Towards Efficient Belief Update for Planning-Based Web Service Composition

Jörg Hoffmann

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 shown in 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 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 ∀x : train(x) ⇒ vehicle(x); attribute range type restrictions ∀x, y : ticketfor(x,y) ⇒ person(y); mutual exclusion ∀x : ¬train(x) ∨ ¬car(x); and bounds on the number of distinct attribute values, such as the axiom ∀x, y1, y2, y3 : (ticketfor(x,y1) ∧ ticketfor(x,y2) ∧ ticketfor(x,y3)) ⇒ (y1 = y2 ∨ y1 = y3 ∨ y2 = y3) 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, jorg.hoffmann@sap.com
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 \( e, d, c \). 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 \( X = \{ x_1, \ldots, x_k \} \) and \( C = \{ c_1, \ldots, c_k \} \), then by \( l[c_1, \ldots, c_k/x_1, \ldots, x_k] \) we denote the substitution, abbreviated \( l[C] \). In the same way, we use substitution for any construct involving variables. By \( T \), we denote the inverse of \( l \). If \( L \) is a set of literals, then \( T_L := \{ l \mid l \in L \} \) and \( L \setminus T_L := \{ l \mid l \notin L \} \).

An ontology \( \Omega \) is a pair \((P, \Phi)\) where \( P \) is a set of predicates and \( \Phi \) is a conjunction of closed first-order formulas. We call \( \Phi \) a theory. A clause is a disjunction of literals with universal quantification on the outside, e.g. \( \forall x. G(x) \lor H(x) \lor I(x) \). A clause is binary if it contains at most two literals. \( \Phi \) 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 \( \forall x, y_1, \ldots, y_{k+1}. (G(x, y_1) \land \ldots \land G(x, y_{k+1})) \rightarrow (y_1 = y_2 \lor y_1 = y_3 \lor \ldots \lor y_k = y_{k+1}) \); to simplify notation, we will refer to such a clause as an image \( \Phi \leq k \). 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 \( \Phi \leq k \) does not appear positively in any binary clause; we refer to such \( \Phi \) as binary with consequence-independent cardinality upper bounds. Note that this includes subsumption relations, attribute range restrictions, mutual exclusion, and cardinality upper bounds.

A web service \( w \) is a tuple \((X_w, \text{pre}_w, \text{eff}_w)\), where \( X_w \) is a set of variables (the inputs), \( \text{pre}_w \) is a conjunction of literals from \( \mathcal{L}X_w \) (the precondition), and \( \text{eff}_w \) is a conjunction of literals from \( \mathcal{L}X_w \) (the effect). 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

---

脚注:

2. 虽然，主要的论证方式与 PMA 有关，例如由 [2, 16, 11] 所示，它缺乏因果关系。但是，语言学对 OWL 不做模型可 decidability；对我们来说是给定了一个新框架用于建模 web 服务，我们不关注此焦点。

3. 注意，定义的先决条件和效果（句法一致的定理）是相当有效的。这是为了使得对于可表的类是有效的。这在将来可能会彻底验证到未来工作，但既而这样的限制可以被放松而不会丢失可表性结果。
First, one piece of bad news is that computing $\bigcap \text{Res}(m, a)$ is very hard in general, and is hard even if $\Phi$ is Horn. This follows directly from earlier results in the area of belief update [4]:

**Proposition 1** Assume a WSC task $(\Omega, W, C, U)$ with fixed arity. Assume a model $m$, an action $a$, and a literal $l$ such that $m \models l$. It is $\Pi^0_2$-complete to decide whether $l \in \bigcap \text{Res}(m, a)$. If $\Phi$ is Horn, then the same decision is $\text{coNP}$-complete.

This shows in particular that it is not necessarily enough to restrict ourselves to a tractable logics for $\Phi$ – 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 $l \in \bigcap \text{Res}(b, a)$ do not appear “out of thin air”:

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

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

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

**Lemma 2** Assume a WSC task $(\Omega, W, C, U)$. Assume a belief $\Phi$, an action $a$, and a literal $l \in \bigcap b$. Then, $l \not\in \bigcap \text{Res}(b, a)$ if there exists a set $L_0$ of literals satisfied by a model $m \in b$, such that $\Phi \land \text{eff}_a \land L_0$ is satisfiable and $\Phi \land \text{eff}_a \land L_0 \land \lnot l$ is unsatisfiable.

Intuitively, $L_0$ is the “reason” why $l$ disappears: it is consistent with the effect and hence true in a model of $\text{Res}(b, a)$; 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 WSC task $(\Omega, W, C, U)$ where $\Phi$ is binary. Assume a belief $\Phi$, an action $a$, and a literal $l \in \bigcap b$. If $l \not\in \bigcap \text{Res}(b, a)$, then $\Phi \land \text{eff}_a \land l$ is unsatisfiable.

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

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

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 WSC task $(\Omega, W, C, U)$ with fixed arity, where $\Phi$ is binary with cardinality upper bounds. Deciding whether $\Phi$ is satisfiable is $\text{NP}$-complete.

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

**Lemma 4** Let $\phi$ be a propositional CNF, with $\phi = \phi_1 \land \phi_2$ where there exists no literal $l$ s.t. $l$ appears in $\phi_1$ and $l$ appears in $\phi_2$. 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 $\phi_1$ and $\phi_2$. A more subtle point is that even dealing with cardinality upper bounds in isolation is tricky. Namely, it is not possible to compute $\bigcap \text{Res}(b, a)$ based only on $\bigcap b$:

**Proposition 3** There exist a WSC task $(\Omega, W, C, U)$ where $\Phi$ consists entirely of cardinality upper bounds, an action $a$, and two reachable beliefs $b$ and $b'$ such that $\forall b \in b'$, but $\forall b \in b'$. 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 minimum and maximum 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 $\text{image}(G) \leq k$. 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 $\Phi_G$, we denote the subset of binary clauses of $\Phi$. Second, if $L$ is a set of literals, $G$ is a predicate with arity 2, and $c$ is a constant, then we denote $L_{|G(c, d)} := \{d \mid G(c, d) \in L\}$. That is, $L_{|G(c, d)}$ selects from $L$ the values of attribute $G$ for $c$. Similarly, $L_{|\neg G(c, d)} := \{d \mid \neg G(c, d) \in L\}$. Third, say $b$ is a belief; we introduce a formal notation for the precise distribution, denoted $\text{D}_b$, of attribute value tuples. Our search states will explicitly keep track of that distribution, and hence contain sufficient information for precise belief update (this is not possible based only on $\bigcap b$, c.f. Proposition 3). $\text{D}_b$ maps any $G$ where $\text{image}(G) \leq k$ in $\Phi$, and any $c \in C$, onto a set of subsets of $C$. Namely, for each $m \in b$, $\text{D}_b(G, c)$ contains the set $\{d \mid m = G(c, d)\}$. Hence, for every $G$ and $c$, $\text{D}_b(G, c)$ specifies which combinations of attribute values occur.

Our search states $s$ are pairs $(L_s, \text{D}_s)$. 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 $\Phi$, 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

---

3 This relates to [14], who show that DL updates can often not be represented in terms of a single changed ABox.
procedure initialise()
(1) \(L_{\text{pre}} := \{l \mid l \text{ appears in pre}_{\text{eff}}\}\)
(2) \(L := \{ l \mid \text{eff}_P \land L_{\text{pre}} = l\}\)
(3) if ex. l.s.t. \(l \in L\) and \(l\) \in \(L\) then return (undefined)
(4) for all image(\(G\)) \leq k in \(\Phi\), \(c \in C\) do
(5) if \(\text{len}_{\text{G},c} > k\) then return (undefined)
(6) if \(\text{len}_{\text{G},c} = k\) then
\(L := L \cup \{-G(c,d) \mid d \in C, d \notin L_{\text{pre}}\}\)
\(D(G,c) := \{D \mid D \subseteq C, L_{\text{pre}} \subseteq D, \quad D \cap L_{\text{\neg G},c} = \emptyset, |D| \leq k\}\)
(8) return \((L, D)\)

Figure 2. The initialise procedure for exact search states.

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

procedure update(s, a)
(1) if \(\text{pre}_{\text{eff}} \not\subseteq L_s\), then return (undefined)
(2) \(L^A := \{ l \mid l \text{ appears in eff}_A\}\)
(3) \(L := \{ l \mid \text{eff}_P \land L^A = l\}\)
(4) if l.s.t. \(l \in L\) and \(l\) \in \(L\) then return (undefined)
(5) for all image(\(G\)) \leq k in \(\Phi\), \(c \in C\) do
(6) if \(\text{len}_{\text{G},c} > k\) then return (undefined)
(7) if \(\text{len}_{\text{G},c} = k\) then
\(L := L \cup \{-G(c,d) \mid d \in C, d \notin L_{\text{pre}}\}\)
(8) \(D(G,c) := \emptyset\)
(9) \(L^{\text{AT}} := L; L := L \cup \{ l \mid l \in L_s, \neg l \}\)
(10) for all image(\(G\)) \leq k in \(\Phi\), \(c \in C\), \(D \in D_s(G,c)\) do
(11) \(L := L \cup \{G(c,d) \mid G(c,d) \in L_s \land L^A\}\)
(12) \(D := L \cup \{D \mid D \subseteq C, \quad D \cap L_{\text{\neg G},c} = \emptyset\}\)
(13) \(D(G,c) := D(G,c) \cup \{D \cup L_{\text{\neg G},c} \mid |D'| = k - \text{len}_{\text{G},c}\}\)
(14) else \(D(G,c) := D(G,c) \cup \{D \cup L_{\text{\neg G},c} \mid |D'| = k - \text{len}_{\text{G},c}\}\)
(15) return \((L, D)\)

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 \(D\) structures. Line (9)
extends \(L\) with all literals from \(L_s\), except those that are contradicted
by \(L_s\). By Lemma 1, the resulting \(L\) is a superset of \(\bigcap \text{Res}(b,a)\).
By Lemma 3, as far as binary clauses are concerned, the resulting
\(L\) is equal to \(\bigcap \text{Res}(b,a)\). 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 \(D_s\) 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 \(D\) is updated correctly.
If an overflow occurs, then all possible ways of minimally repairing
the overflow are generated. If no overflow occurs, then \(D(G,c)\)
simply changes according to the effect and its implications. We have:

**Theorem 1** Assume a WSC task \((\Omega, W, C, U)\) where \(\Phi\) 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 \(D_b \equiv D_s\).

The formal proof of Theorem 1 is quite lengthy, and involves var-
ious (sometimes rather tedious) case distinctions. The proof essen-
tially 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 WSC task \((\Omega, W, C, U)\) with fixed arity and
fixed maximum cardinality, where \(\Phi\) is binary with consequence-
independent cardinality upper bounds. Assume \(b\) is reached by ac-
tion sequence \(\tilde{a}\). Then the corresponding search state \(s\) is computed
in time polynomial in the size of \((\Omega, W, C, U)\) and \(\tilde{a}\), 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-
independency, no interferences arise from the binary clauses.

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

5 Approximate Belief Update

Even though it seems likely that \(k\) will be small in practice, it is ad-
visable to look for more efficient methods. The size of \(D(G,c)\) is
bounded only by \(\frac{\text{|C|}^k}{k!}\). If there are many constants, then enum-
merating \(D\) 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 interleaved. 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
the latter; the former 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 = \text{the}
precise belief intersection - serves both as} L^+ \text{ 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 bi-
nary clauses. Line (4) tests for contradictions in the latter. Similarly,
line (6) aborts the algorithm in case of a conflict with a cardinal-
ity 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.
update(s, a)
(1) if pre_s \not\subseteq L_s then return (undefined)
(2) \(L^A := \{ l \mid l \) appears in eff. \}
(3) \(L := \{ l \mid \Phi_{tg} \land L^A \models l \}
(4) if \(l \in L^A \) then \(L := \{ l \mid l \in L^A \land \neg G(c, d) \}
(5) if \(\forall i \in L \) then return (undefined)
(6) \(L^A_{[G,c]} := k \) then \(L := \{ l \mid l \in L^A \land \neg G(c, d) \}
(7) \(L := \{ l \mid l \in L^A \land \neg G(c, d) \}
(8) L^- := L \cup \{ l \mid l \in L^+ \land l \not\in L \}
(9) L^+ := \{ l \mid l \in L^+ \land l \not\in L \}
(10) if \(\forall i \in L \) then return (undefined)
(11) \(L^A_{[G,c]} > k \) then \(L^- := L^+ \setminus \{ G(c, d) \mid G(c, d) \in L^+ \} \)
(12) \(L^- := L^+ \setminus \{ G(c, d) \mid G(c, d) \in L^+ \} \)
(13) return \(L^- \cup L^+ \).

Figure 4. The update procedure for approximate search states.

Theorem 2 Assume a WSC task \((\Omega, W, C, U)\) where \(\Phi\) 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 \(L^+ \subseteq b \subseteq L^+ \).

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

Corollary 3 Assume a WSC task \((\Omega, W, C, U)\) with fixed arity, where \(\Phi\) is binary with consequence-independent cardinality upper bounds. Assume \(b\) is reached by action sequence \(a\). If the corresponding approximate search state \(s\) is defined, then \(s\) is computed in time polynomial in the size of \((\Omega, W, C, U)\) and \(a\), and \(\cap b \subseteq L^+ \).

Example 3 Re-consider Example 1. Running initialise, we get the state where \(L^- = L^+ = \{ \text{ticketfor}(t, \text{Peter}), \text{ticketfor}(t, \text{Bob}) \}\). Applying \(a_1\), both lines (11) and (12) fire and so we get \(s_1 = \{ \text{update}(a_0, a_1) \}\) where \(L^- = L^+ = \{ \text{ticketfor}(t, \text{Peter}) \}\). Applying \(a_2\), only line (12) fires and so we get \(L^- = \{ \text{ticketfor}(t, \text{Peter}) \}\) and \(L^+ = \{ \text{ticketfor}(t, \text{Mary}), \text{ticketfor}(t, \text{Peter}) \}\).

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-basedWSC, 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