Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty

Jörg Hoffmann*
DERI Innsbruck
Innsbruck, Austria
joerg.hoffmann@deri.org

Piergiorgio Bertoli and **Marco Pistore**
ITC-IRST
Trento, Italy
[bertoli,pistore]@irst.itc.it

Abstract

Thanks to recent advances, AI Planning has become the underlying technique for several applications. Amongst these, a prominent one is automated Web Service Composition (WSC). One important issue in this context has been hardly addressed so far: WSC requires dealing with background ontologies. The support for those is severely limited in current planning tools. We introduce a planning formalism that faithfully represents WSC. We show that, unsurprisingly, planning in such a formalism is very hard. We then identify an interesting special case that covers many relevant WSC scenarios, and where the semantics are simpler and easier to deal with. This opens the way to the development of effective support tools for WSC. Furthermore, we show that if one additionally limits the amount and form of outputs that can be generated, then the set of possible states becomes static, and can be modelled in terms of a standard notion of initial state uncertainty. For this, effective tools exist; these can realize scalable WSC with powerful background ontologies. In an initial experiment, we show how scaling WSC instances are comfortably solved by a tool incorporating modern planning heuristics.

Introduction

Since the mid 90s, AI Planning tools have become many times more scalable, through the invention of heuristic functions and other search techniques, e.g., (Hoffmann & Nebel 2001). A highly relevant application area for planning is automated composition (WSC) of semantic web services (SWS). SWS are pieces of software advertised with a formal description of what they do; *composing* services means to link them together in a way satisfying a complex user requirement. WSC is widely recognized for its huge economic potential. In the wide-spread OWL-S (Coalition 2003) and WSMO (Fensel *et al.* 2006) frameworks, SWS are described akin to planning operators, with preconditions and effects. Hence planning is a prime candidate for realizing WSC.

WSC raises many novel challenges; for example, SWS may exhibit complex interaction interfaces, as addressed, e.g., in (Pistore, Traverso, & Bertoli 2005). Herein, we focus on one important issue that has been hardly addressed so far: semantic web services are embedded in background ontologies, which *constrain the behavior of the involved entities*. To deal with this, WSC must reason about which behaviors are possible. Further, the semantic descriptions may contain only partial information (e.g., a user does not specify every tiny detail about her request), so WSC must deal with uncertainty. These aspects of WSC are best characterized in terms of *planning with background theories*, along the lines of, e.g., (Eiter *et al.* 2003; Giunchiglia *et al.* 2004). We consider the case where there is no observability, i.e., conformant planning. Note here that "observing" during SWS execution may involve requesting additional information from the user, or even crossing enterprise boundaries. So observability is partial at best; extending our notions to conditional planning should be straightforward, and is future work.[1]

Example 1 *We need a web service that helps organizing ceremony lunches by finding a consistent menue and venue. A "lunch-expert" service inputs a description of the ceremony, and outputs the kind of lunch required, e.g., "good" or "top". The lunch must obey complex constraints; e.g., for a "good" lunch, "average" wine can only be tolerated if either the food is "great" or the venue is "scenic",* $\neg good \vee \neg averagewine \vee greatfood \vee scenic$*. A set of "directory" services finds different quality providers. A solution consists of the lunch-expert service and a set of directory services that covers the relevant cases. Note that, even if we observe the outcome of the lunch-expert, this does not identify a unique combination of food, drinks, and venue quality.*

Example 1 illustrates that reasoning over the background ontology is necessary to understand which services should be used – which directory services deal with relevant cases? – and to test whether a given composition is a solution – how many directory services do we need? Note that the example requires so-called *partial matches*, where a service may deal with only part of a requirement: any directory service handles only some particular cases. Partial matches are not supported by most current tools; we get back to this below.

Incorporating background theories into the modern scalable planning tools poses serious challenges, and has not yet even been tried. The background theory incurs a ramification problem, making even the basic step of computing a state transition – which is a form of belief revision – a

*Research supported by EU project SUPER, IST FP6-026850.

[1] (Agarwal *et al.* 2005) include a conditional planning approach to WSC, but do not incorporate background theories.

computationally very hard task. The existing approaches either severely limit the background theories, necessitating a distinction between "basic" and "derived" predicates (Thiébaux, Hoffmann, & Nebel 2005); or are based on general deduction and lacking the planning-specific heuristic techniques, e.g., (Eiter *et al.* 2003; Giunchiglia *et al.* 2004).

Most existing planning tools dealing with WSC, e.g., (Ponnekanti & Fox 2002; Pistore, Traverso, & Bertoli 2005), ignore the background theories and assume *exact matches* of inputs and outputs, based on concept names. In Example 1, this does not work because the lunch-expert output refers to whole lunches, while the directory inputs refer to food, drinks, and/or locations. There are some approaches that allow *plug-in* matches, e.g., (Sirin & Parsia 2004), where a single service must cover all possible cases. In Example 1, this would require a single directory service for all relevant combinations of quality requirements.

We introduce a planning formalism that faithfully represents WSC. In particular, we allow the generation of new constants, corresponding to web service outputs.[2] We show that it is Π_2^p-complete to test whether a given action sequence is a plan. We then make two key contributions:

1. We identify an interesting special case of WSC, termed *forward effects*, where the semantics are much easier to deal with, but which still covers many relevant WSC scenarios. The key observation is that the effects of a web service are often "forward" in the sense that all ramifications concern only propositions involving at least one new constant; in that case, a "backwards-directed" belief revision is not necessary. In WSC with forward effects, plan testing is "only" **coNP**-complete. Planning under uncertainty has the same complexity of plan testing, and scalable planning tools for this case have already been developed. Hence this result opens up a promising line of research extending existing planning tools for WSC.

2. Given forward effects, we identify a restriction on the amount and form of outputs that can be generated, under which the set of possible states becomes static, in a certain sense. It is then no longer necessary to reason about the theory every time an action is applied; one can instead model the possible states in the form of an initial state formula. The restriction is naturally given under *strictly forward effects*, where the web service effects concern *only* new constants. This is still significant, and corresponds to web services modelled as in, e.g., the basic versions of OWL-S. We provide a compilation into conformant planning under initial state uncertainty, and hence enable the application of off-the-shelf modern planning tools. We show empirically that this approach is promising.

We next introduce our formalism, then we define forward effects and our compilation into initial state uncertainty; thereafter we present empirical results, and conclude. Due to space restrictions, some details are moved to (Hoffmann, Bertoli, & Pistore 2007).

[2]E.g., OWL-S outputs model the generated data.

Formalizing WSC

Our formalism, denoted with $\mathcal{WSC}$, follows the relevant literature, e.g., (Winslett 1988; Eiter *et al.* 2003). It relies on a notion of *clausal* theories, which supports a powerful notion of WSC, and at the same time enables our later compilation into initial state uncertainty. Web services correspond to planning operators. Their input/output behavior maps to input/output parameters, on which preconditions and effects are specified. This corresponds to web services in OWL-S "service profiles" and WSMO "capabilities".

We assume a supply of logical predicates, a supply of variable names, and an infinite supply of constant names; we will denote predicates with p, q, variables with x, y, and constants with a, b, c, d, e. *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 has the variable arguments 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 respective substitution, abbreviated as $l[C]$. In the same way, we use the substitution notation for any construct involving variables. Slightly abusing notation, we use a vector of constants, like C above, also to denote the set of constants appearing in it. We refer to positive ground literals as *propositions*.

A *clause* is a disjunction of literals with universal quantification on the outside, e.g., $\forall x.(\neg p(x) \vee q(x))$; a *theory* is a conjunction of clauses. An *operator* o is a tuple $(X_o, \text{pre}_o, Y_o, \text{eff}_o)$, where X_o, Y_o are sets of variables, pre_o is a conjunction of literals from $\mathcal{L}^{X_o}$, and eff_o is a conjunction of literals from $\mathcal{L}^{X_o \cup Y_o}$. The intended meaning is that X_o are the inputs and Y_o the outputs, i.e., the new constants created by the operator. For an operator o, an *action* a is given by $(\text{pre}_a, \text{eff}_a) \equiv (\text{pre}_o, \text{eff}_o)[C_a/X_o, E_a/Y_o]$ where C_a and E_a are vectors of constants; for E_a we require that the constants are pairwise different – it makes no sense to "output the same new constant twice".

Planning tasks are tuples $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. Here, $\mathcal{P}$ is a set of predicates; $\mathcal{T}$ is the background theory; $\mathcal{O}$ is a set of operators; C_0 is a set of constants, the initial constants supply; ϕ_0 is a conjunction of ground literals, describing the possible initial states; ϕ_G is a conjunction of literals with existential quantification on the outside, describing the goal states, e.g., $\exists x, y.(p(x) \wedge q(y))$.[3] All predicates are taken from $\mathcal{P}$, and all constants are taken from C_0. All constructs (e.g., sets and conjunctions) are assumed to be finite.

In what follows, assume we are given a planning task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. To be able to model the creation of constants, *states* in our formalism are enriched with the set of constants that exist in them: a state s is a pair (C_s, I_s) where C_s is a set of constants, and I_s is a C_s*-interpretation*, i.e., an interpretation of the predicates $\mathcal{P}$ over the constants C_s. In other words, I_s is a truth value assignment to all propositions formed from $\mathcal{P}$ and C_s. We write $s \models \phi$ for $I_s \models \phi$, where the quantifiers in ϕ are restricted to C_s.

[3]The existential quantification is needed to give meaning to the creation of new constants.

We next define the outcome of applying actions in states. Given a state s and an action a, a is *applicable in* s, short $appl(s,a)$, if $s \models \text{pre}_a$, $C_a \subseteq C_s$, and $E_a \cap C_s = \emptyset$. That is, on top of the usual precondition satisfaction we require that a's inputs exist and that a's outputs do not yet exist. We allow *parallel actions*. These are sets of actions which are applied at the same point in time; we require that $E_a \cap E_{a'} = \emptyset$ for all $a, a' \in A$. The result of applying a parallel action A in a state s is $res(s,A) :=$

$$\{(C', I') \mid C' = C_s \cup \textstyle\bigcup_{a \in A, appl(s,a)} E_a, \\ I' \in min(s, C', \mathcal{T} \wedge \textstyle\bigwedge_{a \in A, appl(s,a)} \text{eff}_a)\}$$

Here, $min(s, C', \phi)$ is the set of all C'-interpretations that satisfy ϕ and that are minimal with respect to the partial order defined by $I_1 \leq I_2$:iff for all propositions p over C_s, if $I_2(p) = I_s(p)$ then $I_1(p) = I_s(p)$. This is a standard semantics where the ramification problem is addressed by requiring minimal changes to the predecessor state s.

Note that, in the definition of $res(s,A)$, non-applicable actions are allowed. This realizes partial matches: a web service may cover only part of the possible situations. In planning terms, our actions have a *conditional effects semantics*; we do not treat those explicitly here, for the sole purpose of simplifying notation. A parallel action A is *inconsistent* with a state s if $res(s,A) = \emptyset$. This happens in case of conflicts between the subset of actions that are applicable in s; the conflicts can be either direct, or indirect via causing a contradiction in the background theory.

We refer to the set of states possible at a given time as a *belief*. The *initial belief* is $b_0 := \{s \mid C_s = C_0, s \models \mathcal{T} \wedge \phi_0\}$. A parallel action A is inconsistent with a belief b if it is inconsistent with at least one $s \in b$. In the latter case, $res(b,A)$ is undefined; else, it is $\bigcup_{s \in b} res(s,A)$. This is extended to action sequences in the obvious way. A *plan* is a sequence $\langle A_1, \ldots, A_n \rangle$ s.t. for all $s \in res(b_0, \langle A_1, \ldots, A_n \rangle) : s \models \phi_G$.

Example 2 *A part of Example 1 is formalized as follows:*

- $\mathcal{P}$ *contains* {*lunch*, *lunchquality*, *average*, *good*, *top*, *averagewine*, *greatfood*, *scenic*}
- $\mathcal{T}$ *contains* $\forall x.(\neg$*lunchquality*$(x) \vee$ *average*$(x) \vee$ *good*$(x) \vee$ *top*$(x))$ *and* $\forall x.(\neg$*good*$(x) \vee \neg$*averagewine*$(x) \vee$ *greatfood*$(x) \vee$ *scenic*$(x))$
- $\mathcal{O}$ *contains* $ws_{\textit{lunch-expert}} = (\{x\},$ *lunch*$(x), \{y\},$ *lunchquality*$(y))$
- $C_0 = \{$*mylunch*$\}$; $\phi_0 =$ *lunch*$($*mylunch*$)$

A comprehensive investigation of the computational complexity of our formalism is a topic for future work. However, the following result points out that the complicated semantics of action application makes it computationally very hard to even *test plans*. Obviously, this must be done as part of planning. When assuming *fixed arity* – a constant upper bound on the arity of all variable vectors (e.g., used in predicates) – transformation to a propositional representation is polynomial. Even in this case, plan testing is Π_2^p-complete in $\mathcal{WSC}$. Note that the same proof applies when allowing conditional effects instead of parallel actions.

Proposition 1 (Plan testing in $\mathcal{WSC}$) *Assume a $\mathcal{WSC}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$ with fixed arity, and a sequence $\langle A_1, \ldots, A_n \rangle$ of parallel actions. It is Π_2^p-complete to decide whether $\langle A_1, \ldots, A_n \rangle$ is a plan.*

Proof Sketch: Membership: By guess and check, it can be tested in **NP** whether $s' \in res(s,A)$ for arbitrary s, s', and A. Hence one can guess the proposition values along $\langle A_1, \ldots, A_n \rangle$, and test validity using calls to an **NP** oracle.

Hardness: Truth of a QBF formula $\forall X \exists Y \phi[X,Y]$ is reduced to plan testing for a single parallel action A. We introduce a new proposition G, which is set to F initially. The theory is formed by a version of ϕ where the literal $\neg G$ is introduced into every clause. This means that, initially, ϕ trivializes to T. One action in A has an empty (true) precondition, and sets G to T. This ensures that ϕ must hold after A. Further, for each $x \in X$, A includes one action that has precondition x and effect x, as well one action that has precondition $\neg x$ and effect $\neg x$. This ensures that the truth value assignment to X is the same before and after A. The goal is G. The initial belief contains all truth assignments to $X \cup Y$. Since A unconditionally achieves the goal, A is a plan iff it is consistent with all these states. A is consistent with a state s iff there exists a state s' that agrees with s on X, and that satisfies ϕ. ∎

Forward Effects

The high complexity of planning in $\mathcal{WSC}$ motivates the search for interesting special cases. We define a special case where every change an action makes to the state involves a newly generated constant. Formally, assume a $\mathcal{WSC}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. The task has *forward effects* iff:

- For all $o \in \mathcal{O}$, and for all $l[X] \in \text{eff}_o$, we have $X \cap Y_o \neq \emptyset$. In words, the variables of every effect literal contain at least one output variable.
- For all clauses $cl[X] \in \mathcal{T}$, where $cl[X] = \forall X.(l_1[X_1] \vee \cdots \vee l_n[X_n])$, we have $X = X_1 = \cdots = X_n$. In words, in every clause all literals share the same arguments.

We denote this case with $\mathcal{WSC}|_{fwd}$. The first condition means that every ground effect literal of an action contains at least one new constant. The second is a sufficient condition implying that effects involving new constants can only affect literals involving new constants. Intuitively, $\mathcal{WSC}|_{fwd}$ covers the case where a web service outputs some new constants, sets their basic properties relative to the inputs, and relies on the background ontology to describe the ramifications. In Example 2, $ws_{\textit{lunch-expert}}$ outputs a constant of concept *lunchquality*, and lets the ontology formulate the implications of that; $ws_{\textit{lunch-expert}}$ does not in any way affect the status of the constant *mylunch* that was present beforehand. Many WSC scenarios, both from the literature and from real case studies, have forward effects. A simple example is the wide-spread "virtual travel agency", where web services must be linked that book travel and accommodation, generating new constants corresponding to tickets and reservations.

We now observe how the semantics in $\mathcal{WSC}|_{fwd}$ is much simpler than in general $\mathcal{WSC}$, no longer needing the notion

of minimal changes with respect to the previous state. Given a state s and a parallel action A, define $res|_{fwd}(s, A) :=$

$$\{(C', I') \mid C' = C_s \cup \bigcup_{a \in A, appl(s,a)} E_a, \\ I'|_{C_s} = I_s, I' \models \mathcal{T} \wedge \bigwedge_{a \in A, appl(s,a)} \text{eff}_a\}$$

where $I'|_{C_s}$ is the restriction of I' to propositions over C_s.

Proposition 2 (Semantics of $\mathcal{WSC}|_{fwd}$) *Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$, a state s, and a parallel action A. Then $res(s, A) = res|_{fwd}(s, A)$.*

Proof: Recall that $res(s, A)$ is defined as the set of states s' that satisfy $\mathcal{T} \wedge \bigwedge_{a \in A, appl(s,a)} \text{eff}_a$ and whose difference to I_s is minimal on the propositions over C_s. The claim follows because the latter set of propositions has an empty intersection with the set of propositions possibly affected by the action and its ramifications. ∎

Further, in difference to $\mathcal{WSC}$, we can make sure in a pre-process that no inconsistent actions will occur, by filtering out certain actions, without affecting plan existence. An action a is called *contradictory* if $\mathcal{T} \wedge \text{eff}_a$ is unsatisfiable.

Proposition 3 (Inconsistency in $\mathcal{WSC}|_{fwd}$) *Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$, a belief b reachable in the task, and a parallel action A. If A is inconsistent with b, then there exists $a \in A$ so that a is contradictory.*

Proof: By definition, there exists $s \in b$ so that A is inconsistent with s. Now, consider that the outputs of parallel actions are disjoint, and that any action in $\mathcal{WSC}|_{fwd}$ affects only propositions involving at least one of its output constants. Hence there can be no conflicts between different actions in A, hence there must exist a single $a \in A$ so that a is inconsistent with s. The only way a single action can be inconsistent, with any state, is if $\mathcal{T} \wedge \text{eff}_a$ is unsatisfiable. ∎

Obviously, a contradictory action will never yield a successor state; it can be filtered out prior to planning, without affecting plan existence. Thanks to this, and thanks to the simpler semantics as per Proposition 2, plan testing is much easier in $\mathcal{WSC}|_{fwd}$ than in $\mathcal{WSC}$.

Proposition 4 (Plan testing in $\mathcal{WSC}|_{fwd}$) *Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$ with fixed arity, and without contradictory actions. Assume a sequence $\langle A_1, \ldots, A_n \rangle$ of parallel actions. It is* **coNP***-complete to decide whether $\langle A_1, \ldots, A_n \rangle$ is a plan.*

Proof: Hardness is obvious, considering an empty sequence. Membership can be shown by the following guess-and-check argument. Say C is the union of C_0 and all output constants appearing in $\langle A_1, \ldots, A_n \rangle$. We guess an interpretation I of all propositions over $\mathcal{P}$ and C; further, for each $1 \leq t \leq n$, we guess a set C_t of constants. We can then check in polynomial time whether I and the C_t correspond to an execution of $\langle A_1, \ldots, A_n \rangle$. For $1 \leq t \leq n$ and $a \in A_t$, say that a is applicable if $I \models \text{pre}_a$, $C_a \subseteq C_t$, and $E_a \cap C_t = \emptyset$. First, we assert $I \models \mathcal{T}$. Second, for all t and for all $a \in A_t$, assert that, if a is applicable, then $I \models \text{eff}_a$. Third, assert that $C_{t+1} = C_t \cup \{E_a \mid a \in A_t, a \text{ is applicable}\}$. It is easy to see that I and the C_t correspond to an execution iff all three assertions hold; note that I needs not be time-stamped because once an action has generated its outputs then the properties of the respective propositions remain fixed forever. The claim follows because, with fixed arity, we can also test in polynomial time whether I and C_n satisfy ϕ_G. A guess of I and C_t is successful if it corresponds to an execution and does not satisfy ϕ_G. Obviously, $\langle A_1, \ldots, A_n \rangle$ is a plan iff there is no such guess of I and C_t. ∎

In standard notions of planning under uncertainty, plan testing has the same complexity. Research has already resulted in a sizeable number of approaches and scalable tools, e.g., (Cimatti, Roveri, & Bertoli 2004; Hoffmann & Brafman 2006; Bryce, Kambhampati, & Smith 2006). It seems likely that the underlying techniques can be useful also for $\mathcal{WSC}|_{fwd}$. In particular, Conformant-FF (CFF) (Hoffmann & Brafman 2006) is a promising candidate; CFF is based on CNF reasoning, which can be naturally adapted.

Compilation to Initial State Uncertainty

We now show that, under certain additional restrictions, off-the-shelf scalable tools for planning under uncertainty can solve $\mathcal{WSC}|_{fwd}$. The main limiting factors are: (1) Those tools do not allow the generation of new constants. (2) Those tools allow the specification of a clausal formula only for the initial state, not for all states. We adopt the obvious approach to deal with (1), by considering a pre-fixed set of constants, namely the initially available constants, and "potential" constants that can be used to instantiate outputs. Our more subtle observation is that, within $\mathcal{WSC}|_{fwd}$, one can also deal with (2). In what follows, we first introduce our core observation of a case where the set of possible states becomes "static", in a certain sense; we then exploit that observation by providing a compilation into planning under uncertainty.

Our core observation is based on a notion of *compatible actions*. Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. Two actions a, a' are compatible if either $E_a \cap E_{a'} = \emptyset$, or $\text{eff}_a = \text{eff}_{a'}$. That is, a and a' either have disjunct outputs – and hence affect disjunct sets of literals since we are in $\mathcal{WSC}|_{fwd}$ – or their effects agree completely. A set A of actions is compatible if $E_a \cap C_0 = \emptyset$ for all $a \in A$, and every pair of actions in A is compatible.

Proposition 5 (Static states in $\mathcal{WSC}|_{fwd}$) *Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$, a compatible set of actions $\mathcal{A}$, and a state s that can be reached with $\mathcal{A}$. Then $s \models \phi_0$, and whenever $E_a \subseteq C_s$ for $a \in \mathcal{A}$, $s \models \text{eff}_a$.*

Proof: The proof is by induction. In the base case, for $s \in b_0$, the claim holds by definition since $C_s \cap E_a = \emptyset$ for all a. Say s' is reached from s by a parallel action A. For simplicity, consider the case where $A = \{a\}$; the general case follows immediately from that. If a is not applicable to s, with induction assumption there is nothing to prove. Else, because we are in $\mathcal{WSC}|_{fwd}$, $res(s, a) = \{(C', I') \mid C' = C_s \cup E_a, I'|_{C_s} = I_s, s \models \mathcal{T} \wedge \text{eff}_a\}$. With induction assumption applied to s, we have $res(s, a) = \{(C', I') \mid C' = C_s \cup E_a, s \models \mathcal{T} \wedge \phi_0 \wedge \bigwedge_{a' \in \mathcal{A}, E_{a'} \subseteq C_s} \text{eff}_{a'} \wedge \text{eff}_a\}$.

Now, if any $a' \in \mathcal{A}$ has $E_{a'} \subseteq C_s \cup E_a$ but $E_{a'} \not\subseteq C_s$, then we have $E_{a'} \cap E_a \neq \emptyset$ and hence $\text{eff}_{a'} = \text{eff}_a$. ■

With Proposition 5, the possible configurations of all (potential) constants are characterized by the formula $\mathcal{T} \wedge \phi_0 \wedge \bigwedge_{a \in \mathcal{A}} \text{eff}_a$. Before even beginning to plan, we already know how the potentially generated constants will behave. So we can list the possible behaviors of all potential constants in our initial belief, and let the actions change only which constants actually exist. In other words, we can compile into initial state uncertainty, given we settle for a finite set $\mathcal{A}$ of compatible actions. One option is to simply require every action to have its own unique output constants. This appears problematic: every potential constant must be allowed to instantiate the input parameters of every operator, hence necessitating the creation of a new action and yet new output constants. It is unclear where to break this loop, in a sensible way. We instead focus on a restriction of $\mathcal{WSC}|_{fwd}$, *strictly forward effects* (denoted $\mathcal{WSC}|_{sfwd}$), where it suffices to assign unique output constants to individual *operators*, rather than actions. Assume a $\mathcal{WSC}|_{fwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. The task is in $\mathcal{WSC}|_{sfwd}$ iff, for all $o \in \mathcal{O}$, and for all $l[X] \in \text{eff}_o$, $X \subseteq Y_o$.

In $\mathcal{WSC}|_{sfwd}$, all actions based on the same operator have the same effect. Hence the action set is compatible if we choose one or several sets of unique output constants for every subset of operators that has identical effects. Importantly, $\mathcal{WSC}|_{sfwd}$ closely corresponds to web services modelled as in, e.g., (Ponnekanti & Fox 2002) and the basic versions of OWL-S, where inputs and outputs are regarded as independent entities. So the existence of a compilation into standard planning under uncertainty is quite significant.

We compile a $\mathcal{WSC}|_{sfwd}$ task into a task of conformant planning under initial state uncertainty, which takes the form $(\mathcal{P}, \mathcal{A}, \phi_0, \phi_G)$. $\mathcal{P}$ is the finite set of propositions used. $\mathcal{A}$ is a finite set of actions, where each $a \in \mathcal{A}$ takes the form $(\text{pre}(a), \text{eff}(a))$ of a pair of sets of literals over $\mathcal{P}$. ϕ_0 is a CNF formula over $\mathcal{P}$, ϕ_G is a conjunction of literals over $\mathcal{P}$. These notions are given a standard belief state semantics. A state is a truth value assignment to $\mathcal{P}$. The initial belief is the set of states satisfying ϕ_0. The result of executing an action a in a state s is $res(s,a) := s$ if $s \not\models \text{pre}(a)$, and otherwise $res(s,a) := (s \cup add(a)) \setminus del(a)$; here we use the standard notation that gives s in terms of the set of propositions that it sets to T, uses $add(a)$ to denote the positive literals in $\text{eff}(a)$, and $del(a)$ to denote the negative literals in $\text{eff}(a)$. Extension to parallel actions is done by taking the set unions of the positive and negative effect literals. Extension of res to beliefs and the definition of a plan remain unchanged.

Starting with a $\mathcal{WSC}|_{sfwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$, the compilation produces a conformant planning task $(\mathcal{P}', \mathcal{A}', \phi_0', \phi_G')$, as follows:

- For each operator $o \in \mathcal{O}$, create a unique set of new constants $E_o = \{e_1, \ldots, e_k\}$ where $Y_o = \{y_1, \ldots, y_k\}$. We denote $C := C_0 \cup \bigcup_{o \in \mathcal{O}} E_o$.
- $\mathcal{P}'$ contains all instantiations, with C, of $\mathcal{P}$ plus two new predicates, Ex and G. Ex has arity 1 and expresses which constants have yet been brought into existence. G has arity 0 and forms the new goal, i.e., $\phi_G' = G$.
- The actions $\mathcal{A}'$ are the instantiations of all $o \in \mathcal{O}$, where X_o is instantiated with C, and Y_o is instantiated with E_o. The preconditions are enriched with $(\bigwedge_{x \in X_o} Ex(x)) \wedge (\bigwedge_{e \in E_o} \neg Ex(e))$, the effects are replaced by $\bigwedge_{e \in E_o} Ex(e)$.
- The original action effects, i.e., the conjunction of $\text{eff}_o[E_o/Y_o]$ for all operators $o \in \mathcal{O}$, is moved into ϕ_0'. Further, ϕ_0' contains ϕ_0, $\mathcal{T}$ instantiated with C, and $(\bigwedge_{c \in C_0} Ex(c)) \wedge \bigwedge_{c \in C \setminus C_0} \neg Ex(c)) \wedge \neg G$.
- $\mathcal{A}'$ is enriched with goal achievement actions, achieving G under preconditions instantiating ϕ_G with C.

Note that we create only one E_o per operator, and that we do not take into account sets of operators that have identical effects. This is simplifies presentation; the results carry over immediately to more general output creation strategies.

Example 3 *Re-consider the task fragment defined in Example 2. We have $C = \{$mylunch, mylq$\}$ where mylq is the output generated for $ws_{\textit{lunch-expert}}$.*

- $\mathcal{P} = \{$*lunch*, *lunchquality*, *average*, *good*, *top*, *averagewine*, *greatfood*, *scenic*, $Ex, G\}$
- $\mathcal{A}'$ *has all instantiations of* $ws_{\textit{lunch-expert}}$ $[$*mylq*$/y] = (\{x\},$ *lunch*$(x) \wedge Ex(x) \wedge \neg Ex($*mylq*$), Ex($*mylq*$))$
- ϕ_0' *contains* *lunch*(*mylunch*), *lunchquality*(*mylq*) *(initial state, action effects); all instantiations of* $\forall x.(\neg$*lunchquality*$(x) \vee$ *average*$(x) \vee$ *good*$(x) \vee$ *top*$(x))$ *and* $\forall x.(\neg$*good*$(x) \vee \neg$*averagewine*$(x) \vee$ *greatfood*$(x) \vee$ *scenic*$(x))$ *(theory);* $Ex($*mylunch*$)$, $\neg Ex($*mylq*$)$, $\neg G$ *(constants and goal).*

We say that an operator is contradictory if $\mathcal{T} \wedge \text{eff}_o[E_o/Y_o]$ is unsatisfiable; note that, in $\mathcal{WSC}|_{sfwd}$, an operator is contradictory iff all actions based on it are contradictory.

Proposition 6 (Compilation Soundness) *Assume a $\mathcal{WSC}|_{sfwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$ without contradictory operators. Let $\langle A_1, \ldots, A_n \rangle$ be a plan for the compiled task $(\mathcal{P}', \mathcal{A}', \phi_0', \phi_G')$. Then the sub-sequence of non-goal achievement actions in $\langle A_1, \ldots, A_n \rangle$ is a plan for $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$.*

Proof Sketch: For an arbitrary sequence of non-goal achievement actions, denote by b the belief after execution in the original task, and by $\bar{b}$ the belief after execution in the compiled task. For a state s in the original task, denote by $[s]$ the class of all compiled-task states $\bar{s}$ over the constants $C_0 \cup \bigcup_{o \in \mathcal{O}} E_o$ so that $\{c \mid \bar{s}(Ex(c)) = T\} = C_s$, $\bar{s}|_{C_s} = I_s$, and $\bar{s} \models \mathcal{T} \wedge \phi_0 \wedge \bigwedge_{o \in \mathcal{O}} \text{eff}_o[E_o]$. One can prove that $\bar{b} = \bigcup_{s \in b} [s]$. The claim follows directly from that. ■

Proposition 7 (Compilation Completeness) *Assume a $\mathcal{WSC}|_{sfwd}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$ without contradictory operators. Let $\langle A_1, \ldots, A_n \rangle$ be a plan where every operator o appears with at most one instantiation E_o of the outputs. Then $\langle A_1, \ldots, A_n \rangle$ can be extended with goal achievement actions to form a plan for the compiled task $(\mathcal{P}', \mathcal{A}', \phi_0', \phi_G')$ obtained using the outputs E_o.*

Proof: Follows immediately from $\overline{b} = \bigcup_{s \in b}[s]$ as shown for Proposition 6. ∎

As indicated, the proofs of Propositions 6 and 7 remain valid when allowing more than one E_o per operator, and/or when operators with identical effects share output constants. Note that operators have identical effects if several web services provide alternative ways of achieving something, which is quite a usual situation. In the experiments below, all such operators are assigned the same output constants.

A First Experiment

We present results for a scalable WSC scenario, modelled in $\mathcal{WSC}|_{sfwd}$. We implemented the compilation explained above, and ran tests with CFF. Our WSC scenario can be instantiated to study different scalability aspects. The scenario is abstract, but representative for, e.g., telecommunication scenarios where one complex function has to be obtained by concatenating several existing functions, and where several "parallel" services must be established that serve different cases. Concretely, the scenario demands to realize a kind of chain. There are n concepts $A_1, \ldots, A_n$. The goal input is A_1, the goal output is A_n. Beneath each A_i, there is a tree-shaped hierarchy of sub-concepts, with branching b and depth d. For each leaf A_{ij}, there is a web service that takes A_{ij} as input and that outputs A_{i+1}. Hence, a solution must for each A_i put together all services treating the leaves, thereby making sure to obtain a constant of concept A_{i+1}. Sequencing these steps yields the solution.

In our tests, we considered two extreme cases of tree shapes, giving us instances with identical numbers of leaves. Namely, we used the scenario **Broad**, where $d = 1$ and b scales over $2, 4, 8, 16, 32$; and **Deep**, where $b = 2$ and d scales over $1, 2, 3, 4, 5$. In both scenarios, n scaled from 2 to 20. As it turns out, both scenarios yield very similar results: CFF's search spaces are *identical*. **Deep** takes up to an order of magnitude more runtime because the more complex concept hierarchies incur an overhead in CFF's internal SAT testing. Figure 1 shows the results for **Broad**. With up to 8 leaves, CFF easily scales to chains of length 20. Even with 32 leaves, it solves chains of at least length 7.

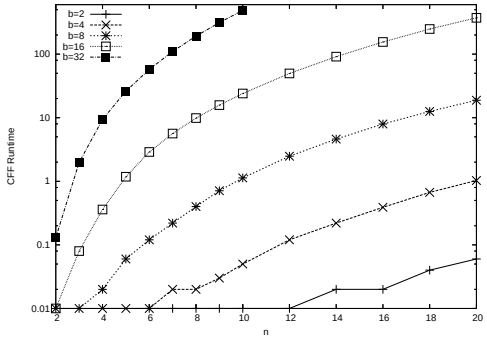

Figure 1: Results for **Broad**.

We also ran the aforementioned deduction-based approach (Eiter *et al.* 2003). As we expected, the lack of heuristic guidance results in much worse performance. With **Broad** and $n = 2$, up to 18 leaves can be solved within 30 minutes; however, with $n = 3$ only 2 leaves are feasible, and no instance is solved for $n > 3$. In **Deep**, only 8 leaves can be solved with $n = 2$. We created a **Broad-Trap** scenario, where a second chain of concepts can be linked, but is completely irrelevant for the goal service. CFF is largely unaffected by this, since the heuristic function correctly excludes the irrelevant parts. The deduction-based approach, by contrast, can solve only up to 14 leaves with $n = 2$, and no longer solves any instance with $n > 2$.

Conclusion and Future Work

We have identified a special case of planning with background theories that is relevant in the WSC context; we have established a new connection to initial state uncertainty and have performed a first experiment. Exploring these results further has the potential to yield powerful and scalable tools for WSC within the near future.

References

Agarwal, V.; Chafle, G.; Dasgupta, K.; Karnik, N.; Kumar, A.; Mittal, S.; and Srivastava, B. 2005. Synthy: A system for end to end composition of web services. *JWS* 3(4).

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Planning graph heuristics for belief space search. *JAIR* 26:35–99.

Cimatti, A.; Roveri, M.; and Bertoli, P. 2004. Conformant planning via symbolic model checking and heuristic search. *AI* 159(1–2):127–206.

Coalition, T. O. S. 2003. OWL-S: Semantic Markup for Web Services.

Eiter, T.; Faber, W.; Leone, N.; Pfeifer, G.; and Polleres, A. 2003. A logic programming approach to knowledge-state planning, II: The DLVK system. *AI* 144(1-2):157–211.

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

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *AI* 153(1-2):49–104.

Hoffmann, J., and Brafman, R. 2006. Conformant planning via heuristic forward search: A new approach. *AI* 170(6–7):507–541.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J.; Bertoli, P.; and Pistore, M. 2007. Web service composition as planning, revisited: In between background theories and initial state uncertainty. Technical report, DERI Innsbruck. Available at http://members.deri.at/ joergh/papers/tr-aaai07.ps.gz.

Pistore, M.; Traverso, P.; and Bertoli, P. 2005. Automated composition of web services by planning in asynchronous domains. In *Proc. ICAPS-05*.

Ponnekanti, S., and Fox, A. 2002. SWORD: A developer toolkit for web services composition. In *Proc. WWW-02*.

Sirin, E., and Parsia, B. 2004. Planning for semantic web services. In *Workshop "Semantic Web Services" at ISWC-04*.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *AI* 168(1–2):38–69.

Winslett, M. 1988. Reasoning about actions using a possible models approach. In *Proc. AAAI'88*.