Towards Efficient Belief Update for Planning-Based Web Service Composition

Jörg Hoffmann[1]

Abstract. At the "functional level", Semantic Web Services (SWS) are described akin to planning operators, with preconditions and effects relative to an ontology; the ontology provides the formal vocabulary and an axiomatisation of the underlying domain. Composing such SWS is similar to planning. A key obstacle in doing so effectively is handling the ontology axioms, which act as state constraints. Computing the outcome of an action involves the frame and ramification problems, and corresponds to belief update. The complexity of such updates motivates the search for tractable classes. Herein we investigate a class that is of practical relevance because it deals with many commonly used ontology axioms, in particular with attribute cardinality upper bounds which are not handled by other known tractable classes. We present an update computation that is exponential only in a comparatively uncritical parameter; we present an approximate update which is polynomial in that parameter as well.

1 Introduction

Semantic Web Services (SWS) are pieces of software advertised with a formal description of what they do; Web Service Composition (WSC) means to link them together in a way satisfying a complex user requirement. WSC is widely recognized for its economic potential. In the wide-spread OWL-S [3] and WSMO [5] frameworks, at the so-called "functional level" (which abstracts from interaction details and specifies only overall functionality), SWS are described akin to planning operators, with preconditions and effects relative to an ontology. Hence planning – planning under uncertainty, since information in the web context cannot be expected to be complete – is a prime candidate for realizing this form of WSC.

In our work, we pursue a kind of *conformant planning* [17]. The tool we develop performs a forward search as per Figure 1. Each s represents (partial) knowledge about the corresponding *belief* b, where as usual b is the set of all situations possible at the given point in time. Maintaining the states s is challenging because it involves a *belief update* problem. Namely, the main difference to most work in conformant planning is that we consider *state constraints*, e.g. [8, 2, 16]: the domain axiomatization given in the ontology. Such axioms are state constraints in the sense that any state that can be encountered, in the given domain, is known to satisfy them. In the presence of such axioms, computing the outcome of an action involves the frame and ramification problems: How do the axioms affect the previous world, and what are their side effects? Following various authors, e.g. [10, 15], we define action outcomes as belief updates, where the "update" is the action effect conjoined with the axioms.

Belief update has been shown to be hard even in tractable logics (e.g. Horn [4]). Since update is a frequently solved sub-problem in

s_0 := *initialise*(); open-list := $\langle s_0 \rangle$
while TRUE **do**
  s := choose(open-list)
  if *is-solution*(s) **then return** path leading to s
  for all calls a of SWS applicable in s **do**
    s' := *update*(s, a); insert(open-list,s')

Figure 1. The main loop of our planner.

planning as per Figure 1, the need for tractable classes is tantalising. In this context, it is of particular interest that practical WSC problems, e.g. the widely used Virtual Travel Agency (VTA) scenario, often come with fairly simple domain axiomatizations. Some of the most typically used axioms are: *subsumption relations*, which herein we write as clauses of the form $\forall x : train(x) \Rightarrow vehicle(x)$; *attribute range type restrictions* $\forall x, y : ticketfor(x, y) \Rightarrow person(y)$; *mutual exclusion* $\forall x : \neg train(x) \vee \neg car(x)$; and bounds on the number of distinct attribute values, such as the axiom $\forall x, y_1, y_2, y_3 : (ticketfor(x, y_1) \wedge ticketfor(x, y_2) \wedge ticketfor(x, y_3)) \Rightarrow (y_1 = y_2 \vee y_1 = y_3 \vee y_2 = y_3)$ which is a *cardinality upper bound* saying that at most two persons may travel on the same ticket.

The above raises the question which classes of axioms allow a polynomial time belief update. To our knowledge, the only existing work exploring this question is DL-Lite [6, 7], a fragment of DL for which belief update can be done efficiently, and the new belief can be represented in terms of a single ABox. The latter is necessary since the updated belief will be visible to the user.

DL-Lite does not allow cardinality upper bounds. In this paper, we identify a tractable fragment which includes such bounds. A key difference to DL-Lite is that we don't require beliefs to be understandable for a user: the representation is internal to the planner, and so we are completely free in how to define the search states s. We show that this enables us to deal with cardinality upper bounds, in time exponential only in the maximum bound k imposed by any such bound. The belief update algorithm we present deals also with *binary clauses*, i.e., clauses of at most two literals, such as subsumption relations, attribute range type restrictions, and mutual exclusion.

One would usually expect k to be 1 or 2 (rather than, say, 17). However, in large tasks the complexity of the update can become critical even for small k. We hence also pursue the idea of sacrificing either of soundness or completeness, for tractability. We present an approximate update algorithm that is polynomial also in k.

A few words are in order regarding our planning formalism. In difference to DL-Lite, and in line with the usual planning formalisms, we make a closed world assumption where a finite set of constants is fixed. The motivation for this is simply that it is closer to existing planning tools, and hence is expected to make it easier to eventually build on that work. The other main design decision regards the semantics of belief update. We adopt the *possible models approach*

[1] SAP Research, CEC Karlsruhe, Germany, joe.hoffmann@sap.com

(PMA)[18], which addresses the frame and ramification problems via a set-based notion of minimal change. Alternative semantics should be considered in the future: from an application perspective, at the time of writing there isn't sufficient material on concrete use cases in order to tell whether one or the other semantics is more practical. The PMA has been used in many recent works related to formal semantics for WSC, e.g. [15, 1, 6], and is hence somewhat canonical.[2]

Section 2 introduces our planning formalism. Section 3 establishes some core observations. Sections 4 and 5 present our exact respectively approximate update algorithms. Section 6 discusses closely related work and Section 7 concludes. For lack of space, we omit all proofs and many other details such as notions of, and algorithms for, output constants and a construct for more flexible updates of attribute values. The full paper is available as a TR [12].

2 WSC Formalism

Our formalism follows standard notions from conformant planning, extended by modelling constructs for axioms. Our terminology is as used in the WSC area; it should be obvious how this corresponds to planning terminology. We denote predicates with G, H, I, variables with x, y, and constants with c, d, e. We treat equality as a "built-in" predicate. *Literals* are possibly negated predicates whose arguments are variables or constants; if all arguments are constants, the literal is *ground*. Given a set X of variables, we denote by $\mathcal{L}^X$ the set of all literals which use only variables from X. If l is a literal, we write $l[X]$ to indicate that l uses variables X. If $X = \{x_1, \dots, x_k\}$ and $C = \{c_1, \dots, c_k\}$, then by $l[c_1, \dots, c_k/x_1, \dots, x_k]$ we denote the substitution, abbreviated $l[C]$. In the same way, we use substitution for any construct involving variables. By $\bar{l}$, we denote the inverse of l. If L is a set of literals, then $\overline{L} := \{\bar{l} \mid l \in L\}$ and $\bigwedge L := \bigwedge_{l \in L} l$.

An *ontology* Ω is a pair $(\mathcal{P}, \Phi)$ where $\mathcal{P}$ is a set of predicates and Φ is a conjunction of closed first-order formulas. We call Φ a *theory*. A *clause* is a disjunction of literals with universal quantification on the outside, e.g. $\forall x. \neg G(x) \vee H(x) \vee I(x)$. A clause is *binary* if it contains at most two literals. Φ is binary if it is a conjunction of binary clauses. The only non-binary clauses we will consider are *cardinality upper bounds*, taking the form $\forall x, y_1, \dots, y_{k+1}. (G(x, y_1) \wedge \dots G(x, y_{k+1})) \Rightarrow (y_1 = y_2 \vee y_1 = y_3 \vee \dots \vee y_k = y_{k+1})$; to simplify notation, we will refer to such a clause as $image(G) \leq k$. A theory is *binary with cardinality upper bounds* if it consists entirely of binary clauses and cardinality upper bounds. We will consider the special case where every predicate G with a bound $image(G) \leq k$ does not appear positively in any binary clause; we refer to such Φ as *binary with consequence-independent cardinality upper bounds*. Note that this includes subsumption relations, attribute range type restrictions, mutual exclusion, and cardinality upper bounds.

A *web service* w is a tuple $(X_w, \mathrm{pre}_w, \mathrm{eff}_w)$, where X_w is a set of variables (the *inputs*), pre_w is a conjunction of literals from $\mathcal{L}^{X_w}$ (the *precondition*), and eff_w is a conjunction of literals from $\mathcal{L}^{X_w}$ (the *effect*).[3] Before a web service can be applied, its inputs must be instantiated with constants, yielding a *service*; to avoid confusion with the search states s, we refer to services as *actions* a (which is in accordance with the usual planning terminology). Formally, for a web service $(X, \mathrm{pre}, Y, \mathrm{eff})$ and tuple of constants C_a, an action a is given by $(\mathrm{pre}_a, \mathrm{eff}_a) = (\mathrm{pre}, \mathrm{eff})[C_a/X]$. By convention, given an arbitrary action a, we will use C_a to denote a's input instantiation.

$\mathcal{WSC}$ *tasks* are tuples $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$. Ω is an ontology, $\mathcal{W}$ is a set of web services, and $\mathcal{C}$ is a set of constants. $\mathcal{U}$ is the user requirement, a pair $(\mathrm{pre}_\mathcal{U}, \mathrm{eff}_\mathcal{U})$ of precondition and effect. For complexity considerations, we will restrict $\mathcal{WSC}$ tasks to have *fixed arity*, meaning a constant upper bound on predicate arity, the number of parameters of any web service, and the depth of quantifier nesting in Φ. Further, we will sometimes assume *fixed maximum cardinality*, meaning a constant upper bound on k in any axiom $image(G) \leq k$.

The semantics of our formalism relies on a notion of *beliefs*, where each belief is a set of *models*. Each model is an interpretation of all propositions formed from $\mathcal{P}$ and $\mathcal{C}$. The *initial belief* b_0 is undefined if $\Phi \wedge \mathrm{pre}_\mathcal{U}$ is not satisfiable; else, $b_0 := \{m \mid m \models \Phi \wedge \mathrm{pre}_\mathcal{U}\}$. A *solved belief* is a belief b s.t., for all $m \in b$, $m \models \mathrm{eff}_\mathcal{U}$.

It remains to define how actions affect models and beliefs. Say m is a model and a is an action; as stated, we define the outcome $Res(m, a)$ following [18]. We say that a is *applicable* in m if $m \models \mathrm{pre}_a$. If a is not applicable in m, then $Res(m, a)$ is undefined. Otherwise, $Res(m, a) := \{m' \mid m' \in min(m, \Phi \wedge \mathrm{eff}_a)\}$. Here, $min(m, \phi)$ is the set of all m' that satisfy ϕ and that are minimal with respect to the partial order defined by $m_1 \leq m_2$:iff for all propositions p, if $m_2(p) = m(p)$ then $m_1(p) = m(p)$. That is, m' differs in a set-inclusion minimal subset of values from m.

Say b is a belief. $Res(b, a)$ is undefined if there exists $m \in b$ so that $Res(m, a)$ is undefined, or so that $Res(m, a) = \emptyset$. Else, $Res(b, a) := \bigcup_{m \in b} Res(m, a)$. The Res function is extended to sequences $\langle a_1, \dots, a_n \rangle$ in the obvious way. A *solution* is a sequence $\langle a_1, \dots, a_n \rangle$ s.t. $Res(b_0, \langle a_1, \dots, a_n \rangle)$ is a solved belief.

Example 1 *Given predicate ticketfor with $image(ticketfor) \leq 2$, and constants $t, Peter, Bob, Mary$. Initially, $ticketfor(t, Peter) \wedge ticketfor(t, Bob)$. Say we apply a_1 with effect $ticketfor(t, Mary)$. We get two resulting states, one with $ticketfor(t, Peter) \wedge ticketfor(t, Mary)$ and one with $ticketfor(t, Bob) \wedge ticketfor(t, Mary)$ (but none with only $ticketfor(t, Mary)$, since that would not be a minimal change). Say we now apply a_2 with effect $ticketfor(t, Peter)$. We get two states, with $ticketfor(t, Peter) \wedge ticketfor(t, Mary)$ and $ticketfor(t, Peter) \wedge ticketfor(t, Bob)$, respectively.*

3 Basic Observations

We make a number of basic observations: lemmas used in our update computations, and negative results supporting our design decisions. We first make some general observations about belief intersections, then we consider binary clauses and cardinality upper bounds.

Before thinking about how to update beliefs, one needs to think about how to represent beliefs, and, even, which aspects of beliefs to represent. Every belief may contain an exponential number of different models, and hence symbolic representations should be utilized, and/or only a partial knowledge should be maintained. Herein, we focus on the latter. Inspired by recent techniques from conformant planning [13] (with no state constraints), we aim at maintaining only belief *intersections*: the set of literals that are true in all models of a belief b, $\bigcap_{m \in b} \{l \mid m \models l\} =: \bigcap b$. Based on $\bigcap b$, we can determine whether an action a is applicable to b, namely iff $\mathrm{pre}_a \subseteq \bigcap b$, and whether b is solved, namely iff $\mathrm{eff}_\mathcal{U} \subseteq \bigcap b$. So, ideally, we wish to define the search states s from Figure 1 as sets L_s of literals: if b is a belief and s the corresponding search state, then we want to have $L_s = \bigcap b$. The question is, how do we maintain those s?

[2] Notably, one of the main arguments made against the PMA, e.g. by [2, 16, 11] is that it lacks a notion of causality. However, ontology languages such as OWL do not model causality; all we are given is a set of axioms. Hence this criticism does not apply for WSC (unless one proposes an entirely new framework for modelling web services, which is not our focus here).

[3] Note that this definition of preconditions and effects (conjunctions of literals) is quite restrictive. This is intended since we're looking for tractable classes in here. It remains to be verified in future work if and inhowfar this restriction can be relaxed without losing our tractability results.

First, one piece of bad news is that computing $\bigcap Res(m,a)$ is very hard in general, and is hard even if Φ is Horn. This follows directly from earlier results in the area of belief update [4]:

Proposition 1 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ with fixed arity. Assume a model m, an action a, and a literal l such that $m \models l$. It is Π_2^p-complete to decide whether $l \in \bigcap Res(m,a)$. If Φ is Horn, then the same decision is coNP-complete.*

This shows in particular that it is not necessarily enough to restrict ourselves to a tractable logics for Φ – at least in the case of Horn logics, that does not make the update problem tractable. The question arises whether the same is the case for binary clauses. As one might suspect, the answer is "no". The following two technical observations can be used to prove this fact; they are also used further below to prove the correctness of our update computations.

First, literals $l \in \bigcap Res(b,a)$ do not appear "out of thin air":

Lemma 1 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$. Assume a belief b and an action a. Then $\bigcap Res(b,a) \subseteq \{l \mid \Phi \wedge \mathit{eff}_a \models l\} \cup \bigcap b$.*

This is due to the PMA, which, if $l \notin \bigcap b$ and $\Phi \wedge \text{eff}_a \not\models l$, generates $m' \in Res(b,a)$ so that $m' \not\models l$.

Lemma 1 means that, in general, $\bigcap Res(b,a)$ can be computed in two steps: (A) determine $\{l \mid \Phi \wedge \text{eff}_a \models l\}$; (B) determine which $l \in \bigcap b$ do not disappear, i.e., $l \in \bigcap Res(b,a)$. Obviously, (A) is just deduction in Φ. The more tricky part is (B). The following observation characterizes exactly when $l \in \bigcap b$ disappears:

Lemma 2 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$. Assume a belief b, an action a, and a literal $l \in \bigcap b$. Then, $l \notin \bigcap Res(b,a)$ iff there exists a set L_0 of literals satisfied by a model $m \in b$, such that $\Phi \wedge \mathit{eff}_a \wedge \bigwedge L_0$ is satisfiable and $\Phi \wedge \mathit{eff}_a \wedge \bigwedge L_0 \wedge l$ is unsatisfiable.*

Intuitively, L_0 is the "reason" why l disappears: it is consistent with the effect and hence true in a model of $Res(b,a)$; but it excludes l. We can conclude that, for binary clauses, a literal disappears only if its opposite is necessarily true:

Lemma 3 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ where Φ is binary. Assume a belief b, an action a, and a literal $l \in \bigcap b$. If $l \notin \bigcap Res(b,a)$, then $\Phi \wedge \mathit{eff}_a \wedge l$ is unsatisfiable.*

Namely: by Lemma 2 there exists L_0 so that $\Phi \wedge \bigwedge L_0 \wedge l$ is satisfiable, but $\Phi \wedge \text{eff}_a \wedge \bigwedge L_0 \wedge l$ is unsatisfiable; with binary Φ, this implies that $\Phi \wedge \text{eff}_a \wedge l$ is unsatisfiable. By Lemmas 1 and 3, and since reasoning in grounded binary Φ is polynomial, we get:

Corollary 1 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ with fixed arity, where Φ is binary. Assume a belief b, and an action a; let $L := \{l \mid \Phi \wedge \mathit{eff}_a \models l\}$. Then $\bigcap Res(b,a) = L \cup (\bigcap b \setminus \overline{L})$. Given $\bigcap b$, this can be computed in time polynomial in the size of $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$.*

Corollary 1 is a moderately interesting result since binary clauses are somewhat complementary to DL-Lite. The more important use of Lemmas 1, 2, and 3 will be below where we consider the combination of binary clauses with cardinality upper bounds. Our first observation regarding that combination is:

Proposition 2 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ with fixed arity, where Φ is binary with cardinality upper bounds. Deciding whether Φ is satisfiable is NP-complete.*

By a straightforward reduction from VERTEX COVER. We sidestep this source of intractability by restricting ourselves to Φ that are binary with *consequence-independent* cardinality upper bounds (c.f. Section 2): any predicate G with a bound $image(G) \leq k$ does not appear positively in the binary clauses. Note that G appears only negatively in the clause $image(G) \leq k$. This removes the problem:

Lemma 4 *Let ϕ be a propositional CNF, with $\phi = \phi_1 \wedge \phi_2$ where there exists no literal l s.t. l appears in ϕ_1 and and $\bar{l}$ appears in ϕ_2. Let l be a literal s.t. $\phi \models l$. Then either $\phi_1 \models l$ or $\phi_2 \models l$.*

This is easy to see based on the lack of conflicts between ϕ_1 and ϕ_2. A more subtle point is that even dealing with cardinality upper bounds in isolation is tricky. Namely, it is not possible to compute $\bigcap Res(b,a)$ based only on $\bigcap b$:

Proposition 3 *There exist a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ where Φ consists entirely of cardinality upper bounds, an action a, and two reachable beliefs b and b' s.t. $\bigcap b = \bigcap b'$, but $\bigcap Res(b,a) \neq \bigcap Res(b',a)$.*

A model m may disappear when applying an action a', and not be re-created when a' is inverted. This leads to beliefs b where $b \neq \{m \mid m \models \Phi \wedge \bigcap b\}$,[4] and further to b, b' s.t. $\bigcap b = \bigcap b'$ but $b \neq b'$.

This means that it is *not* possible to, as envisioned, define the search states s simply as sets L_s – at least not if we want to ensure that L_s is exactly the intersection of the corresponding belief. We need to augment s with additional information. We have experimented for some time with methods augmenting s with the min and max number of attribute values present in any model of the belief. The intuition behind such an approach would be that cardinality upper bounds affect only *how many*, not *which* attribute values there are. However, this is not true since the cardinality upper bounds are intermingled with action effects; this makes capturing the precise distribution of attribute value tuples a surprisingly tricky task.

It remains an open question whether beliefs in the presence of cardinality upper bounds can be represented concisely. Herein, we present two alternative options. The first option, Section 4, takes time and space that is exponential (only) in the maximum k of any upper bound $image(G) \leq k$. The second option, Section 5, takes polynomial time also in k, but sacrifices precision and guarantees only one of soundness or completeness (the user may choose which one).

4 Exact Belief Update

We now specify search states s and associated *initialise* and *update* procedures that enable us to maintain precise belief intersections. We need three notations. First, by $\Phi_{|2}$, we denote the subset of binary clauses of Φ. Second, if L is a set of literals, G is a predicate with arity 2, and c is a constant, then we denote $L_{|G,c} := \{d \mid G(c,d) \in L\}$. That is, $L_{|G,c}$ selects from L the values of attribute G for c. Similarly, $L_{|-G,c} := \{d \mid \neg G(c,d) \in L\}$. Third, say b is a belief; we introduce a formal notation for the precise distribution, denoted $\mathcal{D}_b$, of attribute value tuples. Our search states will explicitly keep track of that distribution, and hence contain suficient information for precise belief update (this is not possible based only on $\bigcap b$, c.f. Proposition 3). $\mathcal{D}_b$ maps any G where $image(G) \leq k$ in Φ, and any $c \in \mathcal{C}$, onto a set of subsets of $\mathcal{C}$. Namely, for each $m \in b$, $\mathcal{D}_b(G,c)$ contains the set $\{d \mid m \models G(c,d)\}$. Hence, for every G and c, $\mathcal{D}_b(G,c)$ specifies which combinations of attribute values occur.

Our search states s are pairs $(L_s, \mathcal{D}_s)$. Consider Figures 2 and 3. In lines (1) to (3), Figure 2 determines all logical consequences, L, of the initial literals and the binary part of Φ, and checks whether L is contradictory. Thereafter, cardinality upper bounds are handled; note that this can be done separately because of Lemma 4. Line (5) detects any violated upper bounds. Line (6) says that, for any cardinality upper bound where we already have the maximum number

[4] This relates to [14], who show that DL updates can often not be represented in terms of a single changed ABox.

procedure *initialise*()
(1) $L^{\text{pre}_{\mathcal{U}}} := \{l \mid l$ appears in $\text{pre}_{\mathcal{U}}\}$
(2) $L := \{l \mid \Phi_{|2} \wedge \bigwedge L^{\text{pre}_{\mathcal{U}}} \models l\}$
(3) **if** ex. l s.t. $l \in L$ and $\bar{l} \in L$ **then return** (undefined)
(4) **for** all $image(G) \leq k$ in Φ, $c \in \mathcal{C}$ **do**
(5) **if** $|L^{\text{pre}_{\mathcal{U}}}_{|G,c}| > k$ **then return** (undefined)
(6) **if** $|L^{\text{pre}_{\mathcal{U}}}_{|G,c}| = k$ **then**
$L := L \cup \{\neg G(c,d) \mid d \in \mathcal{C}, d \notin L^{\text{pre}_{\mathcal{U}}}_{|G,c}\}$
(7) $\mathcal{D}(G,c) := \{D \mid D \subseteq \mathcal{C}, L^{\text{pre}_{\mathcal{U}}}_{|G,c} \subseteq D,$
$D \cap L_{|-G,c} = \emptyset, |D| \leq k\}$
(8) **return** $(L, \mathcal{D})$

Figure 2. The *initialise* procedure for exact search states.

of allowed attribute values, all other values are disallowed. Line (7) sets the $\mathcal{D}(G,c)$ value combination sets as appropriate, taking every combination that adheres to all constraints.

procedure *update*(s,a)
(1) **if** $\text{pre}_a \not\subseteq L_s$ **then return** (undefined)
(2) $L^A := \{l \mid l$ appears in $\text{eff}_a\}$
(3) $L := \{l \mid \Phi_{|2} \wedge \bigwedge L^A \models l\}$
(4) **if** ex. l s.t. $l \in L$ and $\bar{l} \in L$ **then return** (undefined)
(5) **for** all $image(G) \leq k$ in Φ, $c \in \mathcal{C}$ **do**
(6) **if** $|L^A_{|G,c}| > k$ **then return** (undefined)
(7) **if** $|L^A_{|G,c}| = k$ **then**
$L := L \cup \{\neg G(c,d) \mid d \in \mathcal{C}, d \notin L^A_{|G,c}\}$
(8) $\mathcal{D}(G,c) := \emptyset$
(9) $L^{AT} := L$; $L := L \cup \{l \mid l \in L_s, \bar{l} \notin L\}$
(10) **for** all $image(G) \leq k$ in Φ, $c \in \mathcal{C}$, $D \in \mathcal{D}_s(G,c)$ **do**
(11) **if** $|D \cup L^A_{|G,c} \setminus L^{AT}_{|-G,c}| > k$ **then**
(12) $L := L \setminus \{G(c,d) \mid G(c,d) \in L_s \setminus L^A\}$
(13) $\mathcal{D}(G,c) := \mathcal{D}(G,c) \cup$
$\{D' \cup L^A_{|G,c} \mid D' \subseteq D \setminus (L^A_{|G,c} \cup L^{AT}_{|-G,c}),$
$|D'| = k - |L^A_{|G,c}|\}$
(14) **else** $\mathcal{D}(G,c) := \mathcal{D}(G,c) \cup \{D \cup L^A_{|G,c} \setminus L^{AT}_{|-G,c}\}$
(15) **return** $(L, \mathcal{D})$

Figure 3. The *update* procedure for exact search states.

The *update* procedure, Figure 3, is more complicated. Line (1) tests whether a is applicable. Lines (2) to (7) are analogous to lines (1) to (6) of Figure 2. Line (8) initialises the $\mathcal{D}$ structures. Line (9) extends L with all literals from L_s, except those that are contradicted by L. By Lemma 1, the resulting L is a superset of $\bigcap Res(b,a)$. By Lemma 3, as far as binary clauses are concerned, the resulting L is equal to $\bigcap Res(b,a)$. For cardinality upper bounds, Lemma 3 does not apply, which necessitates lines (10) to (12) to check if further "old" belief intersection literals disappear. Namely, applying Lemma 2, an old attribute value (even if it is not contradicted) survives only if there exists no model $m \in b$ so that, after the effects and their direct consequences have been applied, m contains too many attribute values. To figure out whether or not the latter is the case, the information given by $\mathcal{D}_s$ is exploited, in a straightforward way. (Note that this information is indeed required here. Assume that all we know is the maximum number of attribute values in any model $m \in b$. Then we would not know whether or not these are the same values as set by the action effects, and hence we could not decide whether or not an overflow occurs.)

Lines (13) and (14), finally, make sure that $\mathcal{D}$ is updated correctly. If an overflow occurs, then all possible ways of minimally repairing the overflow are generated. If no overflow occurs, then $\mathcal{D}(G,c)$ simply changes according to the effect and its implications. We have:

Theorem 1 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ where Φ is binary with consequence-independent cardinality upper bounds. Assume b is a reachable belief, and s is the corresponding search state. Then: (1) b is defined iff s is defined; (2) if b is defined, then $\bigcap b = L_s$; (3) if b is defined, then $\mathcal{D}_b \equiv \mathcal{D}_s$.*

The formal proof of Theorem 1 is quite lenghty, and involves various (sometimes rather tedious) case distinctions. The proof essentially spells out the intuitive arguments given above. Our main result here is that, provided a maximum cardinality is fixed, maintaining belief intersections is tractable:

Corollary 2 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ with fixed arity and fixed maximum cardinality, where Φ is binary with consequence-independent cardinality upper bounds. Assume b is reached by action sequence $\vec{a}$. Then the corresponding search state s is computed in time polynomial in the size of $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ and $\vec{a}$, and $\bigcap b = L_s$.*

Note that it is indeed a non-trivial consequence of our particular setting that the behavior is exponential only in the maximum k of any $image(G) \leq k$. The enabling properties are: (1) $image(G) \leq k$ does not interfere in any way with $image(H) \leq k$, if $G \neq H$; (2) similarly, the bound on the number of y in $G(c,y)$ does not interfere with the bound on y in $G(c',y)$ if $c \neq c'$; (3) due to consequence-independence, no interferences arise from the binary clauses.

Example 2 *Re-consider Example 1. Running* initialise*, we get the state s_0 where $L = \{ticketfor(t, Peter), ticketfor(t, Bob)\}$ and $\mathcal{D}(ticketfor, t) = \{\{Peter, Bob\}\}$. Applying a_1, we get $s_1 =$ update(s_0, a_1) where $L = \{ticketfor(t, Mary)\}$ and $\mathcal{D}(ticketfor, t) = \{\{Peter, Mary\}, \{Bob, Mary\}\}$. Applying a_2, we get $s_2 =$ update(s_0, a_2) where $L = \{ticketfor(t, Peter)\}$ and $\mathcal{D}(ticketfor, t) = \{\{Peter, Mary\}, \{Bob, Peter\}\}$.*

5 Approximate Belief Update

Even though it seems likely that k will be small in practice, it is advisable to look for more efficient methods. The size of $\mathcal{D}(G,c)$ is bounded only by $\binom{|\mathcal{C}|}{k}$. If there are many constants, then enumerating $\mathcal{D}$ will become critical even for, say, $k > 2$. We now tackle this complexity by approximation methods. The search states s are pairs (L^-_s, L^+_s) where L^-_s and L^+_s respectively under-approximate and over-approximate the belief intersection. Both approximations are maintained simultaneously because they are interlinked. Depending on how one tests action applicability and solutions, one obtains a pessimistic/sound (but incomplete) planning procedure, or an optimistic/complete (but unsound) planning procedure. We show here only the former; the latter can be obtained by minor modifications.

The *initialise* procedure changes only slightly because, there, no update is performed. In fact the procedure is exactly as shown in Figure 2, except that the returned s takes the form (L, L) where L – the precise belief intersection – serves both as L^- and as L^+.

Consider Figure 4. Line (1) tests pessimistically whether a is not applicable: the preconditions are tested against L^-_s. Thereafter, lines (2) and (3) determine the effects and their implications over the binary clauses. Line (4) tests for contradictions in the latter. Similarly, line (6) aborts the algorithm in case of a conflict with a cardinality upper bound (separate treatment of the two kinds of conflicts is justified by Lemma 4). Line (7) adds the consequences of the upper bounds to the implied literals.

procedure *update*(s, a)

(1) **if** $\text{pre}_a \not\subseteq L_s^-$ **then return** (undefined)
(2) $L^A := \{l \mid l \text{ appears in eff}_a\}$
(3) $L := \{l \mid \Phi_{|2} \wedge \bigwedge L^A \models l\}$
(4) **if** ex. l s.t. $l \in L$ and $\bar{l} \in L$ **then return** (undefined)
(5) **for** all $image(G) \leq k$ in Φ, $c \in \mathcal{C}$ **do**
(6)   **if** $|L^A_{|G,c}| > k$ **then return** (undefined)
(7)   **if** $|L^A_{|G,c}| = k$ **then**
        $L := L \cup \{\neg G(c, d) \mid d \in \mathcal{C}, d \notin L^A_{|G,c}\}$
(8) $L^- := L \cup \{l \mid l \in L_s^-, \bar{l} \notin L\}$
(9) $L^+ := L \cup \{l \mid l \in L_s^+, \bar{l} \notin L\}$
(10) **for** all $image(G) \leq k$ in Φ, $c \in \mathcal{C}$ **do**
(11)   **if** $|L^-_{|G,c}| > k$ **then**
        $L^+ := L^+ \setminus \{G(c, d) \mid G(c, d) \in L_s^+ \setminus L^A\}$
(12)   **if** $|(\mathcal{C} \setminus L^-_{|-G,c}) \cup L^A_{|G,c}| > k$ **then**
        $L^- := L^- \setminus \{G(c, d) \mid G(c, d) \in L_s^- \setminus L^A\}$
(13) **return** (L^-, L^+)

Figure 4. The *update* procedure for approximate search states.

Lines (8) and (9) initialise the consideration of old intersection literals. All of those which are not contradicted are taken into a respective approximate set (c.f. Lemmas 1 and 3). Line (11) says that, if even the under-approximation violates a bound, then certainly the old attribute values get lost unless they are protected by the effect (c.f. Lemma 2). Line (12) says that, if the number of constants that could potentially be attribute values violates a bound, then it may happen that the old attribute values get lost, unless they are protected by the effect (c.f. Lemma 2). Note that the order of lines (11) and (12) is important because line (12) changes L^-. If one executes (12) before (11), then the condition of (11) is always false, and L^+ is still an over-approximation but an unnecessarily generous one. We get:

Theorem 2 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ where Φ is binary with consequence-independent cardinality upper bounds. Assume b is a reachable belief, and s is the corresponding approximate search state. Then: (1) if b is undefined, then s is undefined; (2) if s is defined, then $L_s^- \subseteq \bigcap b \subseteq L_s^+$.*

As for Theorem 1, the proof of Theorem 2 is lenghty and involves various case distinctions. Our main result of this section is:

Corollary 3 *Assume a $\mathcal{WSC}$ task $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ with fixed arity, where Φ is binary with consequence-independent cardinality upper bounds. Assume b is reached by action sequence $\vec{a}$. If the corresponding approximate search state s is defined, then s is computed in time polynomial in the size of $(\Omega, \mathcal{W}, \mathcal{C}, \mathcal{U})$ and $\vec{a}$, and $\bigcap b \supseteq L_s^-$.*

Example 3 *Re-consider Example 1. Running* initialise*, we get the state s_0 where $L^- = L^+ = \{ticketfor(t, Peter), ticketfor(t, Bob)\}$. Applying a_1, both lines (11) and (12) fire and so we get $s_1 =$* update(s_0, a_1) *where $L^- = L^+ = \{ticketfor(t, Mary)\}$. Applying a_2, only line (12) fires and so we get $L^- = \{ticketfor(t, Peter)\}$ and $L^+ = \{ticketfor(t, Mary), ticketfor(t, Peter)\}$.*

6 Related Work

[6] introduces DL-Lite, where the updated belief can be represented in terms of a new ABox computed in polynomial time. DL-Lite is somewhat complementary to binary clauses. Disjunction is allowed only in the form of subsumption rules in the TBox, and is binary in that sense. However, [6] allow unqualified existential quantification, membership assertions (ABox literals) using variables, and updates involving general (constructed) DL concepts. On the other hand, DL-Lite does not allow clauses with two positive literals, and DL-Lite (like any DL) does not allow predicates of arity greater than 2. Most importantly, DL-Lite does not allow cardinality upper bounds.

[7] considers a variant of DL-Lite where ABox assertions do not allow variables, and hence updates cannot be represented in terms of a new ABox. [7] show that the update from [6] can be re-used to compute the exact set of (restricted) ABox assertions after the update; this approximates the update in the sense that this set of assertions does not suffice to characterize the exact set of models. This is quite different from our approximation techniques as per Section 5, where we use approximation (without exactness guarantees) not to handle a different language, but to obtain efficiency.

[9, 10] address planning with belief update semantics (other than the PMA); they do not identify tractable classes.

7 Conclusion

In planning-based WSC, one of the fundamental difficulties is the complexity of computing the outcome of actions. Since practical domain axiomatizations for WSC are often simple, there is hope to tackle this complexity by identifying tractable fragments. We make a first step in this direction, showing how cardinality upper bounds can be handled, in combination with binary clauses. Many questions are left open. For example: Are our algorithms here the best possible ones, or is there an exact update algorithm that is polynomial also in k? Can one efficiently deal with cardinality lower bounds? We hope that some of these questions will be clarified in future work.

REFERENCES

[1] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter, 'Integrating description logics and action formalisms: First results', in *AAAI*, (2005).

[2] G. Brewka and J. Hertzberg, 'How to do things with worlds: On formalizing actions and plans', *JLC*, **3**, 517–532, (1993).

[3] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services, 2003.

[4] T. Eiter and G. Gottlob, 'On the complexity of propositional knowledge base revision, updates, and counterfactuals', *AI*, **57**, 227–270, (1992).

[5] D. Fensel, H. Lausen, A. Polleres, J. deBruijn, M. Stollberg, D. Roman, and J. Domingue, *Enabling Semantic Web Services – The Web Service Modeling Ontology*, Springer-Verlag, 2006.

[6] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati, 'On the update of description logic ontologies at the instance level', in *AAAI*, (2006).

[7] G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati, 'On the approximation of instance level update and erasure in DL', in *AAAI*, (2007).

[8] M. Ginsberg and D. Smith, 'Reasoning about action I: A possible worlds approach', *AI*, **35**, 165–195, (1988).

[9] J. Hertzberg and S. Thiebaux, 'Turning an action formalism into a planner - a case study', *JLC*, **4**, 617–654, (1994).

[10] A. Herzig, J. Lang, P. Marquis, and T. Polacsek, 'Updates, actions, and planning', in *IJCAI*, (2001).

[11] Andreas Herzig and Omar Rifi, 'Propositional belief base update and minimal change', *AI*, **115**, 107–138, (1999).

[12] J. Hoffmann. Towards efficient belief update for planning-based web service composition, 2008. Available at http://members.deri.at/ joergh/papers/tr-ecai08.pdf.

[13] Jörg Hoffmann and Ronen Brafman, 'Conformant planning via heuristic forward search: A new approach', *AI*, **170**(6–7), 507–541, (2006).

[14] H. Liu, C. Lutz, M. Milicic, and F. Wolter, 'Updating description logic ABoxes', in *KR*, (2006).

[15] Carsten Lutz and Ulrike Sattler, 'A proposal for describing services with DLs', in *DL*, (2002).

[16] N. McCain and H. Turner, 'A causal theory of ramifications and qualifications', in *IJCAI*, (1995).

[17] D. E. Smith and D. Weld, 'Conformant Graphplan', in *AAAI*, (1998).

[18] M. Winslett, 'Reasoning about actions using a possible models approach', in *AAAI*, (1988).