video-on-demand’ systems requires fine-grain indexing and retrieval mechanisms allowing users access to specific segments of the repositories containing specific types of information. Annotation of video is usually carried out by humans following strict guidelines. Video material is usually annotated with ‘meta-data’ such as names of the people involved in the production of the visual record, places, dates, and keywords that capture the essential content of what is depicted. Still, there are a few problems with human annotation. Firstly, the cost and time involved in the production of fine-grained semantic “surrogates” of the programme is extremely high; secondly, humans are rather subjective when assigning descriptions to visual records; and thirdly, the level of annotation required to satisfy a user’s needs can hardly be achieved with the use of mere keywords. In order to tackle these problems, indexing methods based on image processing have been developed [Chang *et al.*, 1998]. Content-based indexing and retrieval of visual records is based on features such as colour, texture, and shape. Yet visual understanding is not well advanced and is very difficult even in closed domains. As a consequence, various ways to explore the use of collateral linguistic material have been studied for tasks

such as automatic indexing [De Jong *et al.*, 2000], classification [Sable and Hatzivassiloglou, 1999], or understanding [Srihari, 1995] of visual records.

In this paper, we present an integrated solution to the problem of multimedia indexing and search: the MUMIS¹ concept. Our solution consists of using information extracted from different sources (structured, semi-structured, free, etc.), modalities (text, speech), and languages (English, German, Dutch) all describing the same event to carry out data-base population, indexing, and search. The novelty of our approach is not only the use of these ‘heterogeneous’ sources of information but also the combination or cross-source fusion of the information obtained from the separate sources. Single-document, single-language information extraction is carried out by independent systems that share a semantic model and multi-lingual lexicon of the domain. The results of all information extraction systems are merged by a process of alignment and rule-based reasoning that also uses the semantic model. In the rest of this paper we describe in detail the context of the project, and each of the modules involved in the automatic derivation of annotations. However, the emphasis is on the merging component.

2 The MUMIS Project

In MUMIS various software components operate off-line to generate formal annotations from multi-source linguistic data in Dutch, English, and German to produce a composite time-coded index of the events on the multimedia programme. The domain chosen for tuning the software components and for testing is football.

A corpus of collected textual data in the three languages was used to build a multi-lingual lexicon and shared ontology of the football domain. Based on this shared model, three different off-line Information Extraction components, one per language, were developed (see section 3). They are used to extract the key events and actors from football reports and to produce XML output. A merging component or cross-document co-reference mechanism has been developed to merge the information produced by the three IE systems (see section 4). Audio material is being analysed by Phicos [Steinbiss *et al.*, 1993], an HMM-based recognition system,

*Funded by EC’s 5th Framework HLT Programme under grant number IST-1999-10651

¹Multimedia Indexing and Searching Environment, see <http://parlevink.cs.utwente.nl/projects/mumis/>

Formal text
England 1 - 0 Germany Shearer (52) Bookings Beckham (42)...
Ticker
41 mins: Beckham is shown a yellow card for retaliating on Ulf Kirsten seconds after he is denied a free-kick. 40' Hoekschoop Engeland met David Beckham. Slecht getrap. Meteen maakt Beckham daarna een fout en krijgt een gele kaart.
Match
David Beckham - a muted force in attack - was shown a yellow card for a late challenge on Kirsten...
Transcription
...it's gonna be a card here for David Beckham it is yellow mmm well again his was the name in the post match headlines... David Beckham hielt die Sohle noch drüber schauen Sie mit dem Hinterteil auch harter Einsatz gegen Kirsten und Collina zeigt ihm Gelb eine der Unarten leider von David Beckham Beckham met*x Kirsten dat is nou weer dom wat die Beckham doet ja zal ie dat dan nooit leren Kirsten overdrijft nu hoor maar Kirsten gaat 't duel in geeft een zet en dan reageert Beckham op deze manier in ieder geval krijgt ie dan weer geel

Table 1: Different accounts of the same event in different languages

in order to obtain transcriptions of the football commentaries (spontaneous speech). It uses acoustic models, word-based language models (unigram and bigram) and a lexicon. For Dutch, English, and German different recognition systems have been developed (i.e., different phone sets, lexicons, and language models are used).

JPEG keyframe extraction from MPEG movies around a set of pre-defined time marks - result of the information extraction component - is being carried out to populate the database. The on-line part of MUMIS consists of a state of the art user interface allowing the user to query the multimedia database. The interface makes use of the lexica in the three target languages and domain ontology to assist the user while entering his/her query. The hits of the query are indicated to the user as thumbnails in the storyboard together with extra information about each of the retrieved events. The user can select a particular fragment and play it.

2.1 Domain and Ontology

An analysis of the domain and a user study led us to propose 31 types of event for a football match (*kick-off*, *substitution*, *goal*, *foul*, *red card*, *yellow card*, etc.) that need to be identified in the sources in order to produce a semantic index. The following elements associated with these events are extracted: players, teams, times, scores, and locations on the pitch.

An ontology has been developed for these events and their actors, it contains some 300 concept nodes related as an 'is-a' hierarchy. The link between the concepts and the three languages consists of a flexible XML format which maps concepts into lexical entries.

Sources used for information extraction are: formal texts, tickers, commentaries, and audio transcriptions (see Table 1). Ticker reports are particularly important in the generation of formal annotations. These texts are a verbal account of events over time stamps. They also follow a specific text structure consisting of a 'ticker header', in which information about lists of players and the result of the game is usually stated, and 'ticker sections' grouping together sentences describ-

ing events under single time stamps. Another very valuable source for the generation of the annotations are the spoken transcriptions that, even with the many errors they contain, still provide exact temporal information.

3 Extracting Information from Heterogeneous Sources

Information extraction is the process of mapping natural language into template-like structures representing the key (semantic) information from the text. These structures can be used to populate a database, used for summarization purposes, or as a semantic index as in our approach. Key to the information extraction process is the notion of "domain", scenario or template that represents what information should be extracted. IE has received a lot of attention in the last decade, fuelled by the availability of on-line texts and linguistic resources and the development of the Message Understanding Conferences [Grishman and Sundheim, 1996]. Traditionally, IE applications have tended to concentrate on a small number of events (typically one), MUMIS addresses the challenge of multi-event extraction.

Multi-lingual IE has been tried in the M-LaSIE system [Gaizauskas *et al.*, 1997], where the same underlying components and a bi-lingual dictionary are used for two different languages (English and French). MUMIS differs from that system in that it operates with three different off-line information extraction components, one per language, that produce the same "language-free" representation. In this paper we give only a brief description of the English and German IE systems.

3.1 Extraction from English Sources

IE from English sources is based on the combination of GATE² components for finite state transduction [Cunningham *et al.*, 2002] and Prolog components for parsing and discourse interpretation. The components of the system are: tokeniser, segmenter, gazetteer lookup (based on lists of entities of the domain), semantic tagger, shallow pronominal co-referencer, part-of-speech tagger, lemmatiser, chart parser, discourse interpreter (ontology-based co-referencer), and template extractor. These components are adapted and combined to produce four different system configurations for processing different text-types and modalities (transcriptions, formal texts, semi-formal texts, and free texts). The analysis of formal texts and transcriptions is being done with finite state components because the very nature of these linguistic descriptions make appropriate the use of shallow natural language processing techniques. For example, in order to recognise a *substitution* in a formal text it is enough to identify players and their affiliations, time stamps, perform shallow co-reference and identification of a number of regular expressions to extract the relevant information. In our system, regular expressions operate on annotations (not on strings) and produce semantic information. We make use of the Java Annotation Pattern Engine (JAPE) formalism [Cunningham *et al.*, 2002] to code our regular grammar. Below, we present

²GATE is a free architecture for natural language engineering.

one example of the use of JAPE that accurately identify *substitutions* in speech transcriptions:

```
Rule: Subs1
(({Player})) ({Token.string == 'is'}) ({Replace})
(({Player})):annotate --> :annotate.SubsEvent =
{rule = "Subs1"}
```

Complex linguistic descriptions are fully analysed because of the need to identify logical subjects and objects as well as to solve pronouns and definite expressions (e.g., “the Barcelona striker”) relying on domain knowledge encoded in the ontology of the domain. Domain knowledge establishes, for example, that the two players involved in a *substitution* belong to the same team. This semantic constraint is used in cases such as “he is replaced by Ince” to infer that the antecedent of the pronoun “he” belongs to the “English” team (because “Ince” does).

Logic-based information extraction rules operate on logical forms produced by the parser and enriched during discourse interpretation. They rely on the ontology to check constraints (e.g., type checking, ontological distance, etc.). The following logic-based rule is used to extract the participants of a substitution when syntactic and semantic information is available:

```
replace_event(E) & lsubj(E,P1) & lobj(E,P2)
& player(P1) & player(P2) => substitution_event(E) &
player_1(E,P1) & player_2(E,P2)
```

‘lsubj’ and ‘lobj’ represent the logical subject and object of an event. Note that, contrary to the regular case, the ‘lsubj’ and ‘lobj’ relations, being semantic in nature, are long distance relations.

3.2 Extraction from German Sources

The German IE system is based on an integrated set of linguistic tools called SCHUG: Shallow and Chunk based Unification Grammar [Declerck, 2002]. The chunking procedure of SCHUG consists of a rule-based sequence of cascades (based on the work by [Abney, 1996]), which produces a rich linguistic representation, including grammatical functions and resolution of co-reference and ellipsis. In order to detect these accurately, an analysis of the clauses of a sentence is required. Clauses are subparts of a sentence that correspond to a (possibly complex) semantic unit. Each clause contains a main verb with its complements (grammatical functions) and possibly other chunks (modifiers).

Applied to the football domain, SCHUG inspects the common MUMIS ontology and enriches the linguistic annotation produced with domain-specific information encoded in the ontology. Below (see Table 2) we show one example of the semantic annotation generated by SCHUG when applied to an on-line ticker text (game England-Germany). Here, various relations (player, location, etc.) and events (*free-kick*, *fail to score a goal*) that are relevant to the football domain are recognised.

Some relations are not explicitly mentioned, but can still be inferred by the MUMIS system. For example, the team for which “Ziege” is playing can be inferred from the ontological information that a player is part of a team and the instance of

```
7. Ein Freistoss von Christian
Ziege aus 25 Metern geht ueber das Tor.
''A 25-meter free-kick by Christian
Ziege goes over the goal.''

<events>
<event id="e1" clause="cls1" event-name=
"free-kick">
<relations>
<relation id="r1" reltype="player"
player-name="Christian Ziege" in-team="Germany"/>
...
</relations>
</event>
<event id="e2" clause="cls1" event-name=
"goal-scene-fail">
...
</event>
</events>
```

Table 2: Semantic annotation in SCHUG

this particular team can be extracted from additional texts or meta-data. In this way, information not present in the text directly can be added by additional information extraction and reasoning.

Since formal texts require only little linguistic analysis, but rather an accurate domain-specific interpretation of the jargon used, a module has been defined within SCHUG, which in a first step maps the formal texts onto an XML annotation, giving the domain semantic of the expressions in the text (the approach taken for formal texts is similar to the one followed by the English IE system). In a second step SCHUG *merges* all the XML annotated formal texts about one game. Those merged annotations are generated at a level that requires only little linguistic analysis, and basically reflect domain specific information about actors and events involved in the text. The SCHUG module applied at this level also extracts meta-data information: name of the game, date and time of the game, intermediate and final scores etc. This is quite important, since the meta-data can guide the use of the annotations produced so far for supporting linguistic analysis and information extraction applied to more complex documents.

4 Merging

Merging, also known as cross-document co-reference [Bagga and Baldwin, 1999], is the process of deciding whether two linguistic descriptions from different sources refer to the same entity or event. The merging component in MUMIS combines the partial information as extracted from various sources, such that more complete annotations can be obtained. Radev and McKeown [1998] developed a knowledge-based multi-document summarization system based on information extraction and merging of single ‘terrorist’ events. The novelty of our approach consists in applying merging to multiple events extracted from multiple sources.

As is to be expected, complete recognition of events in natural language sentences is extremely difficult. Often, events will be only partially recognised. The merging component of the MUMIS project aims to fill in missing aspects of events with information gathered from other documents. For example, the Dutch information extraction system recognised in

document *A* on the match Netherlands-Yugoslavia from the European championships 2000 that a save was performed in the 31st minute. In addition, it recognised the names of two players: Van der Sar (the Dutch goalkeeper) and Mihajlovic (a Yugoslavian player), but it could not figure out which of these two players performed the save. In document *B* it recognised a *free-kick* in the 30th minute, and the names of the same two players. Again, it did not succeed in finding out which player took the *free-kick*.

The fact that the same two players are involved, plus the small difference between the time-stamps, strongly suggests that both descriptions are about the same event in reality. The merging part of the MUMIS project matches these partial data together, and concludes that it was Mihajlovic who took the *free-kick*, followed by a *save* by Van der Sar.

The merging component consists of several parts which will be described in more detail below.

4.1 Scenes

Given the example above, it is clear that matching together individual events from two different documents is not the right approach: a *save* event cannot be matched with a *free-kick* event, they are two totally different events. Besides, it is clear that players' names will play an important role in the matching of information from one document with information from another document. In order to take players' names into consideration, an *unknown* event was introduced, such that if it was not clear what a player did, this could be represented by letting that player perform this unknown event (the information extraction systems provide this information).

Now the event is still the fundamental concept, but the merging process aims at matching together *groups* of events instead of single events. Such a group of events is called a *scene*. The author of a text is considered as a "semantic filter", which determines which events should be taken together in the same scene. If events are mentioned in the same text fragment, they belong to the same scene. In the ticker documents this does not give rise to ambiguities, since their text fragments are clearly distinguished from each other (see Table 1).

4.2 Two Document Alignment

The merging algorithm compares all the scenes extracted from document *A* with all the scenes extracted from document *B*, and examines whether or not a scene from document *A* might be matched with a scene from document *B*. There are several aspects to be taken into account: players involved in the scenes, distance between time stamps, whether the scenes contain the same events or not, etc.

A scene from document *A* may match more than one scene from document *B* (see Figure 1).

The strength of a matching can be calculated in different ways, and it will not be surprising that the number of players involved in both scenes will be of great influence. This influence is that great, that taking only the number of common players' names as a measure of the strength of a binding, gave the best results. We restricted matching of scenes to those scenes which were not further than (arbitrarily) five minutes apart.

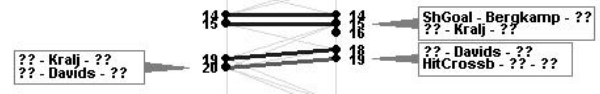


Figure 1: Two document alignment. Vertical lines denote documents, numbers are time stamps, thin lines indicate possible bindings, thick lines denote strongest bindings.

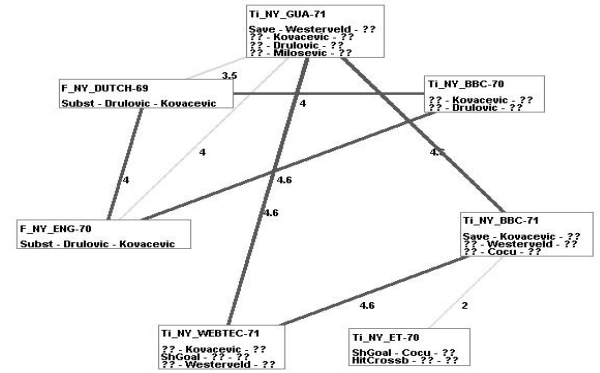


Figure 2: Finding complete subgraphs. The graph as a whole is found by the alignment process. Inside 3 complete subgraphs are found (two triangles indicated by the thick lines, and one single node).

In order to choose the best matchings, the algorithm starts by selecting two scenes s_1 and s_2 such that the binding between s_1 and s_2 is (one of) the strongest. In general, such a choice will remove other bindings, in particular the bindings between s_1 and s_2 with other scenes. Since these two describe the same fragment of the game, scenes before s_1 can now no longer match scenes after s_2 (and vice versa). That is to say, bindings which "cross" the matching of s_1 and s_2 are removed. Thus the possible bindings are cut in two parts, and the algorithm continues recursively with both halves, until all choices have been made. This two document algorithm is applied to every pair of documents.

4.3 Multi Document Alignment

The next step is to join connected scenes from various documents together. Start with a set consisting of one (arbitrary) scene, and extend this set by those scenes which are connected to the starting scene. Repeat this for all the scenes added to the set until no further extension is found. This set of scenes, together with the bindings between the scenes (chosen by the two document algorithm) naturally form a *connected graph*.

Repeat this graph building procedure for the remaining scenes, until all scenes are included in a graph. Notice, that such a graph may consist of one scene only. Notice also, that a graph may contain more scenes than there are documents, since scenes may be connected through a sequence of two document matches.

4.4 Complete Sub-graphs

Ideally, a graph should be *complete*, expressing that every two scenes in the graph do match, and thus all scenes do contain some common information. That is to say, the scenes in a complete graph are all about the same fragment during the football game which is described in the documents. However, in practice not every graph which results from the procedure above, is complete (see Figure 2).

Scenes may partially overlap, and thus give rise to sequences of connections where the first and the last are no longer connected. We isolate the *strongest complete sub-graphs* from every non-complete graph, since such a sub-graph describes one fragment in reality. A given (non-complete) graph is divided into its strongest complete sub-graphs by going through all the bindings in the graph, ordered by their strength (starting with the strongest one), and adding scenes and edges to a sub-graph whenever possible without violating the completeness of this sub-graph. If that is not possible, a new sub-graph is started, and some bindings may be removed. The final result is a set of complete graphs of matching scenes from many documents, such that the scenes inside a graph describe the same fragment in reality.

4.5 Rules

Consider the example given before, about the *save* event and the *free-kick* event, with the players Van der Sar and Mihajlovic. These scenes are now in the same graph, and thus may be combined (see Figure 3). However, not every combination will be correct, for example, Mihajlovic and Van der Sar will not both take the free-kick. Also, it will not be correct to let Van der Sar (the Dutch keeper) take the free kick, and Mihajlovic perform the save. In order to combine this partial information into scenes containing complete events, *rules* are needed. In the MUMIS project several kinds of rules, all expressing some *domain knowledge*, have been developed. A first kind are the event internal rules, for example, rules saying that the two players involved in a substitution event belong to the same team, or that a keeper typically will not take a corner.

A second kind of rule takes into account the role of the teams in the fragment, i.e., whether a team is attacking or defending at that moment. To determine whether a team is attacking or defending, all players involved in the scene are checked for their normal position in the field. Then events are characterised as offensive events (such as *corner*, *shot on goal*), defensive events (*save*, *clearance*), and neutral events (*throw-in*, *yellow card*). The second kind of rule makes sure that offensive events are performed by players from the attacking team, and defensive events are combined with payers from the defending team.

A third kind of rule makes sure that players will not perform impossible combinations of events, e.g., the player who takes a corner will (unless he's very fast) not deliver a shot-on-goal as well.

To apply these rules, a background database of player information is created, containing names of players, their normal position in the field, the team they belong to, etc.

Yet another kind of rule is based on an ontology of events and is used to unify certain events which according to the

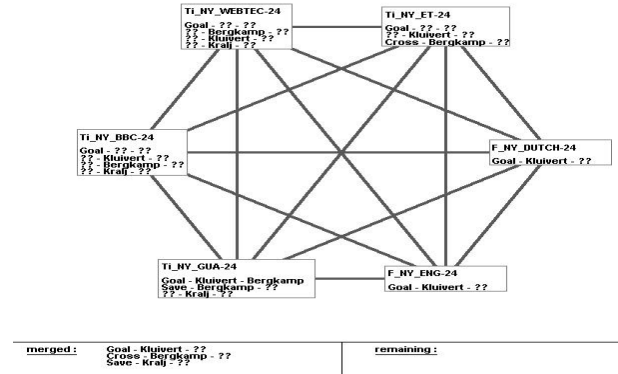


Figure 3: Merging. Bottom left is the result of merging all information in all nodes of the graph. Bottom right is the remaining information that could not be decided upon (in this case nothing).

chosen semantics in MUMIS are not the same. For example, a clearance and a save are two different events according to the MUMIS semantics, but an author may well use these terms in a way different from the MUMIS semantics. Therefore the relationships between such concepts are expressed in an ontology, such that rules may use sub-typing or super-typing relationships between concepts.

4.6 Scenarios

After merging the scenes extracted from the various documents into more complete scenes, the order of the events within a scene may be incorrect with respect to the order as it was in reality. The merging process itself does not take the ordering of events into account, and besides, authors often mention events in the opposite order. For example, a player scores after a corner – but the scoring is mentioned first.

Based on the original texts in the source documents, a series of *scenarios* is extracted, describing typical orders in which events occur [Schank and Abelson, 1977].

4.7 Evaluation

Given the limited availability of background information on players (to which team they belong, on what position they play, etc.), the merging component could only be tested in a *case study*. The match Netherlands-Yugoslavia from the European championships in 2000 to perform such a case study. Based on this example, the results of the merging approach are very promising.

The result of the alignment process applied to this match produced 63 complete graphs, of which 28 consist of only one node (a scene), containing information from one source text only. Looking at the original texts, only five of these one-scene graphs might have been combined with other graphs, the main problem being that some texts mention some information in one scene, whereas other documents divide the same information over several scenes. In such cases the alignment algorithm chooses the best match, leaving the other part unmatched.

Of the remaining 35 multi scene graphs, no graph contained unjustified bindings. This seems to be a very promising aspect, since the rules used in the graph forming part of the algorithm are formulated in a very general way, without using any specific information on the concrete football match.

The result of the merging process consisted partly of the elimination of several errors caused by syntactical ambiguities in the single document information extraction, and partly of joining together partial events into more complete events. Like the alignment process, this merging process is also based on general semantical rules, which do not use any concrete details about the specific football match taken as a case study. There were no errors introduced by these merging rules, and in many cases the quality of the extracted information improved considerably.

5 Conclusions and Future Work

The development of huge multimedia databases and the need for accurate navigation of its content require a new generation of “intelligent” tools for producing fine-grained surrogates or indices of the multimedia content. By taking advantage of ontology, domain lexica, and a “language-free” representation of the contents, MUMIS facilitates conceptual search overcoming the keyword barrier.

MUMIS takes advantage of reliable, but coarse-grained content, obtained from formal texts and fine-grained, but sometimes partial, content obtained from free texts and transcriptions. It indirectly solves problems of incomplete information found in any source by combining results from multiple sources. By relying on the analysis of textual instead of visual sources, MUMIS makes possible the derivation of fine-grained semantic indices.

Cross-document co-reference is still in its infancy, MUMIS advances research in that direction by providing a methodology that uses robust named entity alignment from multiple sources together with domain specific, semantic rules.

There are still many points on which the merging algorithm may be improved. For example, some documents are more reliable than other documents. Furthermore, not all texts are of the same type, so-called formal texts differ from ticker texts, and from more free texts. As yet, differences in quality of the source documents has not been taken into account, and a weighted integration of formal texts and other texts still has to be performed. A related point of improvement is to check the consistency of certain elements of information in comparison to other elements of information, and in particular to the state on the football field as may be derived from all previously mentioned events. Finally we mention the improvements in connection to overlapping scenes, where events mentioned in one text fragment in one document may be spread over several text fragments in another document. At this point improvements are also possible.

Acknowledgements

We wish to thank Eduard Hoenkamp and Marco Puts for many useful discussions.

References

- [Abney, 1996] Steven Abney. Partial parsing via finite-state cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information (ESSLI)*, 1996.
- [Bagga and Baldwin, 1999] A. Bagga and B. Baldwin. Cross-document coreference: Annotations, Experiments, and Observations. In *Proceeding of the Workshop on Coreference and its Applications (ACL99)*, 1999.
- [Chang et al., 1998] S.F. Chang, W. Chen, H.J. Meng, H. Sundaram, and D. Zhong. A Fully Automated Content-based Video Search Engine Supporting Spatio Temporal Queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5), 1998, pp. 602–615.
- [Cunningham et al., 2002] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL2002*, 2002.
- [Declerck, 2002] Thierry Declerck. A set of tools for integrating linguistic and non-linguistic information. In *Proceedings of SAAKM 2002, ECAI-2002, Lyon*, 2002.
- [Gaizauskas et al., 1997] R. Gaizauskas, K. Humphreys, S. Azzam, and Y. Wilks. Concepticons vs. lexicons: An architecture for multilingual information extraction. In M.T. Pazienza, editor, *SCIE-97, LNCS/LNAI*, Springer-Verlag, 1997, pp. 28–43.
- [Grishman and Sundheim, 1996] R. Grishman and B. Sundheim. Message understanding conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June 1996.
- [De Jong et al., 2000] F.M.G. de Jong, J.-L. Gauvain, D. Hiemstra, K. Netter, Language-Based Multimedia Information Retrieval”. In *Proceedings of RIAO*, 2000, Paris, pp. 713-725.
- [Radev and McKeown, 1998] R. Radev and K.R. McKeown. Generating Natural Language Summaries from Multiple On-Line Sources. *Computational Linguistics*, 24(3), September 1998, pp. 469–500.
- [Sable and Hatzivassiloglou, 1999] C. Sable and V. Hatzivassiloglou. Text-based approaches for the categorization of images. In *Proceedings of ECDL*, 1999.
- [Schank and Abelson, 1977] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Publishers, 1977.
- [Srihari, 1995] R.K. Srihari. Automatic Indexing and Content-Based Retrieval of Captioned Images. *Computer*, 28(9), September 1995, pp. 49–56.
- [Steinbiss et al., 1993] V. Steinbiss, H. Ney, R. Haeb-Umbach, B.-H. Tran, U. Essen, R. Kneser, M. Oerder, H.-G. Meier, X. Aubert, C. Dugast, and D. Geller. The Philips Research System for Large-Vocabulary Continuous-Speech Recognition. In *Proc. of Eurospeech '93*, Berlin, Germany, 1993, pp. 2125–2128.

Active Learning with Strong and Weak Views: A Case Study on Wrapper Induction

Ion Muslea, Steven N. Minton, Craig A. Knoblock

muslea@isi.edu, minton@fetch.com, knoblock@isi.edu

U. of Southern California, Fetch Technologies, Inc., U. of Southern California,
4676 Admiralty Way
Marina del Rey, CA 90292

Abstract

Multi-view learners reduce the need for labeled data by exploiting disjoint sub-sets of features (*views*), each of which is sufficient for learning. Such algorithms assume that each view is a *strong view* (i.e., perfect learning is possible in each view). We extend the multi-view framework by introducing a novel algorithm, Aggressive Co-Testing, that exploits both strong and *weak views*; in a weak view, one can learn a concept that is strictly more general or specific than the target concept. Aggressive Co-Testing uses the weak views both for detecting the most informative examples in the domain and for improving the accuracy of the predictions. In a case study on 33 wrapper induction tasks, our algorithm requires significantly fewer labeled examples than existing state-of-the-art approaches.

1 Introduction

Labeling training data for learning algorithms is a tedious, error prone, time consuming process. Active learning addresses this issue by detecting and asking the user to label only the most informative examples in a domain. In this paper, we focus on Co-Testing [Muslea *et al.*, 2000], an active learning technique for domains with multiple *views*; i.e., domains with disjoint sub-sets of features, each of which is sufficient for learning. Co-Testing is a 2-step iterative algorithm that (1) uses the few available labeled examples to learn a hypothesis in each view and (2) queries (i.e., asks the user to label) examples on which the views predict a different label. Such queries are highly informative because they correct mistakes made by one of the views: whenever the views disagree, at least one of them must be wrong.

Co-Testing was successfully applied to wrapper induction [Muslea *et al.*, 2000], an industrially important application. In wrapper induction the goal is to learn rules that extract the relevant data from collections of Web pages that share the same underlying structure; e.g., extract the book titles and prices from `amazon.com`. For wrapper induction, Co-Testing uses two views: the sequences of tokens that precede and follow the extraction point, respectively. The extraction rules learned in these views are finite automata that consume an item's prefix or suffix within the page, respectively.

The main limitation of existing Co-Testing algorithms [Muslea *et al.*, 2000; 2002a] is that they are designed to use only views that are adequate for learning, thus being unable to also exploit imperfect views that would permit a faster convergence to the target concept. To address this problem, we extend the multi-view learning framework by introducing the idea of learning from *strong* and *weak* views. By definition, a *strong view* consists of features that are adequate for learning the target concept; in contrast, in a *weak view* one can only learn a concept that is more general or specific than the target concept. We introduce a novel algorithm, *Aggressive Co-Testing*, that exploits both strong and weak views without additional data engineering costs. We also describe a case study on wrapper induction, which shows that Aggressive Co-Testing clearly outperforms state-of-the-art algorithms.

To illustrate the idea of strong and weak views, consider the task of extracting fax numbers from a directory of restaurant Web pages such as Zagat. The two wrapper induction views described above are *strong views* because each of them is (typically) sufficient to extract the item of interest [Muslea *et al.*, 2000]. In addition to these two strong views, we can also exploit a view that consists of tokens *within* the item to be extracted. In this view, we learn the grammar “(*Number*) *Number* - *Number*” that describes the content of the fax numbers. This additional view is a *weak view* because the grammar above represents a concept *more general* than the target one; i.e., it cannot discriminate between fax and phone numbers that appear within the same Web page.

Aggressive Co-Testing for wrapper induction works as follows: first, it uses a few labeled examples to learn a rule in each view (i.e., one weak and two strong rules). Then it queries an unlabeled example on which the two strong rules extract *different* strings, *both* of which are inconsistent with the content-based grammar. Each such query is likely to represent a mistake not only in one, but in both strong views, thus leading to faster convergence. We use a collection of 33 difficult extraction tasks to show that using the weak view dramatically reduces the need for labeled data: compared with existing state of the art active learners, our novel algorithm requires between 45% and 81% fewer labeled examples.

2 Related work

The idea of exploiting complementary information sources (i.e., types of features) appears in various multi-strategy

learners. Of particular interest are two recent papers [Kushmerick *et al.*, 2001; Nahm and Mooney, 2000] in which the authors use sets of features that clearly do *not* have the same expressive power. This work can be seen as learning from strong and weak views, even though it was not formalized as such, and it was not used for active learning.

Kushmerick *et al.* [2001] focus on classifying the lines of text on a business card as a person's name, affiliation, address, phone number, etc. In this domain, the strong view consists of the words that appear on each line, based on which a Naive Bayes text classifier is learned. In the weak view, one can exploit the relative order of the lines on the card by learning a hidden Markov Model that predicts the probability of a particular ordering of the lines on the business card (e.g., name followed by address, followed by phone number).

This weak view defines a class of concepts that is *more general* than the target concept: all line orderings are possible, even though they are not equally probable. The order of the text lines cannot be used by itself to accurately classify the lines. However, when combined with the strong view, the ordering information leads to a classifier that clearly outperforms the stand-alone strong view [Kushmerick *et al.*, 2001].

Another algorithm that can be seen as learning from strong and weak views is DISCOTEX [Nahm and Mooney, 2000], which extracts job titles, salaries, locations, *etc* from computer science job postings to the newsgroup `austin.jobs`. DISCOTEX proceeds in four steps: first, it uses RAPIER [Califf and Mooney, 1999] to learn extraction rules for each item of interest. Second, it applies the learned rules to a large, unlabeled corpus of job postings and creates a database that is populated with the extracted data. Third, by text mining this database, DISCOTEX learns to predict the value of each item based on the values of the other fields. For example, it may discover that "IF the job requirements include C++ and CORBA THEN the development platforms include Windows". Finally, when the system is deployed and the RAPIER rules fail to extract an item, the mined rules are used to predict the item's content.

In this scenario, the RAPIER rules represent the *strong view* because they are sufficient for extracting the data of interest. In contrast, the mined rules represent the *weak view* because they cannot be learned or used by themselves. Furthermore, as DISCOTEX discards all but the most accurate of the mined rules, which are highly-specific, it follows that the weak view can be used to learn only concepts that are *more specific* than the target concept. Nahm and Mooney show that these mined rules improve the extraction accuracy by capturing information that complements the RAPIER extraction rules.

3 Preliminaries

In this section we first explain the main idea behind Co-Testing algorithms [Muslea *et al.*, 2000; Muslea, 2002], and then we describe the strong and weak views that we use for wrapper induction.

3.1 Background: the Co-Testing approach

Figure 1 provides a formal description of the Co-Testing family of algorithms. Given a base learner $\mathcal{L}$, a set L of labeled

Given: - a base learner $\mathcal{L}$

- a learning problem with features $\mathbf{V}=\{a_1, a_2, \dots, a_N\}$
- two views $\mathbf{V1}$ and $\mathbf{V2}$, where $\mathbf{V}=\mathbf{V1} \cup \mathbf{V2}$ and $\mathbf{V1} \cap \mathbf{V2} = \emptyset$
- the sets L and U of labeled and unlabeled examples
- number N of queries to be made

LOOP for N iterations

- use $\mathcal{L}$, $\mathbf{V1}(L)$, and $\mathbf{V2}(L)$ to create classifiers h_1 and h_2
- let $ContentionPoints = \{ u \in U \mid h_1(u) \neq h_2(u) \}$
- IF $ContentionPoints$ is empty THEN quit
- let $Query = \text{SelectQuery}(ContentionPoints)$
- remove $Query$ from U and ask for its label l
- add labeled $Query$ to L

- CreateOutputHypothesis(h_1, h_2)

Figure 1: Co-Testing algorithms repeatedly query examples for which the two views make a different prediction.

examples, and a set U of unlabeled ones, Co-Testing works as follows: first, it learns the classifiers h_1 and h_2 by applying the algorithm $\mathcal{L}$ to the projection of the examples in L onto the two views, $\mathbf{V1}$ and $\mathbf{V2}$. Then it applies h_1 and h_2 to all unlabeled examples in U and detects the set of *contention points*, which are unlabeled examples for which h_1 and h_2 predict a different label. Finally, it asks the user to label one of the contention points and repeats the whole process.

The various members of the Co-Testing family differ from each other with two respects: the strategy used to select the next query, and the manner in which the *output hypothesis*¹ is constructed. In other words, each Co-Testing algorithm is uniquely defined by the choice of the heuristics **SelectQuery()** and **CreateOutputHypothesis()**. In turn, these two heuristics depend on the properties of both the application domain and the base learner $\mathcal{L}$.

We consider here two types of query selection strategies:

- *random*: randomly choose a contention point. This strategy is appropriate for base learners that lack the capability of estimating the confidence of their predictions.
- *max-confidence*: choose the contention point on which both h_1 and h_2 make the most confident prediction. This strategy is appropriate for high accuracy domains (e.g., wrapper induction), in which there is little or no noise. On such tasks, discovering examples that are misclassified "with high confidence" translates into queries that "fix big mistakes," thus leading to fast convergence.

We also consider two "output hypothesis" heuristics:

- *winner-takes-all*: the output hypothesis is the one learned in the view that makes the smallest number of mistakes over the N queries.
- *majority vote*: examples are labeled according to the prediction of most views (requires at least three views).

3.2 Wrapper induction: the strong views

In wrapper induction, each item of interest is described by three strings of variable length: the item's content, together

¹Once training is completed, the *output hypothesis* is used to predict the label of all new, unseen examples.

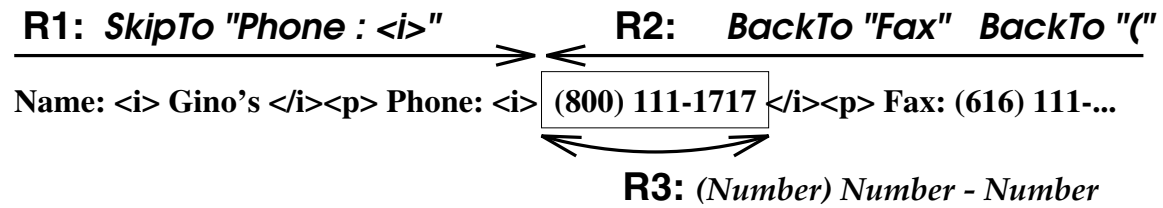


Figure 2: The *forward* and *backward* strong rules (i.e., **R1** and **R2**) find the beginning of the phone number by consuming its suffix or prefix, respectively. **R3** is a content-based grammar that describes the structure of the item to be extracted.

with its prefix and suffix within the document. As this is not a typical machine learning representation in which an example's description in each view consists of a fixed set of features, we describe here in detail how Co-Testing can be applied to wrapper induction. As a first step, we introduce the basic ideas in STALKER [Muslea *et al.*, 2001], which is the wrapper induction algorithm that we use as base learner.

Consider the illustrative task of extracting phone numbers from Web pages similar to the one shown in Figure 2. In STALKER, an *extraction rule* consists of a *start rule* and an *end rule* that identify the beginning and the end of the item, respectively. Given that start and end rules are extremely similar, we describe here only the former. In order to find the beginning of the phone number, one can use the start rule

R1 = SkipTo "Phone :<i>"

This rule is applied *forward*, from the beginning of the document, and it ignores everything until it finds the string Phone:<i>. For a slightly more complicated extraction task, in which toll-free numbers appear in italics and the other ones in bold, one can use a disjunctive start rule such as

R1' = *either* SkipTo "Phone :<i>"
or SkipTo "Phone :"

An alternative way to detect the beginning of the phone number is to use the start rule

R2 = BackTo "Fax" BackTo "("

which is applied *backward*, from the *end* of the document. **R2** ignores everything until it finds "Fax" and then, again, skips back to the first open parenthesis.

As shown in [Muslea *et al.*, 2001], the above extraction rules can be learned based on user-provided examples of items to be extracted. Note that **R1** and **R2** represent descriptions of the *same concept* (i.e., start of phone number) in two *different views*. That is, the views **V1** (*forward view*) and **V2** (*backward view*) consist of the sequences of tokens that *precede* and *follow* the beginning of the item, respectively.

Note that both **V1** and **V2** represent *strong views*: as the Web pages to be wrapped share the same underlying structure, STALKER can be seen as uncovering and exploiting this underlying structure for extraction purposes. Consequently, both the forward and backward rules are expected to extract the relevant data from any page.

3.3 Wrapper induction: the weak view

Besides the two strong views above, one can also use a third, content-based view, which describes the actual item to be extracted. For example, when extracting phone numbers, one

may exploit the fact that they can be described by a simple grammar: "(Number) Number - Number". Similarly, when extracting URLs, one can take advantage of the fact that a typical URL starts with the string "http://www.", ends with the string ".html", and contains no HTML tags.

In this paper, we use the following features to describe the content of each item to be extracted:

- the *length range* (in tokens) of the seen examples. For instance, phone numbers in the format "(Number) Number - Number" consist of six tokens (i.e., the three numbers, the dash, and the two parentheses).
- the *token types* that appear in the training examples. This feature consists of the set of the most specific *wildcards* (e.g., *Number*, *AllCaps*, etc) that match the tokens encountered in the item to be extracted. For example, in the phone number case, this list consists of two wildcards: *Number* and *Punctuation*. The complete hierarchy of wildcards is described in Figure 3.
- a *start-pattern* such as "http://www." or "(Number)", which describes the beginning of the item of interest.
- an *end-pattern* such as "AlphaNum.html" or "Number - Number", which describes the end of the item.

In order to learn the content-based description of an item, we use as base learner a simplified version of the DataPro algorithm [Lerman and Minton, 2000]. After tokenizing each of the user-provided examples of strings to be extracted, the weak-view learner proceeds as follows:

- the *length range* is determined by finding the examples that contain the largest and the smallest number of tokens;
- the *token types* are obtained by going through the tokens that appear in the labeled examples and adding to the set of "seen types" the most specific wildcard that covers it.
- a start-pattern of *length one* consists of the *most specific* wildcard that covers the first token in *all* labeled examples; if all examples start with the same token, such as "(" in the phone number example, the actual token is preferred to the most specific wildcard. A start-pattern of length *k* is generated by repeating the procedure above for the first, second, ..., up to *k*-th position.
- the end-pattern is learned in the same manner as the start pattern, but using the *k* tokens at the end of the item.

Note that, unlike the forward and backward views, the content-based view is a *weak view* because, for many extraction tasks, this view does not uniquely define the item of inter-

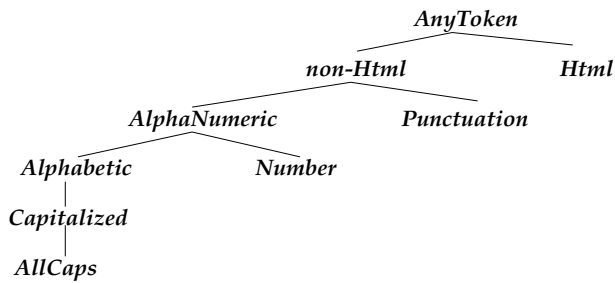


Figure 3: The hierarchy of wildcards used for wrapper induction. The parent-child relationship denotes the *IsMoreGeneralThan* relationship. For example, the most general wildcard is *AnyToken*, which matches all possible tokens. *non-HTML*, which is a child of *AnyToken*, denotes all tokens than are not HTML tags (i.e., alphanumeric tokens and punctuation signs).

est. This is a consequence of the fact the view uses only features that describe the content of each item. For Web pages that contain several items with similar descriptions, such as multiple email addresses, phone numbers, URLs, or names, the content-based grammar cannot discriminate between the various items with similar descriptions.

4 Aggressive Co-Testing

We introduce now Aggressive Co-Testing, which provides a framework for naturally exploiting both strong and weak views. For Aggressive Co-Testing, the contention points are defined as unlabeled examples on which the *strong views* predict a different label. For the wrapper induction problem, Aggressive Co-Testing uses the two strong and one weak views described earlier (i.e., the forward, backward, and content-based views). Consequently, the contention points are unlabeled documents from which the forward and backward rules extract different strings. Aggressive Co-Testing uses the labeled examples to learn one hypothesis in each view, detects the contention points, and then uses the following heuristics:

- **SelectQuery()** returns the contention point on which both strong rules violate the largest number of constraints learned in the weak view; e.g., the extracted strings are longer than the seen examples, the start- and end- patterns do not match, etc. This is a *max-confidence* querying strategy because the content-based view is *maximally confident* that the strong rules extract incorrect strings.
- **CreateOutputHypothesis()** uses the three views for *majority voting*. That is, given a new, unseen document, both strong rules are applied to it; if they extract the same string, this string is returned as the answer. Otherwise the “winner” is the strong rule that violates the fewest constraints learned in the weak view. Note that this flexible approach allows Co-Testing to use the most appropriate strong rule for each document in the dataset.

To better understand how Aggressive Co-Testing works, we contrast it now with Naive Co-Testing [Muslea *et al.*, 2000], which uses only the two strong views. Both algorithms detect the contention points in the same manner, but

they use different query selection strategies and output hypotheses. More precisely, Naive Co-Testing *randomly* queries one of the contention points and generates a *winner-takes-all* output hypothesis (i.e., the rule that makes the fewest mistakes on the queries extracts the data from *all* documents).

5 Empirical Evaluation

The algorithms in the experimental comparison

In this empirical evaluation we compare the following algorithms: Aggressive Co-Testing, Naive Co-Testing, Query-by-Bagging, and Random Sampling. The first two algorithms were described in the previous section; Random Sampling, which is used as strawman, is identical with Naive Co-Testing, except that it randomly queries one of the unlabeled examples instead of one of the contention points.

Query-by-Bagging [Abe and Mamitsuka, 1998] is the only single-view active learner that can be used in a straightforward manner with STALKER and, more generally, for wrapper induction.² Even Query-by-Boosting [Abe and Mamitsuka, 1998], which is similar to Query-by-Bagging, cannot use STALKER as a base learner: as STALKER rarely - if ever - makes mistakes on small training sets, it eliminates the ability of the boosting algorithm to generate a diverse committee.

Query-by-Bagging is based on the idea of creating a *committee* of extraction rules and then querying the example on which the committee is the most split (i.e., the rules in the committee extract the largest number of distinct strings); the algorithm’s actual predictions are made by *majority voting* the committee of rules. Query-by-Bagging generates a committee of 10 extraction rules, each of which is learned by training STALKER with examples obtained by *re-sampling with replacement* the original training set. We are forced to use such a small committee because of the scarcity of the training data: as STALKER is expected to train on a handful of examples, sampling-with-replacement from a few examples leads to few distinct training sets for creating the committee. Query-by-Bagging is run once in each view, and we report only the best results, which are obtained in the forward view.

The datasets

In order to empirically compare the algorithms above, we use the wrapper induction testbed introduced by Kushmerick [2000]. It consists of 206 extraction tasks from 30 Web-based information sources.³ As shown in [Muslea *et al.*, 2001], on most of these 206 tasks STALKER learns 100% accurate rules from just one or two randomly-chosen labeled examples. We consider here the 33 most difficult tasks in the testbed, which were also used in previous work on multi-view learning [Muslea *et al.*, 2000; 2002b]:

- the 28 tasks on which 20 random examples are insufficient for learning 100%-accurate rules in both strong views;

²Typical wrapper induction algorithms do *not* have the properties that active learners require of their base learners; e.g., the ability to evaluate the confidence of each prediction [Lewis and Gale, 1994], or to randomly sample hypotheses from the version space [Seung *et al.*, 1992], or to generate *most specific* and *most general* extraction rules [Cohn *et al.*, 1994].

³These datasets can be obtained from the RISE repository: <http://www.isi.edu/~muslea/RISE/index.html>.

- the five additional tasks on which, in order to learn 100%-accurate rules in both strong views, STALKER requires a large number of random examples [Muslea, 2002].

The empirical results

For each of these 33 tasks, we use 20-fold cross-validation to compare the performance of the algorithms above. Within each fold, the algorithms start with *the same* two randomly-chosen examples and then make a succession of queries. In the end, the error rate is averaged over the 20 folds.

Figure 4 summarizes the algorithms' performance over the 33 tasks. In each graph, the X axis shows the number of queries made by the algorithm, while the Y axis shows the number of tasks for which a 100% accurate rule was learned after exactly X queries. Each algorithm is allowed to make 18 queries, for a total of 20 labeled examples. *By convention*, the "19 queries" data point denotes tasks for which a 100% accurate rule is *not* learned even after 18 queries.

Aggressive Co-Testing clearly outperforms the other algorithms: it makes an average of 2.43 queries over the 30 tasks that are solved with 100% accuracy; furthermore, on 11 of these 30 tasks, a single query is sufficient to learn the correct rule. In contrast, Naive Co-Testing, which comes second, makes an average of 4.4 queries per task and converges in a single query on just four of the 33 tasks. Also note that Aggressive Co-Testing solves correctly two of the five tasks that cannot be solved by Naive Co-Testing; the other two algorithms fail to solve 23 and 26 of the 33 tasks, respectively.

Even though Aggressive Co-Testing makes 45% fewer queries than Naive Co-Testing, at first glance the difference between 2.43 and 4.4 queries per task may seem small. However, one must take into account that wrapper induction is used in information agents [Knoblock *et al.*, 2001], which typically use hundreds of extraction rules; in this context, Aggressive Co-Testing makes a tremendous difference.

To put our work into a larger context, we briefly compare the results above with the ones obtained by WIEN [Kushmerick, 2000], which is the only wrapper induction system for which there are published results for all the extraction tasks used here. As the two experimental setups are not identical,⁴ this is just an *informal* comparison that contrasts Co-Testing with a state-of-the-art approach to wrapper induction.

The results in [Kushmerick, 2000] can be summarized as follows: WIEN, which uses random sampling, learns the correct extraction rule on 15 of the 33 task. On these 15 tasks, WIEN requires between 25 and 90 labeled examples⁵ to learn the correct rule. For the same 15 tasks, both Aggressive and Naive Co-Testing learn 100% accurate rules from at most eight labeled examples (two random plus at most six queries).

⁴Instead of using cross-validation, WIEN repeatedly splits the dataset into randomly chosen training and test sets.

⁵For WIEN, an example consists of a document in which *all* items of interest are labeled; e.g., a page with 15 labeled names represents a *single* example. In contrast, STALKER counts the 15 labeled strings as 15 examples. We convert the WIEN results into equivalent STALKER-like ones by multiplying the number of WIEN labeled pages by the average number of item occurrences per page.

Discussion

The empirical results deserve several comments. First of all, the experiments illustrate the benefits of a framework that naturally integrates strong and weak views: Aggressive Co-Testing exploits the strengths and mitigates the weaknesses of each individual view. For example, we do *not* use the weak view to identify the contention points because its mistakes may be "unfixable" (remember that in a weak view one learns a concept more general/specific than the one of interest). On the other hand, we use the weak view both to detect the highly informative contention points and to find the most appropriate strong view for each prediction.

In contrast to Aggressive Co-Testing, existing multi-view learners [Blum and Mitchell, 1998; Muslea *et al.*, 2000] can use only the strong views, thus losing an important source of information. Similarly, single-view learners must either pool all features together or simply ignore all but one view. Note that, in practice, pooling the features together may not be a straightforward task:

- in DISCOTEX [Nahm and Mooney, 2000], the text mining features (the weak view) are the extracted items, which become available only after the extraction rules are learned and applied to the unlabeled corpus.
- the main contribution of [Kushmerick *et al.*, 2001] consists of a novel algorithm that exploits features from both the strong and weak views (i.e., the words in each text line, and the lines's order within the business card).

Second, we ran an additional experiment to determine the usefulness of the weak view with respect to each heuristic (i.e., query selection and output hypothesis). We considered two hybrid algorithms between Aggressive and Naive Co-Testing: one uses the *random* and *majority vote* heuristics, while the other uses *max-confidence* and *winner-takes-all*; i.e., each hybrid exploits the weak view in only one of the two heuristics. Because of space constraints, we just summarize our findings: the two hybrids outperform Naive and underperform Aggressive Co-Testing; more precisely, they make an average of 3.0 and 3.9 queries per task, respectively. In other words, the weak view improves both the query selection and the output hypothesis.

Finally, note that on three of the 33 tasks, Aggressive Co-Testing fails to learn 100%-accurate rules; in fact, on these tasks Query-by-Bagging and Random Sampling obtain more accurate rules than both Aggressive and Naive Co-Testing. This happens because, on these three tasks, the *backward* view is significantly less accurate than the *forward* one. Consequently, the distribution of the queries is skewed towards mistakes of the "bad view", which are not informative for either view: the "good view" extracts the correct string anyway, while the "bad view" is inadequate to learn the target concept. To cope with this problem, we plan to use view validation [Muslea *et al.*, 2002b], which predicts whether or not the strong views are appropriate for a particular task.

6 Conclusion and Future Work

In this paper we introduce the concepts of strong and weak views and present a novel active learner that naturally integrates and exploits both types of views. In a case study on

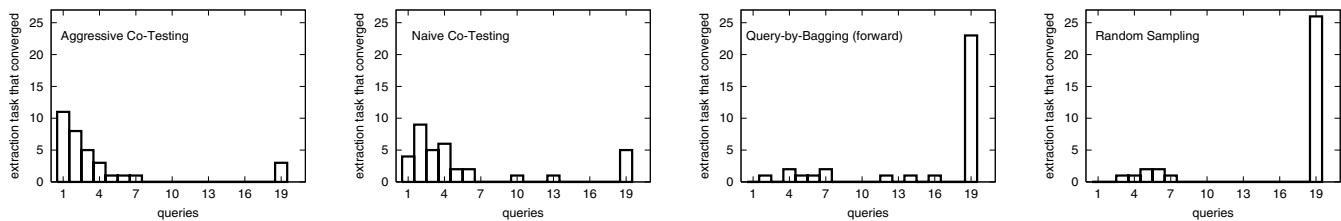


Figure 4: Convergence results on the 33 wrapper induction tasks.

wrapper induction, we show that weak views represent a powerful source of information that can be used both to detect highly informative examples and to improve the algorithm’s predictions. On a set of 33 difficult extraction tasks, our novel algorithm converges by making up to 81% fewer queries than other state of the art active learners.

We intend to continue our work along several directions. First, we plan to investigate the use of weak views for semi-supervised multi-view learning [Blum and Mitchell, 1998]. Second, we intend to extend view validation [Muslea *et al.*, 2002b] so that it also accounts for weak views. Finally, we are interested in a theoretical analysis of learning from strong and weak views.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copy right annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

- [Abe and Mamitsuka, 1998] Naoki Abe and Hiroshi Mamitsuka. Query learning using boosting and bagging. In *Proceedings of ICML-98*, pages 1–10, 1998.
- [Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT-98*, pages 92–100, 1998.
- [Califf and Mooney, 1999] Mary Elaine Califf and Raymond Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI-99*, pages 328–334, 1999.
- [Cohn *et al.*, 1994] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.
- [Knoblock *et al.*, 2001] Craig Knoblock, Steven Minton, Jose-Luis Ambite, Naveen Ashish, Ion Muslea, and Andrew Philpot. The Ariadne approach to Web-based Information Integration. *International Journal of Cooperative Information Sources*, 10(1/2):145–169, 2001.
- [Kushmerick *et al.*, 2001] Nicholas Kushmerick, Edward Johnston, and Stephen McGuinness. Information extraction by text classification. In *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.
- [Kushmerick, 2000] Nicholas Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence Journal*, 118(1-2):15–68, 2000.
- [Lerman and Minton, 2000] Kristina Lerman and Steven Minton. Learning the common structure of data. In *Proceedings of AAAI-2000*, pages 609–614, 2000.
- [Lewis and Gale, 1994] David Lewis and William Gale. A sequential algorithm for training text classifiers. In *Proceedings of Research and Development in Information Retrieval*, pages 3–12, 1994.
- [Muslea *et al.*, 2000] Ion Muslea, Steven Minton, and Craig Knoblock. Selective sampling with redundant views. In *Proceedings of AAAI-2000*, pages 621–626, 2000.
- [Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig Knoblock. Hierarchical wrapper induction for semistructured sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.
- [Muslea *et al.*, 2002a] Ion Muslea, Steven Minton, and Craig Knoblock. Active + Semi-supervised Learning = Robust Multi-view Learning. In *Proceedings of ICML-2002*, pages 435–442, 2002.
- [Muslea *et al.*, 2002b] Ion Muslea, Steven Minton, and Craig Knoblock. Adaptive view validation: A first step towards automatic view detection. In *Proceedings of ICML-2002*, pages 443–450, 2002.
- [Muslea, 2002] Ion Muslea. *Active Learning with Multiple Views*. PhD thesis, Department of Computer Science, University of Southern California, 2002.
- [Nahm and Mooney, 2000] Un-Yong Nahm and Raymond Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of AAAI-2000*, pages 627–632, 2000.
- [Seung *et al.*, 1992] H. Sebastian Seung, Manfred Oppel, and Haim Sompolinski. Query by committee. In *Proceedings of COLT-1992*, pages 287–294, 1992.

Bayesian Information Extraction Network

Leonid Peshkin and Avi Pfeffer

Harvard University
Cambridge, MA 02138, USA
{pesha,avi}@eecs.harvard.edu

Abstract

Dynamic Bayesian networks (DBNs) offer an elegant way to integrate various aspects of language in one model. Many existing algorithms developed for learning and inference in DBNs are applicable to probabilistic language modeling. To demonstrate the potential of DBNs for natural language processing, we employ a DBN in an information extraction task. We show how to assemble wealth of emerging linguistic instruments for shallow parsing, syntactic and semantic tagging, morphological decomposition, named entity recognition etc. in order to incrementally build a robust information extraction system. Our method outperforms previously published results on an established benchmark domain.

1 Information Extraction

Information extraction (IE) is the task of filling in template information from previously unseen text which belongs to a pre-defined domain. The resulting database is suited for formal queries and filtering. IE systems generally work by detecting patterns in the text that help identify significant information. Researchers have shown [Freitag and McCallum, 1999; Ray and Craven, 2001] that a probabilistic approach allows the construction of robust and well-performing systems. However, the existing probabilistic systems are generally based on Hidden Markov Models (HMMs). Due to this relatively impoverished representation, they are unable to take advantage of the wide array of linguistic information used by many non-probabilistic IE systems. In addition, existing HMM-based systems model each target category separately, failing to capture relational information, such as typical target order, or the fact that each element only belongs to a single category. This paper shows how to incorporate a wide array of knowledge into a probabilistic IE system, based on dynamic Bayesian networks (DBN)—a rich probabilistic representation that generalizes HMMs.

Let us illustrate IE by describing seminar announcements which got established as one of the most popular benchmark domains in the field [Califf and Mooney, 1999; Freitag and McCallum, 1999; Soderland, 1999; Roth and Yih, 2001; Ciravegna, 2001]. People receive dozens of seminar announcements weekly and need to manually extract informa-

tion and paste it into personal organizers. The goal of an IE system is to automatically identify target fields such as location and topic of a seminar, date and starting time, ending time and speaker. Announcements come in many formats, but usually follow some pattern. We often find a header with a gist in the form “PostedBy: john@host.domain; Who: Dr. Steals; When: 1 am;” and so forth. Also in the body of the message, the speaker usually precedes both location and starting time, which in turn precedes ending time as in: ‘‘Dr. Steals presents in Dean Hall at one am.’’ The task is complicated since some fields may be missing or may contain multiple values.

This kind of data falls into the so-called semi-structured text category. Instances obey certain structure and usually contain information for most of the expected fields in some order. There are two other categories: free text and structured text. In *structured text*, the positions of the information fields are fixed and values are limited to pre-defined set. Consequently, the IE systems focus on specifying the delimiters and order associated with each field. At the opposite end lies the task of extracting information from *free text* which, although unstructured, is assumed to be grammatical. Here IE systems rely more on syntactic, semantic and discourse knowledge in order to assemble relevant information potentially scattered all over a large document.

IE algorithms face different challenges depending on the extraction targets and the kind of the text they are embedded in. In some cases, the target is uniquely identifiable (single-slot), while in others, the targets are linked together in multi-slot association frames. For example, a conference schedule has several slots for related speaker, topic and time of the presentation, while a seminar announcement usually refers to a unique event. Sometimes it is necessary to identify each word in a target slot, while some benefit may be reaped from partial identification of the target, such as labeling the beginning or end of the slot separately. Many applications involve processing of domain-specific jargon like *Internetese*—a style of writing prevalent in news groups, e-mail messages, bulletin boards and online chat rooms. Such documents do not follow a good grammar, spelling or literary style. Often these are more like a stream-of-consciousness ranting in which ascii-art and pseudo-graphic sketches are used and emphasis is provided by all-capitals, or using multiple exclamation signs. As we exemplify below, syntactic analysers easily fail on such

Position	1	2	3	4	5	6	7	8	9	10
Tag	<speaker>				<location>			<s-time>		
Phrase	<i>Doctor</i>	<i>Steals</i>	<i>Presents</i>	<i>in</i>	<i>Dean</i>	<i>Hall</i>	<i>at</i>	1	<i>am</i>	.
Lemma	Dr.		present	in		hall	at		am	
PoS	NNP	NNP(VB)	VB(NNS)	IN	NNP	NN(NNP)	IN	CD	NN(RB)	.
Syn.segm.	NP	NP (VP)	VP	PP	NP	NP	PP	NP	NP(VP)	
Semantic	Title	LstName			LstName	Location			Time	
Length	3	6	8	2	4	4	2	1	2	1
Case	Upper	Upper	Upper	lower	Upper	Upper	lower		lower	

Table 1: Sample phrase and its representation in multiple feature values for ten tokens.

corpora.

Other examples of IE application domains include job advertisements [Califf and Mooney, 1999] (RAPIER), executive succession [Soderland, 1999] (WHISK), restaurant guides [Muslea *et al.*, 2001] (STALKER), biological publications [Ray and Craven, 2001] etc. Initial interest in the subject was stimulated by ARPA’s Message Understanding Conferences (MUC) which put forth challenges e.g. parsing newswire articles related to terrorism (see e.g. Mikheev [1998]). Below we briefly review various IE systems and approaches which mostly originated from MUC competitions.

Successful IE involves identifying abstract patterns in the way information is presented in text. Consequently, all previous work necessarily relies on some set of textual features. The overwhelming majority of existing algorithms operate by building and pruning sets of induction rules defined on these features (SRV, RAPIER, WHISK, LP²). There are many features that are potentially helpful for extracting specific fields, e.g. there are tokens and delimiters that signal the beginning and end of particular types of information. Consider an example in table 1 which shows how the phrase “Doctor Steals presents in Dean Hall at one am.” is represented through feature values. For example, the lemma “am” designates the end of a time field, while the semantic feature “Title” signals the speaker, and the syntactic category NNP (proper noun) often corresponds to speaker or location. Since many researchers use the seminar announcements domain as a testbed, we have chosen this domain in order to have a good basis of comparison.

One of the systems we compare to (specifically designed for single-slot problems) is SRV [Freitag, 1998]. It is built on three classifiers of text fragments. The first classifier is a simple look-up table containing all correct slot-fillers encountered in the training set. The second one computes the estimated probability of finding the fragment tokens in a correct slot-filler. The last one uses constraints obtained by rule induction over predicates like token identity, word length and capitalization, and simple semantic features.

RAPIER [Califf and Mooney, 1999] is fully based on bottom-up rule induction on the target fragment and a few tokens from its neighborhood. The rules are templates specifying a list of surrounding items to be matched and potentially, a maximal number of tokens for each slot. Rule generation begins with the most specific rules matching a slot. Then rules for identical slots are generalized via pair-wise merging,

until no improvement can be made. Rules in RAPIER are formulated as lexical and semantic constraints and may include PoS tags. WHISK [Soderland, 1999] uses constraints similar to RAPIER, but its rules are formulated as regular expressions with wild cards for intervening tokens. Thus, WHISK encodes a relative, rather than absolute position of tokens with respect to the target. This enables modeling long distance dependencies in the text. WHISK performs well on both single-slot and multi-slot extraction tasks.

Ciravegna [2001] presents yet another rule induction method (LP)². He considers several candidate features such as lemma, lexical and semantic categories and capitalization to form a set of rules for inserting tags into text. Unlike other approaches, (LP)² generates separate rules targeting the beginning and ending of each slot. This allows for more flexibility in subjecting partially correct extractions to several refinement stages, also relying on rule induction to introduce corrections. Emphasizing the relational aspect of the domain, Roth and Yoh [2001] developed a knowledge representation language that enables efficient feature generation. They used the features in a multi-class classifier SNOW-IE to obtain the desired set of tags. The resulting method (SNOW-IE) works in two stages: the first filters out the irrelevant parts of text, while the second identifies the relevant slots.

Freitag & McCallum [1999] use hidden Markov models (HMM). A separate HMM is used for each target slot. No pre-processing or features is used except for the token identity. For each hidden state, there is a probability distribution over tokens encountered as slot-fillers in the training data. Weakly analogous to templates, hidden state transitions encode regularities in the slot context. In particular prefix and suffix states are used in addition to target and background slots to capture words frequently found in the neighborhood of targets. Ray&Craven [2001] make one step further by setting HMM hidden states in a product space of syntactic chunks and target tags to model the text structure. The success of the HMM-based approaches demonstrate the viability of probabilistic methods for this domain. However, they do not take advantage of the linguistic information used by the other approaches. Furthermore, they are limited by using a separate HMM for each target slot, rather than extracting data in an integrated way.

The main contribution of this paper is in demonstrating how to integrate various aspects of language in a single probabilistic model, to incrementally build a robust information extraction system based on a Bayesian network. This sys-

tem overcomes the following dilemma. It is tempting to use a lot of linguistic features in order to account for multiple aspects of text structure. However, deterministic rule induction approaches seem vulnerable to the performance of feature extractors in pre-processing steps. This presents a problem since syntactic instruments that have been trained on highly-polished grammatical corpora, are particularly unreliable on weakly grammatical semi-structured text. Furthermore incorporating many features complicates the model which often has to be learned from sparse data, which harms performance of classifier-based systems.

2 Features

Our approach is statistical, which generally speaking means that learning corresponds to inferring frequencies of events. The statistics we collect originates in various sources. Some statistics reflect regularities of the language itself, while others correspond to the peculiarities of the domain. With this in mind we design features which reflect both aspects. There is no limitation on the possible set of features. Local features like part-of-speech, number of characters in the token, capitalization and membership in syntactic phrase are quite customary in the IE. In addition one could obtain such characteristics of the word as imaginability, frequency of use, familiarity, or even predicates on numerical values. Since there is no need for features to be local, one might find useful including frequency of a word in the training corpus or number of occurrences in the document. Notice that the same set of features would work for many domains. This includes semantic features along with orthographic and syntactic features.

Before we move on to presenting our system for probabilistic reasoning, let us discuss in some detail notation and methods we used in preliminary data processing and feature extraction. To use the data efficiently, we need to *factor* the text into “orthogonal” features. Rather than working with thousands of listemes (generic words¹) in the vocabulary, and combining their features, we compress the vocabulary by an order of magnitude by lemmatisation or stemming. Orthographic and syntactic information is kept in feature variables with just a few values each.

Tokenization

Tokenization is the first step of textual data processing. A token is a minimal part of text which is treated as a unit in subsequent steps. In our case tokenization mostly involves separating punctuation characters from words. This is particularly non-trivial for separating a period [Manning and Schutze, 2001] since it requires identifying sentence boundaries. Consider a sentence: *Speaker: Dr. Steals, Chief Exec. of rich.com, worth \$10.5 mil.*

Lemmatization

We have developed a simple lemmatiser which combines outcome of some standard lemmatisers and stemmers into a lookup table. Combined with lemmatisation is a step of spell

¹A *word* is a sequence of alphabetical characters, which has some meaning assigned to it. This would cover words found in general and special vocabulary as well abbreviations, proper names and such.

checking to catch misspelled words. This is done by interfacing with the UNIX *ispell* utility.

Gazetteer

Our original corpus contains about 11,000 different listemes. This does not take into account tokens consisting of punctuation characters, numbers and such. About 10% are proper nouns. The question of building a vocabulary automatically was previously addressed in IE literature (see e.g. Riloff [1996]). We use the intersection of two sets. The first set consists of words encountered as part of target fields and in their neighborhood. The second set consists of words frequently seen in the corpus. Aside from vocabulary there are two reserved values for Out-of-Vocabulary (OoV) words and Not-a-Word (NaW). For example see blank slots in the *lemma* row of Table 1. The first category encodes rare and unfamiliar words, which are still identified as words according to their part of speech. The second category is for mixed alphanumeric tokens, punctuation and symbolic tokens.

Syntactic Categories

We used LTChunk software from U.of Edinburgh NLP group [Mikheev *et al.*, 1998]. It produces 47 PoS tags from UPenn TreeBank set [Marcus *et al.*, 1994]. We have clustered these into 7 categories: cardinal numbers (CD), nouns (NN), proper nouns (NNP), verbs (VB), punctuation (.), preposition/conjunction (IN) and other (SYM). The choice of clusters seriously influences the performance, while keeping all 47 tags will lead to large CPTs and sparse data.

Syntactic Chunking

Following Ray&Craven [2001], we obtain syntactic segments (aka syntactic chunks) by running the *Sundance* system [Riloff, 1996] and flattening the output into four categories corresponding to noun phrase (NP), verb phrase (VP), prepositional phrase (PP) and other (N/A). Table 1 shows a sample outcome. Note that both the part-of-speech tagger and the syntactic chunker easily get confused by non-standard capitalization of a word “Presents” as shown by incorrect labels in parenthesis. “Steals” is incorrectly identified as a verb, whose subject is “Doctor” and object is “Presents”. Remarkably, other state-of-the-art syntactic analysis tools [Charniak, 1999; Ratnaparkhi, 1999] also failed on this problem.

Capitalization and Length

Simple features like capitalization and length of word are used by many researchers (e.g. SRV [Freitag and McCallum, 1999]) Case representation process is straightforward except for the choice of number of categories. We found useful introducing an extra category for words which contain both lower and upper case letters (not counting the initial capital letter) which tend to be abbreviations.

Semantic Features

There are several semantic features which play important role in a variety of application domains. In particular, it is useful to be able to recognize what could be a person’s name, geographic location, various parts of address, etc. For example, we are using a list of secondary location identifiers provided by US postal service, which identifies as such words like *hall*, *wing*, *floor* and *auditorium*. We also use a list of 100000 most

popular names from US census bureau; the list is augmented by rank which helps to decide in favor of first or last name for cases like “Alexander”. In general this task could be helped by using a hypernym feature of WordNet project [Fellbaum, 1998]. The next section presents probabilistic model which makes use of the aforementioned feature variables.

3 BIEN

We convert the IE problem into a classification problem by assuming that each token in the document belongs to a target *class* corresponding to either one of the target tags or the background (compare to Freitag [1999]). Furthermore, it seems important not to ignore the information about interdependencies of target fields and document segments. To combine advantages of stochastic models with feature-based reasoning, we use a Bayesian network.

A dynamic Bayesian network (DBN) is ideal for representing probabilistic information about these features. Just like a Bayesian network, it encodes interdependence among various features. In addition, it incorporates the element of time, like an HMM, so that time-dependent patterns such as common orders of fields can be represented. All this is done in a compact representation that can be learned from data. We refer to a recent dissertation [Murphy, 2002] for a good overview of all aspects of Dynamic Bayesian Networks.

Each document is considered to be a single stream of tokens. In our DBN, called the Bayesian Information Extraction Network (BIEN), the same structure is repeated for every index. Figure 1 presents the structure of BIEN. This structure contains state variables and feature variables. The most important state variable, for our purposes, is “Tag” which corresponds to information we are trying to extract. This variable classifies each token according to its target information field, or has the value “background” if the token does not belong to any field. “Last Target” is another hidden variable which reflects the order in which target information is found in the document. This variable is our way of implementing a memory in a “memory-less” Markov model. Its value is deterministically defined by the last non-background value of “Tag” variable. Another hidden variable, “Document Segment”, is introduced to account for differences in patterns between the header and the main body of the document. The former is close to the structured text format, while the latter to the free text. “Document Segment” influences “Tag” and together these two influence the set of observable variables which represent features of the text discussed in section 2. Standard inference algorithms for DBNs are similar to those for HMMs. In a DBN, some of the variables will typically be observed, while others will be hidden. The typical inference task is to determine the probability distribution over the states of a hidden variable over time, given time series data of the observed variables. This is usually accomplished using the forward-backward algorithm. Alternatively, we might want to know the most likely sequence of hidden variables. This is accomplished using the Viterbi algorithm. Learning the parameters of a DBN from data is accomplished using the EM algorithm (see e.g. Murphy [2002]). Note that in principle, parts of the system could be trained separately on independent corpus

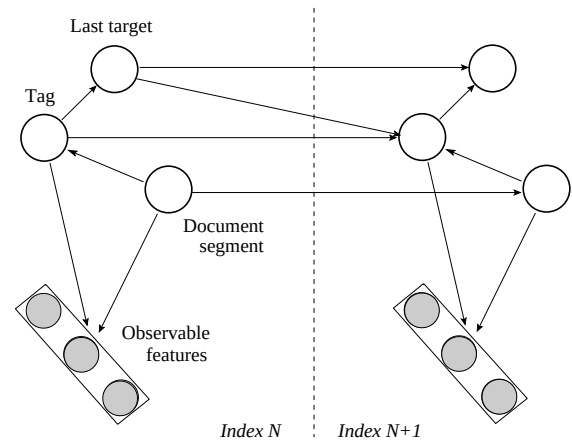


Figure 1: A schematic representation of BIEN.

to improve performance. For example, one could learn independently the conditional vocabulary of email/newsgroup headers, or learn a probability of part-of-speech conditioned on a word, to avoid dependence on external PoS taggers. Also prior knowledge about the domain and the language could be set in the system this way. The fact that *etime* almost never precedes *stime* as well as the fact that *speaker* is never a verb could be encoded in a conditional probability table (CPT). In large DBNs, exact inference algorithms are intractable, and so a variety of approximate methods have been developed. However, the number of hidden state variables in our model is small enough to allow exact algorithms to work. Indeed, all hidden nodes in our model are discrete variables which assume just a few values. “DocumentSegment” is binary in {Header, Body} range; “LastTarget” has as many values as “Tag”—four per number of target fields plus one for the background.

4 Results

Several researchers have reported results on the CMU seminar announcements corpus, which we have chosen in order to have a good basis of comparison. The CMU seminar announcements corpus consists of 485 documents. Each announcement contains some tags for target slots. On average *starting time* appears twice per document, while *location* and *speaker* 1.5 times, with up to 9 *speaker* slots and 4 *location* slots per document. Sometimes multiple instances of the same slot differ, e.g. speaker Dr. Steals also appears as Joe Steals². *Ending time*, *speaker* and *location* are missing from 48%, 16% and 5% of documents correspondingly. In order to demonstrate our method, we have developed a web site which works with arbitrary seminar announcement and reveals some semantic tagging. We also make available a list of errors in the original corpus, along with our new derivative seminar announcement corpus³.

²Obtaining 100% performance on the original corpus is impossible since some tags are misplaced and in general the corpus is not marked uniformly—sometimes secondary occurrences are ignored.

³The corpus and demo for this paper are available from <http://www.eecs.harvard.edu/~pesha/papers.html>

System	stime	etime	location	speaker
SNOW-IE	99.6	96.3	75.2	73.8
RAPIER	95.9	94.6	73.4	53.1
SRV	98.5	77.9	72.7	56.3
HMM	98.5	62.1	78.6	76.6
WHISK	92.6	86.1	66.6	18.3
(LP) ²	99.0	95.5	75.0	77.6
BIEN	96.0	98.8	87.1	76.9

Table 2: F1 performance measure for various IE systems.

The performance is calculated in the usual way, by *precision* $P = \frac{\text{correct answers}}{\text{answers produced}}$ and *recall* $R = \frac{\text{correct answers}}{\text{total correct}}$ combined into F measure geometrical average $F = \frac{2P \cdot R}{P + R}$. We report results using the same ten-fold cross validation test as other publications concerning this data set [Roth and Yih, 2001; Ciravegna, 2001]. The data is split randomly into training and testing set. The reported results are averaged over five runs. Table 2 presents a comparison with numerous previous attempts at the CMU seminar corpus. The figures are taken from Roth and Yih [2001]. BIEN performs comparably to the best system in each category, while notably outperforming other systems in finding *location*. This is partly due to the “LastTarget” variable. “LastTarget” variable turns out to be generally useful. Here is the learned conditional probability table (CPT) for $P(\text{Target}|\text{LastTarget})$, where the element (I, J) corresponds to the probability to get target tag J after target tag I was seen. We learn that initial tag is *stime* or *speaker* with 2:1 likelihood ratio; *etime* is naturally the most likely follower to *stime* and in turn forecasts *location*.

		Current Tag			
		stime	etime	location	speaker
Last Target	none	.66	0	.01	.33
	stime	.07	.46	.32	.14
	etime	.19	.002	.63	.18
	location	.37	.004	.02	.61
	speaker	.64	.002	.13	.22

Other variables turn out to be useless, e.g. the number of characters does not add anything to the performance, and neither does the initially introduced “SeenTag” variable which kept track of all tags seen up to the current position. Table 3 presents performance of BIEN with various individual features turned off. Note that figures for complete BIEN are

BIEN	stime	etime	location	speaker
no semantic	92.0	95.0	73.0	57.0
no memory	99.0	95.5	75.0	74.6
no lemma	54.5	53.4	29.0	18.4
no length	98.1	98.6	87.9	79.8
no case	98.1	98.5	78.3	71.8
complete	98.0	98.8	88.5	79.7

Table 3: F1 performance comparison across implementations of BIEN with disabled features.

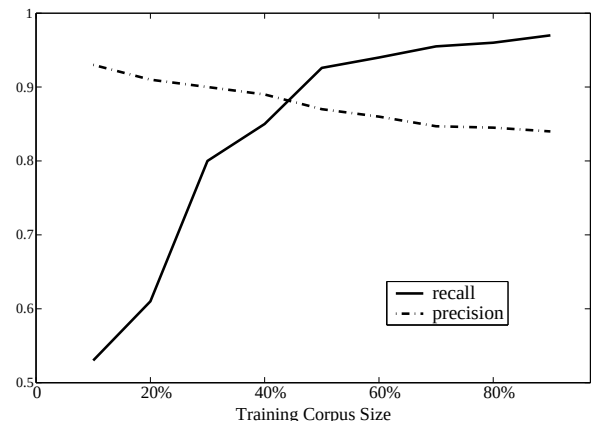


Figure 2: A learning curve for precision and recall with growing training sample size.

slightly better than in Table 2 since we pushed the fraction of the training data to the maximum. Capitalization helps identify *location* and *speaker*, while losing it does not damage performance drastically. Although information is reflected in syntactic and semantic features, most names in documents do not identify a speaker. One would hope to capture all relevant information by syntactic and semantic categories, however BIEN does not fare well without observing “Lemma”. Losing the semantic feature seriously undermines performance in *location* and *speaker* categories—ability to recognize names is rather valuable for many domains.

Reported figures are based on 80%-20% split of the corpus. Increasing the size of training corpus did not dramatically improve the performance in terms of F measure, as further illustrated by Figure 2, which presents a learning curve—precision and recall averaged over all fields, as a function of training data fraction. Trained on a small sample, BIEN acts very conservatively rarely picking fields, therefore scoring high precision and poor recall. Having seen hundreds of target field instances and tens of thousands of negative samples, BIEN learns to generalize, which leads to generous tagging i.e. lower precision and higher recall.

So far we provide results obtained on the original CMU seminar announcements data, which is not very challenging. Most documents contain the header section with all the target fields easily identifiable right after the corresponding key word. We have created a derivative dataset in which documents are stripped of headers and two extra fields are sought: *date* and *topic*. Indeed this corpus turned out to be more difficult, with our current set of features we obtain only 64% performance on *speaker* and 68% performance on *topic*. *Date* does not present a challenge except for cases of regular weekly events or relative dates like “tomorrow”. Admittedly, the bootstrapping test performance is not a guarantee of systems performance on novel data since preliminary processing, i.e. tokenization and gazetteering, as well as choice for PoS tag set, lead to a strong bias towards the training corpus.

5 Discussion

We have described how to integrate various aspects of language into a single probabilistic model, and to incrementally build a robust IE system based on a Bayesian network. Currently, we are working on learning the structure of BIEN automatically. It seems to subject itself nicely to structural EM [Friedman, 1998; Murphy, 2002]. The first step is automatic selection of relevant features. Another direction of current work is using approximate inference. We have tried LBP (Loopy-belief Propagation) [Murphy *et al.*, 1999; Murphy, 2002], but for the current structure of BIEN it seems to give no gain. More challenging applications which require larger, stronger connected networks, will benefit from approximate inference algorithms. It will enable quick on-line inference on the network learned off-line with exact methods, as well as learning for cases where exact inference is infeasible. One such network will result from integrating a PoS tagger and other feature extractors into BIEN. This is a natural extension of BIEN since various text processing routines are mutually dependent. Consider for example PoS tagging, sentence boundary detection and named entities recognition. Another complex BIEN structure will result if we try to better reflect complex relational information [Califf and Mooney, 1999; Roth and Yih, 2001; 2002] e.g. to process cases like seminar cancellations and rescheduling; and handle multi-slot extraction, e.g. multiple seminar announcements and conference schedules.

Acknowledgments

Kevin Murphy provided *BNT*, Kobi Gal helped to handle the corpus, anonymous reviewers gave helpful feedback.

References

- [Califf and Mooney, 1999] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conf. on Artificial Intelligence*, pages 328–334, 1999.
- [Charniak, 1999] Eugene Charniak. A maximum-entropy-inspired parser. Technical Report CS-99-12, Brown University, 1999.
- [Ciravegna, 2001] F. Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the Seventeenth International Joint Conf. on Artificial Intelligence*, 2001.
- [Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [Freitag and McCallum, 1999] Dayne Freitag and Andrew McCallum. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.
- [Freitag, 1998] Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.
- [Friedman, 1998] Nir Friedman. The bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conf. on Uncertainty in Artificial Intelligence*, 1998.
- [Manning and Schutze, 2001] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 2001.
- [Marcus *et al.*, 1994] M. Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.
- [McCallum *et al.*, 2000] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conf. on Machine Learning*, pages 591–598, Stanford, CA, 2000.
- [Mikheev *et al.*, 1998] Andrei Mikheev, Claire Grover, and Marc Moens. Description of the LTG system used for MUC-7. In *Seventh Message Understanding Conference (MUC-7)*, 1998.
- [Murphy *et al.*, 1999] Kevin Murphy, Yair Weiss, and Michael Jordan. Loopy-belief propagation for approximate inference. In *Proceedings of the Fifteenth Conf. on Uncertainty in Artificial Intelligence*, 1999.
- [Murphy, 2002] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.
- [Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.
- [Ratnaparkhi, 1999] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.
- [Ray and Craven, 2001] Souyma Ray and Mark Craven. Representing sentence structure in hidden Markov models for information extraction. In *Proceedings of the Seventeenth International Joint Conf. on Artificial Intelligence*, 2001.
- [Riloff, 1996] Ellen Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, 85(1-2):101–134, 1996.
- [Roth and Yih, 2001] Dan Roth and Wen-tau Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the Seventeenth International Joint Conf. on Artificial Intelligence*, 2001.
- [Roth and Yih, 2002] Dan Roth and Wen-tau Yih. Probabilistic reasoning for entity & relation recognition. In *The 20th International Conference on Computational Linguistics*, 2002.
- [Soderland, 1999] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

Hierarchical Hidden Markov Models for Information Extraction

Marios Skounakis^{*†}
marios@cs.wisc.edu

Mark Craven^{†*}
craven@biostat.wisc.edu

Soumya Ray^{*†}
sray@cs.wisc.edu

^{*}Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706

[†]Department of Biostatistics & Medical Informatics
University of Wisconsin
Madison, Wisconsin 53706

Abstract

Information extraction can be defined as the task of automatically extracting instances of specified classes or relations from text. We consider the case of using machine learning methods to induce models for extracting relation instances from biomedical articles. We propose and evaluate an approach that is based on using *hierarchical* hidden Markov models to represent the grammatical structure of the sentences being processed. Our approach first uses a shallow parser to construct a multi-level representation of each sentence being processed. Then we train hierarchical HMMs to capture the regularities of the parses for both positive and negative sentences. We evaluate our method by inducing models to extract binary relations in three biomedical domains. Our experiments indicate that our approach results in more accurate models than several baseline HMM approaches.

1 Introduction

In many application domains, there is the potential to greatly increase the utility of on-line text sources by using automated methods for mapping selected parts of the unstructured text into a structured representation. For example, the curators of genome databases would like to have tools that could accurately extract information from the scientific literature about entities such as genes, proteins, cells, diseases, etc. For this reason, there has been much recent interest in developing methods for the task of *information extraction* (IE), which can be defined as automatically recognizing and extracting instances of specific classes of entities and relationships among entities from text sources.

Machine learning methods often play a key role in IE systems because it is difficult and costly to manually encode the necessary extraction models. Hidden Markov models (HMMs) [Leek, 1997; Bikel *et al.*, 1999; Freitag and McCallum, 2000], and related probabilistic sequence models [McCallum *et al.*, 2000; Lafferty *et al.*, 2001], have been among the most accurate methods for learning information extractors. Most of the work in learning HMMs for information extraction has focused on tasks with semi-structured and other text sources in which English grammar does not play a key

“...
Here we report the identification of an integral membrane ubiquitin-conjugating enzyme. This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.
...”

↓
subcellular-localization(UBC6,endoplasmic reticulum)

Figure 1: An example of the information extraction task. The top of the figure shows part of a document from which we wish to extract instances of the *subcellular-localization* relation. The bottom of the figure shows the extracted tuple.

role. In contrast, the task we consider here is extracting information from abstracts of biological articles [Hirschman *et al.*, 2002]. In this domain, it is important that the learned models are able to represent regularities in the grammatical structure of sentences.

In this paper, we present an approach based on using *hierarchical* hidden Markov models (HHMMs) [Fine *et al.*, 1998] to extract information from the scientific literature. Hierarchical hidden Markov models have multiple “levels” of states which describe input sequences at different levels of granularity. In our models, the top level of the HMMs represent sentences at the level of phrases, and the lower level of the HMMs represent sentences at the level of individual words. Our approach involves computing a shallow parse of each sentence to be processed. During training and testing, the hierarchical HMMs manipulate a two-level description of the sentence parse, instead of just processing the sentence words directly. We evaluate our approach by extracting instances of three binary relations from abstracts of scientific articles. Our experiments show that our approach results in more accurate models than several baseline approaches using HMMs.

An example of a binary relation that we consider in our experiments is the *subcellular-localization* relation, which represents the location of a particular protein within a cell. We refer to the *domains* of this relation as *PROTEIN* and *LOCATION*. We refer to an instance of a relation as a *tuple*. Figure 1 provides an illustration of our extraction task. The top of the figure shows two sentences in an abstract, and the bottom of the figure shows the instance of the target relation

subcellular-localization that we would like to extract from the second sentence. This tuple asserts that the protein UBC6 is found in the subcellular compartment called the endoplasmic reticulum. In order to learn models to perform this task, we use training examples consisting of passages of text, annotated with the tuples that should be extracted from them.

In earlier work [Ray and Craven, 2001], we presented an approach that incorporates grammatical information into single-level HMMs. The approach described in this paper extends the earlier work by using hierarchical HMMs to provide a richer description of the information available from a sentence parse.

Hierarchical HMMs originally were developed by Fine *et al.* (1998), but the application of these models to information extraction is novel, and our approach incorporates several extensions to these models to tailor them to our task. Bikel *et al.* (1999) developed an approach to *named entity* recognition that uses HMMs with a multi-level representation similar to a hierarchical HMM. In their models, the top level represents the classes of interest (e.g. **person name**), and the bottom level represents the words in a sentence being processed. Our approach differs from theirs in several key respects: (i) our input representation for all sentences being processed is hierarchical, (ii) our models represent the shallow phrase structure of sentences, (iii) we focus on learning to extract *relations* rather than entities, (iv) we use *null* models to represent sentences that do not describe relations of interest, and (v) we use a discriminative training procedure. Miller *et al.* (2000) developed an information-extraction approach that uses a lexicalized, probabilistic context-free grammar (LPCFG) to simultaneously do syntactic parsing and semantic information extraction. The genre of text that we consider here, however, is quite different from the news story corpus on which available LPCFGs have been trained. Thus it is not clear how well this intriguing approach would transfer to our task.

2 Sentence Representation

In most previous work on HMMs for natural language tasks, the passages of text being processed have been represented as sequences of tokens. A hypothesis underlying our work is that incorporating sentence structure into the learned models will provide better extraction accuracy. Our approach is based on using syntactic parses of all sentences to be processed. In particular, we use the Sundance system [Riloff, 1998] to obtain a shallow parse of each given sentence.

The representation we use in this paper does not incorporate all of the information provided by the Sundance parser. Instead our representation provides a partially “flattened”, two-level description of each Sundance parse tree. The top level represents each sentence as a sequence of phrase segments. The lower level represents individual tokens, along with their part-of-speech (POS) tags. In positive training examples, if a segment contains a word or words that belong to a domain in a target tuple, the segment and the words of interest are annotated with the corresponding domain. We refer to these annotations as *labels*. Test instances do not contain labels – the labels are to be predicted by the learned IE model.

Figure 2 shows a sentence containing an instance of the

“This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.”			
1	NP_SEGMENT	DET	this
		UNK	enzyme
2	NP_SEGMENT:PROTEIN	UNK:PROTEIN	ubc6
3	VP_SEGMENT	V	localizes
4	PP_SEGMENT	PREP	to
5	NP_SEGMENT:LOCATION	ART	the
		N:LOCATION	endoplasmic
		N:LOCATION	reticulum
6	PP_SEGMENT	PREP	with
7	NP_SEGMENT	ART	the
		N	catalytic
		UNK	domain
8	VP_SEGMENT	V	facing
9	NP_SEGMENT	ART	the
		N	cytosol
	(a)	(b)	(c)

Figure 2: Input representation for a sentence which contains a subcellular-localization tuple: the sentence is segmented into typed phrases and each phrase is segmented into words typed with part-of-speech tags. Phrase types and labels are shown in column (a). Word part-of-speech tags and labels are shown in column (b). The words of the sentence are shown in column (c). Note the grouping of words in phrases. The labels (PROTEIN, LOCATION) are present only in the training sentences.

subcellular-localization relation and its annotated segments. The sentence is segmented into typed phrases and each phrase is segmented into words typed with part-of-speech tags. For example, the second phrase segment is a noun phrase (NP_SEGMENT) that contains the protein name UBC6 (hence the PROTEIN label). Note that the types are constants that are pre-defined by our representation of Sundance parses, whereas the labels are defined by the domains of the particular relation we are trying to extract.

3 Hierarchical HMMs for Information Extraction

A schematic of one of our hierarchical HMMs is shown in Figure 3. The top of the figure shows the *positive model*, which is trained to represent sentences that contain instances of the target relation. The bottom of the figure shows the *null model*, which is trained to represent sentences that do not contain relation instances (e.g. off-topic sentences). At the “coarse” level, our hierarchical HMMs represent sentences as sequences of phrases. Thus, we can think of the top level as an HMM whose states emit phrases. We refer to this HMM as the *phrase HMM*, and its states *phrase states*. At the “fine” level, each phrase is represented as a sequence of words. This is achieved by embedding an HMM within each phrase state. We refer to these embedded HMMs as *word HMMs* and their states as *word states*. The phrase states in Figure 3 are depicted with rounded rectangles and word states are depicted with ovals. To explain a sentence, the HMM would first follow a transition from the START state to some phrase state q_i , use the word HMM of q_i to emit the first phrase of the sentence, then transition to another phrase state q_j , emit another

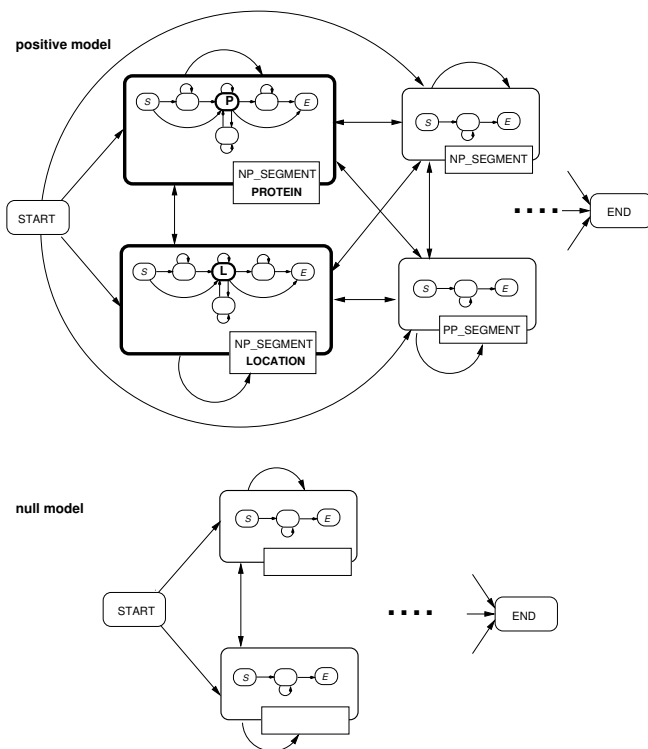


Figure 3: Schematic of the architecture of a hierarchical HMM for the subcellular-localization relation. The top part of the figure shows the positive model and the bottom part the null model. Phrase states are depicted as rounded rectangles and word states as ovals. The types and labels of the phrase states are shown within rectangles at the bottom right of each state. Labels are shown in bold and states associated with non-empty label sets are depicted with bold borders. The labels of word states are abbreviated for compactness.

phrase using the word HMM of q_j and so on until it moves to the END state of the phrase HMM. Note that only the word states have direct emissions.

Like the phrases in our input representation, each phrase state in the HMM has a type and may have one or more labels. Each phrase state is constrained to emit only phrases whose type agrees with the state's type. We refer to states that have labels associated with them as *extraction states*, since they are used to predict which test sentences should have tuples extracted from them.

The architectures of the word HMMs are shown in Figure 4. We use three different architectures depending on the labels associated with the phrase state in which the word HMM is embedded. The word HMMs for the phrase states with empty label sets (Figure 4(a)) consist of a single emitting state with a self-transition. For the extraction states of the phrase HMM, the word HMMs have a specialized architecture with different states for the domain instances, and for the words that come before, between and after the domain instances (Figures 4(b) and 4(c)). All the states of the word HMMs can emit words of any type (part-of-speech). That is, they are *untyped*, in contrast to the *typed* phrase states. The word states are annotated with label sets, and are trained to emit words with identical label sets. For example, the word

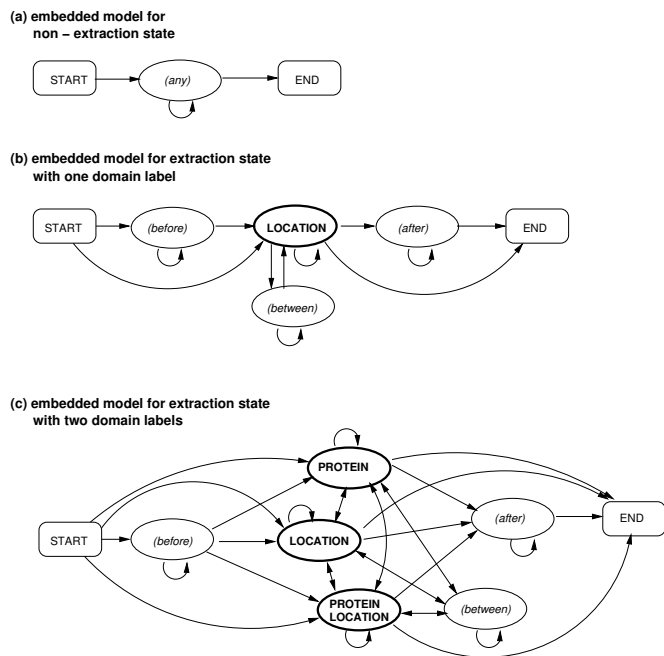


Figure 4: Architectures of the word HMMs for the subcellular-localization relation. Bold text within states denotes domain labels. For states with implicit empty labels, italicized text within parentheses denotes the position of the state's emissions relative to the domain words. The figure shows (a) the structure of the embedded HMMs for phrase states without labels, (b), phrase states with one label and (c) phrase states with two labels.

HMM shown in Figure 4(b) can explain the phrase “the endoplasmic reticulum” by following a transition from the START state to the *(before)* state, emitting the word “the”, transitioning to the LOCATION state, emitting the words “endoplasmic” and “reticulum” with the LOCATION label and then transitioning to the END state. In order for a phrase state to emit a whole phrase, as given by the input representation, and not sequences of words that are shorter or longer than a phrase, we require that the embedded word HMM transition to the end state exactly when it has emitted all the words of a given phrase. Thus word HMMs will always emit sequences of words that constitute whole phrases and transitions between phrase states occur only at phrase boundaries.

The standard dynamic programming algorithms that are used for learning and inference in HMMs – Forward, Backward and Viterbi [Rabiner, 1989] – need to be slightly modified for our hierarchical HMMs. In particular, they need to (i) handle the multiple-levels of the input representation, enforcing the constraint that word HMMs must emit sequences of words that constitute phrases, and (ii) support the use of typed phrase states by enforcing agreement between state and phrase types.

The Forward algorithm for our hierarchical HMMs is defined by the recurrence relationships shown in Table 1. The first three equations of the recurrence relation provide a phrase-level description of the algorithm, and the last three equations provide a word-level description. Notice that the third equation describes the linkage between the phrase level

$\alpha_0(0) = 1$ $\alpha_a(0) = 0, \quad q_a \neq q_0$ $\alpha_a(i) = \begin{cases} \alpha_{a,n}(i, s_i), & \text{if } \text{type}(q_a) = \text{type}(s_i) \\ 0, & \text{otherwise.} \end{cases}$ $\alpha_{a,0}(i, 0) = \sum_b T(q_a q_b)\alpha_b(i-1)$ $\alpha_{a,b}(i, 0) = 0, \quad q_{a,b} \neq q_{a,0}$ $\alpha_{a,b}(i, j) = E(s_{i,j} q_{a,b}) \sum_c T(q_{a,b} q_{a,c})\alpha_{a,c}(i, j-1)$	s_i i -th phrase in sentence s $s_{i,j}$ j -th word in phrase s_i q_a, q_0, q_n a -th, START and END states of the phrase HMM $q_{a,b}, q_{a,0}, q_{a,n}$ b -th, START and END states of the word HMM in phrase state q_a $\alpha_a(i)$ probability of emitting sequence of phrases $s_1 \dots s_i$, starting from the start state and ending at state q_a $\alpha_{a,b}(i, j)$ probability of emitting sequence of words $s_{1,1} \dots s_{i,j}$, starting from the START state and ending at state $q_{a,b}$ $E(s_{i,j} q_{a,b})$ probability that word state $q_{a,b}$ emits word $s_{i,j}$ $T(q_a q_b)$ probability of transition from phrase state q_b to phrase state q_a $T(q_{a,b} q_{a,c})$ probability of transition from word state $q_{a,c}$ to word state $q_{a,b}$
--	--

Table 1: The left side of the table shows the Forward-algorithm recurrence relation for our hierarchical HMMs. The right side of the table defines the notation used in the recurrence relation.

and the word level. The Backward and Viterbi algorithms require similar modifications, but we do not show them due to space limitations.

As illustrated in Figure 2, each training instance for our HMMs consists of a sequence of words, segmented into phrases, and an associated sequence of labels. For a test instance, we would like our trained model to accurately predict a sequence of labels given only the observable part of the sentence (i.e. the words and phrases). We use a discriminative training algorithm [Krogh, 1994] that tries to find model parameters, $\hat{\theta}$, to maximize the conditional likelihood of the labels given the observable part of the sentences:

$$\hat{\theta} = \arg \max_{\theta} \prod_l \Pr(c^l | s^l, \theta). \quad (1)$$

Here s^l is the sequence of words/phrases for the l th instance, and c^l is the sequence of labels for the instance. This training algorithm will converge to a local maximum of the objective function. We initialize the parameters of our models by first doing standard generative training. We then apply Krogh’s algorithm which involves iterative updates to the HMM parameters. To avoid overfitting, we stop training when the accuracy on a held-aside tuning set is maximized.

In order for this algorithm to be able to adjust the parameters of the positive model in response to negative instances and vice-versa, we join our positive and null models as shown in Figure 5. This combined model includes the positive and

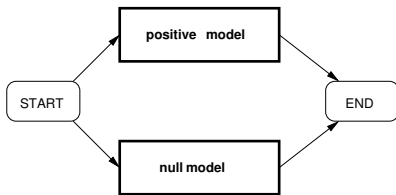


Figure 5: Architecture of the *combined model*. The positive and null models refer to the models in Figure 3.

the null models (shown in Figure 3) as its two submodels, with shared START and END states.

Once a model has been trained, we can use the Viterbi algorithm to predict tuples in test sentences. We extract a tuple from a given sentence if the Viterbi path goes through states with labels for *all* the domains of the relation. For example, for the subcellular-localization relation, the Viterbi path for a sentence must pass through a state with the PROTEIN label and a state with the LOCATION label. This process is illustrated in Figure 6.

4 Hierarchical HMMs with Context Features

In this section we describe an extension to the hierarchical HMMs presented in the previous section that enables them to represent additional information about the structure of sentences within phrases. We refer to these extended HMMs as *Context hierarchical HMMs* (CHHMMs). Whereas the hierarchical HMMs presented earlier partition a sentence s into disjoint observations $s_{i,j}$ where each $s_{i,j}$ is a word, a CHHMM represents s as a sequence of *overlapping* observations $o_{i,j}$. Each observation $o_{i,j}$ consists of a window of three words, centered around $s_{i,j}$, together with the part-of-speech tags of these words. Formally, $o_{i,j}$ is a vector $\langle s_{i,(j-1)}, s_{i,j}, s_{i,(j+1)}, t_{i,(j-1)}, t_{i,j}, t_{i,(j+1)} \rangle$ where $t_{i,j}$ is the part-of-speech tag of word $s_{i,j}$. Note that $o_{i,j}$ and $o_{i,(j+1)}$ share $s_{i,j}, s_{i,(j+1)}, t_{i,j}$ and $t_{i,(j+1)}$, although these features are located in different positions in the two vectors. Figure 7 shows the vectors emitted for the phrase “the endoplasmic reticulum” by a word HMM in the CHHMM.

Using features that represent the previous and next words allows the models to capture regularities about pairs or triplets of words. For instance, a CHHMM is potentially able to learn that the word “membrane” is part of a subcellular location when found in “plasma membrane” while it is not when found in “a membrane”. Furthermore, by using features that represent the part-of-speech of words, the models are able to learn regularities about groups of words with the same part of speech in addition to regularities about individual words.

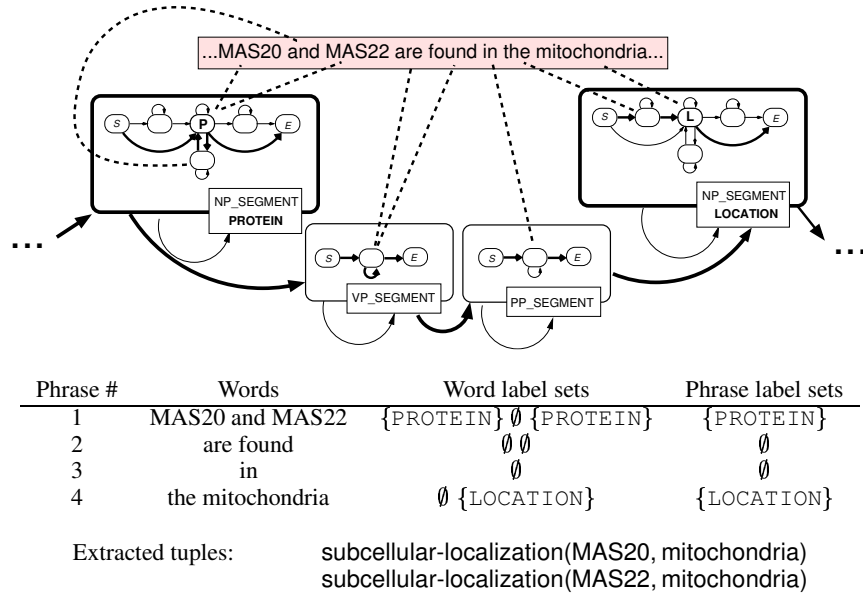


Figure 6: Example of the procedure for extracting tuples of the subcellular-localization relation from the sentence fragment “...MAS20 and MAS22 are found in the mitochondria...”. The top of the figure shows how the most likely path explains the sentence fragment. Bold transitions between states denote the most likely path. Dashed lines connect each state with the words that it emits. The table shows the label sets that are assigned to the phrases and the words of the sentence. The extracted tuples are shown at the bottom of the figure.

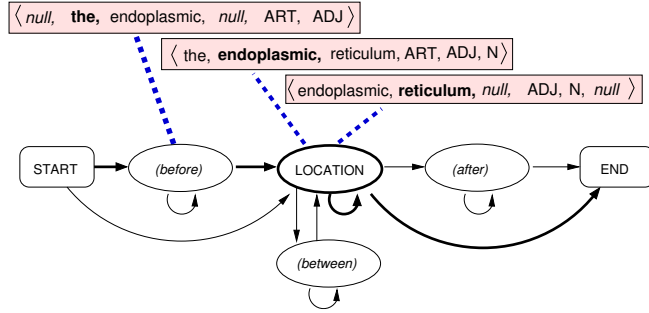


Figure 7: Generation of the phrase “the endoplasmic reticulum” by a word HMM in a CHHMM. The bold arcs represent the path that generates the phrase. The vector observations $o_{i,j}$ emitted by each state are shown in the rectangles above the model and are connected with dotted arcs with the emitting state. The word that would be emitted by each state of the equivalent HHMM is shown in boldface.

The advantages of this representation are especially realized when dealing with an out-of-vocabulary word; in this case part-of-speech tags and neighboring words may be quite informative about the meaning and use of the out-of-vocabulary word. For example, an out-of-vocabulary adjective will rarely be a protein, since proteins are usually nouns.

Because the number of possible observations for a given word state in a CHHMM is very large (all possible vectors representing sequences of three words and their POS tags), to model the probability of an observation $o_{i,j}$, our CHHMMs assume that the features are conditionally independent given the state. Under this assumption, the probability of the obser-

vation $o_{i,j}$ being emitted by state $q_{a,b}$ is then defined as

$$E(o_{i,j}|q_{a,b}) = \prod_{k=1..6} E_k(o_{i,j,k}|q_{a,b}) \quad (2)$$

where $E_k(o_{i,j,k}|q_{a,b})$ is the probability of word state $q_{a,b}$ emitting an observation whose k -th feature is $o_{i,j,k}$.

Note that the features employed by the representation of Equation 2 are clearly not conditionally independent. Consecutive words are not independent of one another and certainly the part-of-speech tag of a word is not independent of the word itself. However, we argue that the discriminative training algorithm we use [Krogh, 1994] can compensate in part for this violation of the independence assumption.

5 Empirical Evaluation

In this section we present experiments testing the hypothesis that our hierarchical HMMs are able to provide more accurate models than HMMs that incorporate less grammatical information. In particular we empirically compare two types of hierarchical HMMs with three baseline HMMs.

- **Context HHMMs:** hierarchical HMMs with context features, as described in the previous section.
- **HHMMs:** hierarchical HMMs without context features.
- **Phrase HMMs:** single-level HMMs in which states are typed (as in the phrase level of an HHMM) and emit whole phrases. These HMMs were introduced by Ray and Craven (2001). Unlike hierarchical HMMs, the states of Phrase HMMs do not have embedded HMMs which emit words. Instead each state has a single multinomial distribution to represent its emissions, and each emitted phrase is treated as a bag of words.

- **POS HMMs:** single-level HMMs in which states emit words, but are typed with part-of-speech tags so that a given state can emit words with only a single POS.
- **Token HMMs:** single-level HMMs in which untyped states emit words.

We evaluate our hypotheses on three data sets that we have assembled from the biomedical literature.¹ The data sets are composed of abstracts gathered from the MEDLINE database [National Library of Medicine, 2003]. The first set contains instances of the subcellular-localization relation. It is composed of 769 positive and 6,360 negative sentences. The positive sentences contain 949 total tuple instances. The number of actual tuples is 404 since some tuples occur multiple times either in the same sentence or in multiple sentences. The second, which we refer to as the disorder-association data set, characterizes a binary relation between genes and disorders. It contains 829 positive and 11,771 negative sentences. The positive sentences represent 878 instances of 145 tuples. The third, which we refer to as the protein-interaction data set, characterizes physical interactions between pairs of proteins. It is composed of 5,457 positive and 42,015 negative sentences. It contains 8,088 instances of 819 tuples.

We use five-fold cross-validation to measure the accuracy of each approach. Before processing all sentences, we obtain parses from Sundance, and then stem words with Porter's stemmer [Porter, 1980]. We map all numbers to a special NUMBER token and all words that occur only once in a training set to an OUT-OF-VOCAB token. Also, we discard all punctuation. The same preprocessing is done on test sentences, with the exception that words that were not encountered in the training set are mapped to the OUT-OF-VOCAB token. The vocabulary is the same for all emitting states in the models, and all parameters are smoothed using *m-estimates* [Cestnik, 1990]. We train all models using the discriminative training procedure referred to in Section 3 [Krogh, 1994].

To evaluate our models we construct *precision-recall* graphs. *Precision* is defined as the fraction of correct tuple instances among those instances that are extracted by the model. *Recall* is defined as the fraction of correct tuple instances extracted by the model over the total number of tuple instances that exist in the data set. For each tuple extracted from sentence s , we calculate a confidence measure as:

$$c(s) = \frac{\delta_n(|s|)}{\alpha_n(|s|)}.$$

Here q_n refers to the END state of the combined model, $\delta_n(|s|)$ is the probability of the most likely path, given by the Viterbi algorithm, and $\alpha_n(|s|)$ is the total probability of the sequence, calculated with the Forward algorithm. We construct precision-recall curves by varying a threshold on these confidences.

Figures 8, 9 and 10 show the precision-recall curves for the three data sets. Each figure shows curves for the five types of

¹Earlier versions of two of these data sets were used in our previous work [Ray and Craven, 2001]. Various aspects of the data sets have been cleaned up, however, and thus the versions used here are somewhat different. All three data sets are available from <http://www.biostat.wisc.edu/~craven/ie/>.

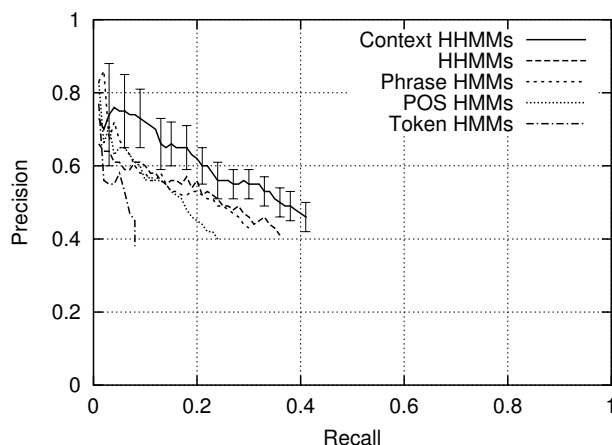


Figure 8: Precision vs. recall for the five types of HMMs on the subcellular-localization data set.

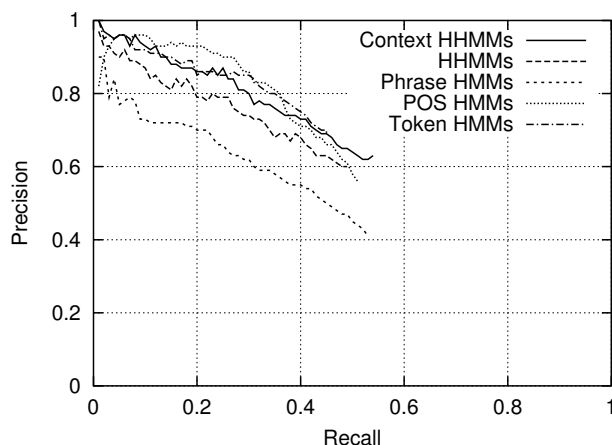


Figure 9: Precision vs. recall for the five types of HMMs on the disorder-association data set.

HMMs described at the beginning of this section. We show error bars for the Context HHMM precision values for the subcellular-localization and protein-interaction data sets. For these two data sets, the hierarchical HMM models clearly have superior precision-recall curves to the baseline models. At nearly every level of recall, the hierarchical HMMs exhibit higher precision than the baselines. Additionally, the HHMMs achieve higher endpoint recall values. The results are not as definitive for the disorder-association data set. Here, the POS HMMs and the Token HMMs achieve precision levels that are comparable to, and in some cases slightly better than, the Context HHMMs. There is not a clear winner for this data set, but the Context HHMMs are competitive.

Comparing the Context HHMMs to the ordinary HHMMs, we see that the former results in superior precision-recall curves for all three data sets. This result demonstrates that clearly there is value in including the context features in hierarchical HMMs for this type of task. In summary, our empirical results support the hypothesis that the ability our hierarchical HMM approach to capture grammatical information about sentences results in more accurate learned models.

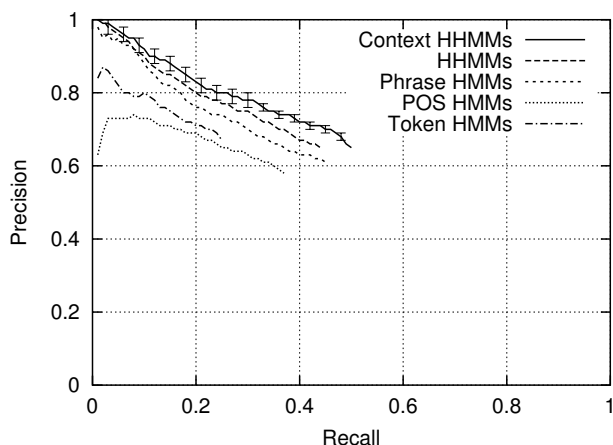


Figure 10: Precision vs. recall for the five types of HMMs on the protein-interaction data set.

6 Conclusion

We have presented an approach to learning models for information extraction that is based on using hierarchical HMMs to represent the grammatical structure of the sentences being processed. We employ a shallow parser to obtain parse trees for sentences and then use these trees to construct the input representation for the hierarchical HMMs.

Our approach builds on previous work on hierarchical HMMs and incorporating grammatical knowledge into information-extraction models. The application of HHMMs to IE is novel and has required us to modify HHMM learning algorithms to operate on a hierarchical input representation. In particular our methods take into account that phrases and states must have matching types, and that phrase states must emit complete phrases. We have also introduced a novel modification of HHMMs in which observations can be feature vectors. With respect to previous work on incorporating grammatical knowledge into IE models, our main contribution is an approach that takes advantage of grammatical information represented at multiple scales. An appealing property of our approach is that it generalizes to additional levels of description of the input text.

We have evaluated our approach in the context of learning IE models to extract instances of three biomedical relations from the abstracts of scientific articles. These experiments demonstrate that incorporating a hierarchical representation of grammatical structure improves extraction accuracy in hidden Markov models.

Acknowledgments

This research was supported in part by NIH grant 1R01 LM07050-01, NSF grant IIS-0093016, and a grant to the University of Wisconsin Medical School under the Howard Hughes Medical Institute Research Resources Program for Medical Schools.

References

[Bikel *et al.*, 1999] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a

name. *Machine Learning*, 34(1):211–231, 1999.

[Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proc. of the 9th European Conf. on Artificial Intelligence*, pages 147–150, Stockholm, Sweden, 1990. Pitman.

[Fine *et al.*, 1998] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.

[Freitag and McCallum, 2000] D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proc. of the 17th National Conf. on Artificial Intelligence*, 2000. AAAI Press.

[Hirschman *et al.*, 2002] L. Hirschman, J. Park, J. Tsujii, L. Wong, and C. Wu. Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, 18:1553–1561, 2002.

[Krogh, 1994] A. Krogh. Hidden Markov models for labeled sequences. In *Proc. of the 12th International Conf. on Pattern Recognition*, pages 140–144, Jerusalem, Israel, 1994. IEEE Computer Society Press.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the 18th International Conf. on Machine Learning*, pages 282–289, Williamstown, MA, 2001. Morgan Kaufmann.

[Leek, 1997] T. Leek. Information extraction using hidden Markov models. M.S. thesis, Dept. of Computer Science and Engineering, Univ. of California, San Diego, 1997.

[McCallum *et al.*, 2000] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of the 17th International Conf. on Machine Learning*, pages 591–598, Stanford, CA, 2000. Morgan Kaufmann.

[Miller *et al.*, 2000] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. A novel use of statistical parsing to extract information from text. In *Proc. of the 6th Applied Natural Language Processing Conf.*, pages 226–233, Seattle, WA, 2000. Association for Computational Linguistics.

[National Library of Medicine, 2003] National Library of Medicine. The MEDLINE database, 2003. <http://www.ncbi.nlm.nih.gov/PubMed/>.

[Porter, 1980] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):127–130, 1980.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[Ray and Craven, 2001] S. Ray and M. Craven. Representing sentence structure in hidden Markov models for information extraction. In *Proc. of the 17th International Joint Conf. on Artificial Intelligence*, pages 1273–1279, Seattle, WA, 2001. Morgan Kaufmann.

[Riloff, 1998] E. Riloff. The sundance sentence analyzer, 1998. <http://www.cs.utah.edu/projects/nlp/>.

Coherent Keyphrase Extraction via Web Mining

Peter D. Turney

Institute for Information Technology, National Research Council of Canada
M-50 Montreal Road, Ottawa, Ontario, Canada, K1A 0R6
peter.turney@nrc-cnrc.gc.ca

Abstract

Keyphrases are useful for a variety of purposes, including summarizing, indexing, labeling, categorizing, clustering, highlighting, browsing, and searching. The task of automatic keyphrase extraction is to select keyphrases from within the text of a given document. Automatic keyphrase extraction makes it feasible to generate keyphrases for the huge number of documents that do not have manually assigned keyphrases. A limitation of previous keyphrase extraction algorithms is that the selected keyphrases are occasionally incoherent. That is, the majority of the output keyphrases may fit together well, but there may be a minority that appear to be outliers, with no clear semantic relation to the majority or to each other. This paper presents enhancements to the Kea keyphrase extraction algorithm that are designed to increase the coherence of the extracted keyphrases. The approach is to use the degree of statistical association among candidate keyphrases as evidence that they may be semantically related. The statistical association is measured using web mining. Experiments demonstrate that the enhancements improve the quality of the extracted keyphrases. Furthermore, the enhancements are not domain-specific: the algorithm generalizes well when it is trained on one domain (computer science documents) and tested on another (physics documents).

1 Introduction

A journal article is often accompanied by a list of keyphrases, composed of about five to fifteen important words and phrases that express the primary topics and themes of the paper. For an individual document, keyphrases can serve as a highly condensed summary, they can supplement or replace the title as a label for the document, or they can be highlighted within the body of the text, to facilitate speed reading (skimming). For a collection of documents, keyphrases can be used for indexing, categorizing (classifying), clustering, browsing, or searching. Keyphrases are most familiar in the context of journal articles, but many other types of documents could benefit from the use of keyphrases, including web pages, email messages, news reports, magazine articles, and business papers.

The vast majority of documents currently do not have keyphrases. Although the potential benefit is large, it would not be practical to manually assign keyphrases to them. This

is the motivation for developing algorithms that can automatically supply keyphrases for a document. Section 2.1 discusses past work on this task.

This paper focuses on one approach to supplying keyphrases, called *keyphrase extraction*. In this approach, a document is decomposed into a set of phrases, each of which is considered as a possible candidate keyphrase. A supervised learning algorithm is taught to classify candidate phrases as *keyphrases* and *non-keyphrases*. The induced classification model is then used to extract keyphrases from any given document [Turney, 1999, 2000; Frank *et al.*, 1999; Witten *et al.*, 1999, 2000].

A limitation of prior keyphrase extraction algorithms is that the output keyphrases are at times incoherent. For example, if ten keyphrases are selected for a given document, eight of them might fit well together, but the remaining two might be outliers, with no apparent semantic connection to the other eight or to each other. Informal analysis of many machine-extracted keyphrases suggests that these outliers almost never correspond to author-assigned keyphrases. Thus discarding the incoherent candidates might improve the quality of the machine-extracted keyphrases.

Section 2.2 examines past work on measuring the coherence of text. The approach used here is to measure the degree of statistical association among the candidate phrases [Church and Hanks, 1989; Church *et al.*, 1991]. The hypothesis is that semantically related phrases will tend to be statistically associated with each other, and that avoiding unrelated phrases will tend to improve the quality of the output keyphrases.

Section 3 describes the Kea keyphrase extraction algorithm [Frank *et al.*, 1999; Witten *et al.*, 1999, 2000]. Each candidate phrase is represented in Kea as a feature vector for classification by a supervised learning algorithm. Four different sets of features are evaluated in this paper. Two of the sets have been used in the past [Frank *et al.*, 1999] and the other two are introduced here, to address the problem of incoherence. The new features are based on the use of web mining to measure the statistical association among the candidate phrases [Turney 2001].

In Section 4, experiments show that the new web mining features significantly improve the quality of the extracted keyphrases, using the author's keyphrases as a benchmark. In the first experiment (Section 4.1), the algorithm is trained and tested on the same domain (computer science

documents). In the second experiment (Section 4.2), the algorithm is trained on one domain (computer science) and tested on another (physics). In both cases, the new features result in a significant improvement in performance. In contrast, one of the old feature sets works well in the first case (intradomain) but poorly in the second case (interdomain).

Section 5 considers limitations and future work. Section 6 presents the conclusions.

2 Related Work

Section 2.1 discusses related work on generating keyphrases and Section 2.2 considers related work on measuring coherence.

2.1 Assignment versus Extraction

There are two general approaches to automatically supplying keyphrases for a document: *keyphrase assignment* and *keyphrase extraction*. Both approaches use supervised machine learning from examples. In both cases, the training examples are documents with manually supplied keyphrases.

In keyphrase assignment, there is a predefined list of keyphrases (in the terminology of library science, a *controlled vocabulary* or *controlled index terms*). These keyphrases are treated as classes, and techniques from text classification (text categorization) are used to learn models for assigning a class to a given document [Leung and Kan, 1997; Dumais *et al.*, 1998]. Usually the learned models will map an input document to several different controlled vocabulary keyphrases.

In keyphrase extraction, keyphrases are selected from within the body of the input document, without a predefined list. When authors assign keyphrases without a controlled vocabulary (in library science, *free text keywords* or *free index terms*), typically from 70% to 90% of their keyphrases appear somewhere in the body of their documents [Turney, 1999]. This suggests the possibility of using author-assigned free text keyphrases to train a keyphrase extraction system. In this approach, a document is treated as a set of candidate phrases and the task is to classify each candidate phrase as either a *keyphrase* or *non-keyphrase* [Turney, 1999, 2000; Frank *et al.*, 1999; Witten *et al.*, 1999, 2000].

Keyphrase extraction systems are trained using a corpus of documents with corresponding free text keyphrases. The GenEx keyphrase extraction system consists of a set of parameterized heuristic rules that are tuned to the training corpus by a genetic algorithm [Turney, 1999, 2000]. The Kea keyphrase extraction system uses a naïve Bayes learning method to induce a probabilistic model from the training corpus [Frank *et al.*, 1999; Witten *et al.*, 1999, 2000]. After training, both systems allow the user to specify the desired number of keyphrases to be extracted from a given input document.

In experimental evaluations, using independent testing corpora, GenEx and Kea achieve roughly the same level of performance, measured by the average number of matches between the author-assigned keyphrases and the machine-extracted keyphrases [Frank *et al.*, 1999; Witten *et al.*, 1999,

2000]. Depending on the corpus, if the user requests five keyphrases to be output, an average of 0.9 to 1.5 of the output phrases will match with the author-assigned keyphrases [Frank *et al.*, 1999]. If the user requests fifteen keyphrases, an average of 1.8 to 2.8 will match with the author's keyphrases.

These performance numbers are misleadingly low, because the author-assigned keyphrases are usually only a small subset of the set of good quality keyphrases for a given document. A more accurate picture is obtained by asking human readers to rate the quality of the machine's output. In a sample of 205 human readers rating keyphrases for 267 web pages, 62% of the 1,869 phrases output by GenEx were rated as "good", 18% were rated as "bad", and 20% were left unrated [Turney, 2000]. This suggests that about 80% of the phrases are acceptable (not "bad") to human readers, which is sufficient for many applications.

Asking human readers to rate machine-extracted keyphrases is a costly process. It is much more economical to use author-assigned keyphrases as a benchmark for evaluating the machine-extracted keyphrases. Jones and Paynter [2001, 2002] argue that it is reasonable to use comparison with authors as a surrogate for rating by human readers. They show (1) the vast majority of Kea keyphrases are rated positively by human readers, (2) the performance of Kea is comparable to GenEx, according to human ratings, (3) readers generally rate author-assigned keyphrases as good, and (4) different human readers tend to agree on the rating of keyphrases.

Furthermore, Gutwin *et al.* [1999] show that machine-extracted keyphrases are useful in an application. The Keyphind system uses keyphrases extracted by Kea in a new kind of search engine. For certain kinds of browsing tasks, Keyphind was judged to be superior to a traditional search engine [Gutwin *et al.*, 1999]. In summary, there is solid evidence that author-assigned keyphrases are a good benchmark for training and testing keyphrase extraction algorithms, but it should be noted that the performance numbers underestimate the actual quality of the output keyphrases, as judged by human readers.

2.2 Coherence

An early study of coherence in text was the work of Halliday and Hasan [1976]. They argued that coherence is created by several devices: the use of semantically related terms, coreference, ellipsis, and conjunctions. The first device, semantic relatedness, is particularly useful for isolated words and phrases, outside of the context of sentences and paragraphs. Halliday and Hasan [1976] called this device *lexical cohesion*. Morris and Hirst [1991] computed lexical cohesion by using a thesaurus to measure the relatedness of words. Recent work on text summarization has used lexical cohesion in an effort to improve the coherence of machine-generated summaries. Barzilay and Elhadad [1997] used the WordNet thesaurus to measure lexical cohesion in their approach to summarization.

Keyphrases are often specialized technical phrases of two or three words that do not appear in a thesaurus such as

WordNet. In this paper, instead of using a thesaurus, statistical word association is used to estimate lexical cohesion. The idea is that phrases that often occur together tend to be semantically related.

There are many statistical measures of word association [Manning and Schütze, 1999]. The measure used here is Pointwise Mutual Information (PMI) [Church and Hanks, 1989; Church *et al.*, 1991]. PMI can be used in conjunction with a web search engine, which enables it to effectively exploit a corpus of about one hundred billion words [Turney, 2001]. Experiments with synonym questions, taken from the Test of English as a Foreign Language (TOEFL), show that word association, measured with PMI and a web search engine, corresponds well to human judgements of synonymy relations between words [Turney, 2001].

3 Kea: Four Feature Sets

Kea generates candidate phrases by looking through the input document for any sequence of one, two, or three consecutive words. The consecutive words must not be separated by punctuation and must not begin or end with stop words (such as “the”, “of”, “to”, “and”, “he”, etc.). Candidate phrases are normalized by converting them to lower case and stemming them. Kea then uses the naïve Bayes algorithm [Domingos and Pazzani, 1997] to learn to classify the candidate phrases as either *keyphrase* or *non-keyphrase*.¹

In the simplest version of Kea, candidate phrases are classified using only two features: $TF \times IDF$ and *distance* [Frank *et al.*, 1999; Witten *et al.*, 1999, 2000]. In the following, this is called the *baseline feature set* (Section 3.1). Another version of Kea adds a third feature, keyphrase frequency, yielding the *keyphrase frequency feature set* (Section 3.2) [Frank *et al.*, 1999]. This paper introduces a new set of features for measuring coherence. When combined with the baseline features, the result is the *coherence feature set* (Section 3.3). When combined with the keyphrase frequency feature set, the result is the *merged feature set* (Section 3.4).

After training, given a new input document and a desired number of output phrases, N , Kea converts the input document into a set of candidate phrases with associated feature vectors. It uses the naïve Bayes model to calculate the probability that the candidates belong to the class *keyphrase*, and then it outputs the N candidates with the highest probabilities.

3.1 Baseline Feature Set

$TF \times IDF$ (Term Frequency times Inverse Document Frequency) is commonly used in information retrieval to assign weights to terms in a document [van Rijsbergen, 1979]. This numerical feature assigns a high value to a phrase that is relatively frequent in the input document (the TF component), yet relatively rare in other documents (the

IDF component). There are many ways to calculate $TF \times IDF$; see Frank *et al.* [1999] for a description of their method. The *distance* feature is, for a given phrase in a given document, the number of words that precede the first occurrence of the phrase, divided by the number of words in the document. The baseline feature set consists of these two features.

The $TF \times IDF$ and *distance* features are real-valued. Kea uses Fayyad and Irani’s [1993] algorithm to discretize the features. This algorithm uses a Minimum Description Length (MDL) technique to partition the features into intervals, such that the entropy of the class is minimized with respect to the intervals and the information required to specify the intervals.

The naïve Bayes algorithm applies Bayes’ formula to calculate the probability of membership in a class, using the (“naïve”) assumption that the features are statistically independent. Suppose that a candidate phrase has the feature vector $\langle T, D \rangle$, where T is an interval of the discretized $TF \times IDF$ feature and D is an interval of the discretized *distance* feature. Using Bayes’ formula and the independence assumption, we can calculate the probability that the candidate phrase is a keyphrase, $p(\text{key} | T, D)$, as follows [Frank *et al.*, 1999]:

$$p(\text{key} | T, D) = \frac{p(T | \text{key}) \cdot p(D | \text{key}) \cdot p(\text{key})}{p(T, D)} \quad (1)$$

The probabilities in this formula are readily estimated by counting frequencies in the training corpus.

3.2 Keyphrase Frequency Feature Set

The keyphrase frequency feature set consists of $TF \times IDF$ and *distance* plus a third feature, *keyphrase-frequency* [Frank *et al.*, 1999]. For a phrase P in a document D with a training corpus C , *keyphrase-frequency* is the number of times P occurs as an author-assigned keyphrase for all documents in C except D . Like the other features, *keyphrase-frequency* is discretized [Fayyad and Irani, 1993]. Equation (1) easily expands to include *keyphrase-frequency* [Frank *et al.*, 1999].

The idea here is that a candidate phrase is more likely to be a keyphrase if other authors have used it as a keyphrase. Frank *et al.* [1999] describe *keyphrase-frequency* as “domain-specific”, because intuition suggests it will only work well when the testing corpus is in the same domain (e.g., computer science) as the training corpus. This intuition is confirmed in the following experiments (Section 4).

3.3 Coherence Feature Set

The coherence feature set is calculated using a two-pass method. The first pass processes the candidate phrases using the baseline feature set. The second pass uses the top K most probable phrases, according to the probability estimates from the first pass, as a standard for evaluating the top L most probable phrases ($K < L$). In the second pass, for each of the top L candidates (including the top K candidates), new features are calculated based on the statistical association

¹ The following experiments use Kea version 1.1.4, which is available at <http://www.nzdl.org/Kea/>.

between the given candidate phrase and the top K phrases. The hypothesis is that candidates that are semantically related to one or more of the top K phrases will tend to be more coherent, higher quality keyphrases.

The coherence feature set consists of $4+2K$ features. The experiments use $K = 4$ and $L = 100$, thus there are twelve features. The first four features are based on the first pass with the baseline model. The first two features are $TF \times IDF$ and *distance*, copied exactly from the first pass. The third feature is *baseline_rank*, the rank of the given candidate phrase in the list of the top L phrases from the first pass, where the list is sorted in order of estimated probability. The fourth feature, *baseline_probability*, is the estimated probability, $p(key|T, D)$, according to the baseline model.

The remaining eight features are based on statistical association, calculated using Pointwise Mutual Information (PMI) and a web search engine [Turney, 2001]. The following experiments use the AltaVista search engine. Let $hits(query)$ be the number of hits (matching web pages) returned by AltaVista when given a query, *query*. Features five to eight are *rank_low₁*, *rank_low₂*, *rank_low₃*, and *rank_low₄*. For $i = 1$ to 100 and $j = 1$ to 4, *rank_low_j* is the normalized rank of the candidate phrase, *phrase_i*, when sorted by the following score:

$$score_low_j(phrase_i) = \frac{hits(low_i \text{ NEAR } low_j)}{hits(low_i)} \quad (2)$$

For the query to AltaVista, *phrase_i* is converted to lower case, *low_i*, but it is not stemmed. The rank of $score_low_j(phrase_i)$ is normalized by converting it to a percentile. The ranking and normalization are performed for each document individually, rather than the corpus as a whole. The query “*low_i NEAR low_j*” causes AltaVista to restrict the search to web pages in which *low_i* appears within ten words of *low_j*, in either order. Equation (2) can be derived from the formula for PMI [Turney, 2001]. The intent is that, with *rank_low₁* for example, if a candidate phrase, *phrase_i*, is strongly semantically connected with *phrase₁*, then *phrase_i* will likely receive a high score from Equation (2), and hence *rank_low₁* will tend to be relatively high.

Features nine to twelve are *rank_cap₁*, *rank_cap₂*, *rank_cap₃*, and *rank_cap₄*. For $i = 1$ to 100 and $j = 1$ to 4, *rank_cap_j* is the normalized rank of the candidate phrase, *phrase_i*, when sorted by the following score:

$$score_cap_j(phrase_i) = \frac{hits(cap_i \text{ AND } low_j)}{hits(cap_i)} \quad (3)$$

For the query to AltaVista, *phrase_i* is converted to *cap_i*, in which the first character of each word in *phrase_i* is capitalized, but the other characters are in lower case, and the words are not stemmed. The query “*cap_i AND low_j*” is intended to find web pages in which *phrase_i* appears in a title or heading (where it would likely be capitalized) and *phrase_j* appears anywhere in the page (where it would likely be in lower case). The rank of $score_cap_j(phrase_i)$ is normalized by converting it to a percentile. The plan is that *rank_cap₁*, for example, will be relatively high when the appearance of *phrase_i* in a heading makes it likely that *phrase₁* will appear in the body. That is, *phrase_i* would be a good phrase to put in the title of a document that discusses *phrase₁*.

3.4 Merged Feature Set

The merged feature set consists of the coherence feature set plus *keyphrase-frequency*. As with the coherence feature set, there is a two-pass process, but the first pass now uses the three features of the keyphrase frequency feature set, instead of the baseline features. The second pass now yields thirteen features, the twelve features of the coherence feature set plus *keyphrase-frequency*. The third and fourth features of the coherence feature set, *baseline_rank* and *baseline_probability*, become *key_freq_rank* and *key_freq_probability* in the merged feature set.

4 Experiments

The first experiment examines the behaviour of the four feature sets when trained and tested on the same domain. The second experiment trains on one domain (computer science) and tests on another (physics).

4.1 Intradomain Evaluation

This experiment compares the four feature sets using the setup of Frank *et al.* [1999]. The corpus is composed of computer science papers from the Computer Science Technical Reports (CSTR) collection in New Zealand. The training set consists of 130 documents and the testing set consists of 500 documents. The *keyphrase-frequency* feature is trained on a separate training set of 1,300 documents. All other features (including the coherence features) are trained using only the 130 training documents. The naïve Bayes algorithm allows the features to be trained separately, since the features are assumed to be statistically independent. The experiments of Frank *et al.* [1999] show that the *keyphrase-frequency* feature benefits from more training than the baseline features. Figure 1 plots the performance on the testing corpus.

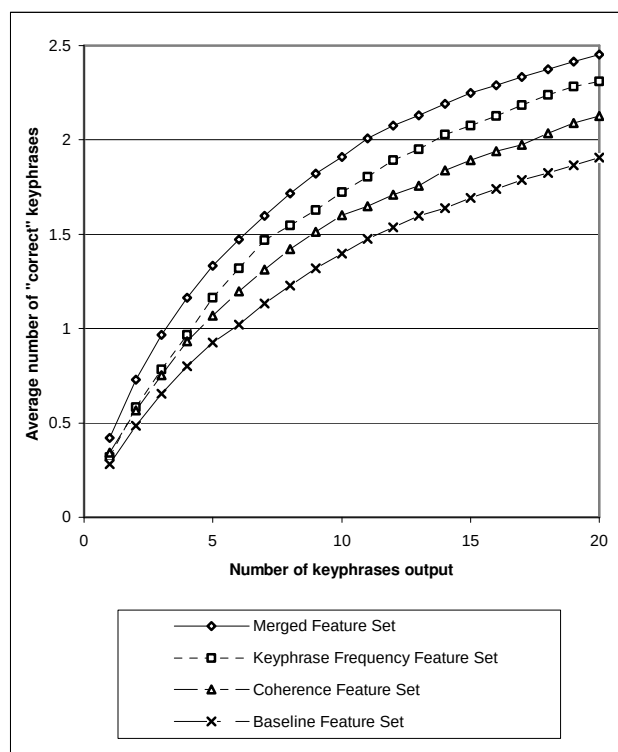


Figure 1. Performance on the CSTR corpus for the four different feature sets. An output keyphrase is considered “correct” when it equals an author-assigned keyphrase, after both are stemmed and converted to lower case.

The paired t-test was used to evaluate the statistical significance of the results [Feelders and Verkooijen, 1995]. There are six possible ways to pair the four curves in Figure 1. The differences between all six pairs are statistically significant with 95% confidence when five or more keyphrases are output. The merged feature set is superior to the keyphrase frequency feature set with 95% confidence when any number of keyphrases are output.

4.2 Interdomain Evaluation

This experiment evaluates how well the learned models generalize from one domain to another. The training domain is the CSTR corpus, using exactly the same training setup as in the first experiment. The testing domain consists of 580 physics papers from the arXiv repository at the Los Alamos National Laboratory (LANL). Figure 2 plots the performance on the testing corpus.

The paired t-test shows no significant differences between the keyphrase frequency feature set and the merged feature set. The coherence feature set is superior to the baseline feature set with 95% confidence when four or more keyphrases are output. The baseline feature set is superior to the keyphrase frequency and merged feature sets with 95% confidence when any number of keyphrases are output.

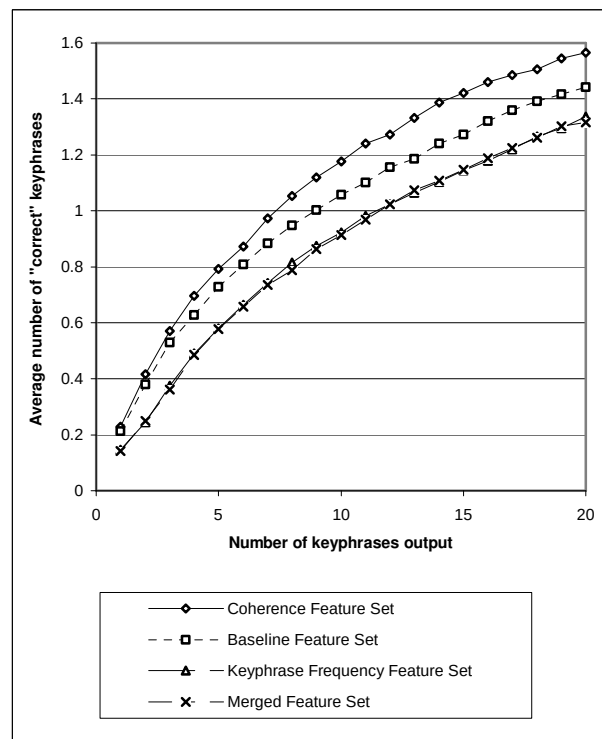


Figure 2. Performance on the LANL corpus for the four different feature sets. Note that the curves for the keyphrase frequency and merged features sets are intermingled, making it appear that there are only three curves, instead of four.

4.3 Discussion

Both experiments show that the coherence features are a significant improvement over the baseline features alone. More of the output keyphrases match with the authors’ keyphrases, which is evidence that their quality has improved [Jones and Paynter, 2001, 2002]. The second experiment confirms the (previously untested) hypothesis of Frank *et al.* [1999], that the *keyphrase-frequency* feature is domain-specific. This feature actually decreases performance below the baseline in the interdomain evaluation. However, the new coherence features are not domain-specific; they improve over the baseline even when the testing domain is different from the training domain. Furthermore, the coherence features can work synergistically with the *keyphrase-frequency* feature, when the training and testing domains correspond. When they do not correspond, at least the merged feature set is no worse than the keyphrase frequency feature set.

5 Limitations and Future Work

The main limitation of the new coherence feature set is the time required to calculate the features. It takes ten queries ($2+2K$, $K = 4$) to AltaVista to calculate one coherence feature vector. Each query takes about one second. At 10 queries per feature vector times 1 second per query times 100 feature vectors per document, we have 1,000 seconds

$((2+2K)L$, $K = 4$, $L = 100$; roughly 15 minutes) per document. The time required by the other aspects of Kea is completely insignificant compared to this. Future work might investigate the use of other, less time-consuming, measures of coherence, such as Morris and Hirst's [1991] measure. However, if we extrapolate current trends in hardware, then it seems that it will not be long before this is no longer an issue, since a local version of AltaVista will easily run on a desktop computer.

6 Conclusion

This paper provides evidence that statistical word association can be used to improve the coherence of keyphrase extraction, resulting in higher quality keyphrases, measured by the degree of overlap with the authors' keyphrases. Furthermore, the new coherence features are not domain-specific.

Acknowledgements

Thanks to Eibe Frank, Gordon Paynter, and the IJCAI reviewers for their helpful comments.

References

- [Barzilay and Elhadad, 1997] Barzilay, R., and Elhadad, M. Using lexical chains for text summarization. In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, 10-17, 1997.
- [Church and Hanks, 1989] Church, K.W., and Hanks, P. Word association norms, mutual information and lexicography. *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pp. 76-83, 1989.
- [Church et al., 1991] Church, K.W., Gale, W., Hanks, P., and Hindle, D. Using statistics in lexical analysis. In Uri Zernik (ed.), *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pp. 115-164. New Jersey: Lawrence Erlbaum, 1991.
- [Domingos and Pazzani, 1997] Domingos, P., and Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 103-130, 1997.
- [Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorization. *Proceedings of the Seventh International Conference on Information and Knowledge Management*, 148-155. ACM, 1998.
- [Fayyad and Irani, 1993] Fayyad, U.M., and Irani, K.B. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 1022-1027, 1993.
- [Feelders and Verkooijen, 1995] Feelders, A., and Verkooijen, W. Which method learns the most from data? Methodological issues in the analysis of comparative studies. *Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, pp. 219-225, 1995.
- [Frank et al., 1999] Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., and Nevill-Manning, C.G. Domain-specific keyphrase extraction. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pp. 668-673. California: Morgan Kaufmann, 1999.
- [Gutwin et al., 1999] Gutwin, C., Paynter, G.W., Witten, I.H., Nevill-Manning, C.G., and Frank, E. Improving browsing in digital libraries with keyphrase indexes. *Journal of Decision Support Systems*, 27, 81-104, 1999.
- [Halliday and Hasan, 1976] Halliday, M.A.K., and Hasan, R. *Cohesion in English*. London: Longman, 1976.
- [Jones and Paynter, 2001] Jones, S. and Paynter, G.W. Human evaluation of Kea, an automatic keyphrasing system. *First ACM/IEEE-CS Joint Conference on Digital Libraries*, Roanoke, Virginia, June 24-29, 2001, ACM Press, 148-156.
- [Jones and Paynter, 2002] Jones, S. and Paynter, G.W. Automatic extraction of document keyphrases for use in digital libraries: Evaluation and applications. *Journal of the American Society for Information Science and Technology (JASIST)*, 53 (8), 653-677, 2002.
- [Leung and Kan, 1997] Leung, C.-H., and Kan, W.-K. A statistical learning approach to automatic indexing of controlled index terms. *Journal of the American Society for Information Science*, 48, 55-66, 1997.
- [Manning and Schütze, 1999] Manning, C.D., and Schütze, H. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: MIT Press, 1999.
- [Morris and Hirst, 1991] Morris, J., and Hirst, G. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1), 21-48, 1991.
- [Turney, 1999] Turney, P.D. *Learning to Extract Keyphrases from Text*. National Research Council, Institute for Information Technology, Technical Report ERB-1057, 1999.
- [Turney, 2000] Turney, P.D. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2, 303-336, 2000.
- [Turney, 2001] Turney, P.D. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, Freiburg, Germany, pp. 491-502, 2001.
- [van Rijsbergen, 1979] van Rijsbergen, C.J. *Information Retrieval*. 2nd edition. London: Butterworths, 1979.
- [Witten et al., 1999] Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C. and Nevill-Manning, C.G. KEA: Practical automatic keyphrase extraction. *Proceedings of Digital Libraries 99 (DL'99)*, pp. 254-256. ACM Press, 1999.
- [Witten et al., 2000] Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., and Nevill-Manning, C.G. *KEA: Practical Automatic Keyphrase Extraction*. Working Paper 00/5, Department of Computer Science, The University of Waikato, 2000.

