Scalable Web Service Composition with Partial Matches

Adina Sirbu and Jörg Hoffmann*

Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
firstname.lastname@deri.org

Abstract. We will investigate scalable algorithms for automated Semantic Web Service Composition (WSC). Our notion of WSC is very general: it allows the generation of new constants by Web service outputs; the composition semantics includes powerful background ontologies; and we use the most general notion of matching, *partial matches*, where several services can cooperate each covering only a part of a requirement. Herein, we define a first formalism. We show that automatic composition is very hard: even testing solutions is Π_p^2-complete. We identify a special case that covers many relevant WSC scenarios, and where solution testing is only **coNP**-complete. While **coNP** is still hard, in the area of planning under uncertainty, scalable tools have been developed that deal with the same complexity. In our next step, we will adapt the techniques underlying one of these tools to develop a scalable tool for WSC. In the long term, we will investigate richer formalisms, and accordingly adapt our algorithms.

1 Introduction

In any task that involves automatic processing of Web services, one needs support for background ontologies. In the case of WSC, almost all existing solutions compile the problem into AI Planning formalisms. The motivation is that planning tools have become many times more scalable in recent years, through the use of heuristic functions and other search techniques, e.g. [8]. The problem here is that those tools cannot handle background ontologies. The following example illustrates the importance of those:

Example 1. A service shall be composed that inputs a constant of concept A, and outputs one of concept C. (E.g., A may be a trip request and C a ticket.) The ontology defines the concepts A, B, $B_1, \ldots, B_n$, C, and states that each $B_i \subseteq B$, and $\bigcup_1^n B_i \supseteq B$. (E.g., B may be a geographical region and the B_i its parts.) An available service ws_{AB} transforms A into B, and for each i an available service ws_{B_iC} transforms B_i into C. A solution first applies ws_{AB}, and then applies the ws_{B_iC} in conjunction.

In Example 1, reasoning over the background ontology is necessary to (1) understand which services can be used, and to (2) test whether a given composition is actually a solution. Such reasoning can be modelled through the *background theories* explored in some planning works, e.g., [4, 6]. However, incorporating this notion into the modern scalable planning tools poses serious challenges, and has not yet even been tried. Due to the background theory, even computing a state transition – which is now a form of belief revision – is a computationally very hard task. The existing planning tools dealing

* Ph.D. Supervisor

with background theories, e.g., [4, 6], map the problem into generic deduction, which is well known for its lack of scalability. The existing planning tools dealing with WSC, e.g., [9, 2], ignore the ontology and assume *exact matches*. In Example 1, this would require $B = B_i$ instead of $B \cap B_i \neq \emptyset$. Obviously, this renders the example unsolvable.

We identify an interesting special case of WSC. We exploit the fact that Web services may output new constants.[1] Now, if all ramifications of a Web service concern only propositions involving at least one new constant, then a belief revision is not necessary. We term this special case WSC with *forward effects*: the effects are "forward" in the sense that no backwards-directed belief revision is necessary. E.g., the services in Example 1 have forward effects. The same holds for many WSC scenarios from the literature, and from real case studies. 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 introduce a framework for planning with background theories.[2] We show that testing whether an action sequence is a solution – *solution testing* – is Π_2^p-complete in general, but only **coNP**-complete with forward effects. Of course, **coNP** is still hard. However, *planning under uncertainty* has the same complexity of solution testing, and scalable tools for this case have already been developed. Adapting them for WSC with forward effects will be our next step. In particular, the Conformant-FF tool [7] is based on CNF reasoning, which can be naturally extended to our setting.

Section 2 introduces our formalism, Section 3 discusses forward effects. Sections 4 provides some details regarding our future developments, Section 5 concludes.

2 WSC with Partial Matches

The (planning) terminology of our formalism corresponds to WSC is as follows. Web services are planning "operators"; their input/output behaviour maps to input/output parameters on which preconditions and effects are specified [1, 3, 5]. The background ontology is the background "theory". The precondition in the goal is equivalent to sets of "initial literals" and "initial constants". The effect in the goal is the "goal condition".

We assume supplies of logical predicates p, q, variable names x, y and constant names a, b, c, d, e; *(ground) literals* are defined as usual. For variables X, $\mathcal{L}^X$ is the set of literals using only variables from X. We write $l[X]$ for a literal l with variable arguments X. For a tuple C of constants substituting X, we write $l[C/X]$. In the same way, we use the substitution notation for any construct involving variables. Positive ground literals are *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

[1] Namely, the outputs model the generated data.

[2] [10] defines a similar WSC formalism, but considering plug-in matches and restricting the background theory, instead of the service effects, to obtain efficiency.

of constants; for E_a we require that the constants are pairwise different. In this way "operators" are Web services and "actions" are service calls. $\mathcal{WSC}$ *tasks* are tuples $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. Here, $\mathcal{P}$ are predicates; $\mathcal{T}$ is the 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 from $\mathcal{P}$, all constants are from C_0, all constructs are finite.

Assume we are given a task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$. *States* in our formalism are pairs (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 all propositions formed from the predicates $\mathcal{P}$ and the constants C_s. We need to define the outcome of applying actions in states. Given a state s and an action a, a is *applicable in* s if $I_s \models \text{pre}_a$, $C_a \subseteq C_s$, and $E_a \cap C_s = \emptyset$. We allow *parallel actions*. These are sets of actions which are applied at the same point in time. The result of applying a parallel action A in a state s is $res(s, A) :=$

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

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 [11]. A is *inconsistent* with s iff $res(s, A) = \emptyset$; this can happen in case of conflicts. Note that $res(s, A)$ allows non-applicable actions. This realizes partial matches: a Web service can be applied as soon as it matches at least one possible situation.

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 *solution* is a sequence $\langle A_1, \ldots, A_n \rangle$ s.t. for all $s \in res(b_0, \langle A_1, \ldots, A_n \rangle) : s \models \phi_G$.

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, solution testing is Π_2^p-complete in $\mathcal{WSC}$. Further, we have proved that polynomially bounded solution existence is Σ_3^p-complete.

Theorem 1 (Solution testing in $\mathcal{WSC}$). *Assume a $\mathcal{WSC}$ task 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 solution.*

3 Forward Effects

The high complexity of $\mathcal{WSC}$ motivates the search for interesting special cases. As stated, here we define a special case where every change an action makes to the state involves a new constant. A $\mathcal{WSC}$ task $(\mathcal{P}, \mathcal{T}, \mathcal{O}, C_0, \phi_0, \phi_G)$ has *forward effects* iff:

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

– 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.

The set of all such tasks is denoted with $\mathcal{WSC}|_{fwd}$. The second condition implies that effects involving new constants can only affect literals involving new constants. Given a state s and a parallel action A, define $res|_{fwd}(s, A) :=$

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

Here, $I'|_{C_s}$ denotes the restriction of I' to the propositions over C_s.

Proposition 1 (Semantics of $\mathcal{WSC}|_{fwd}$). *Assume a $\mathcal{WSC}|_{fwd}$ task, a state s, and a parallel action A. Then $res(s, A) = res|_{fwd}(s, A)$.*

Hence, the action semantics becomes a lot simpler with forward effects, no longer needing the notion of minimal changes with respect to the previous state. We get:

Proposition 2 (Solution testing in $\mathcal{WSC}|_{fwd}$). *Assume a $\mathcal{WSC}|_{fwd}$ task with fixed arity, and 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 solution.*

It is currently an open problem what the complexity of deciding polynomially bounded solution existence is in $\mathcal{WSC}|_{fwd}$. With Proposition 2, membership in Σ_p^2 is easy to see. However, we suspect that the problem is actually **coNP**-complete. We have one half of a proof, but some tricky issues must still be resolved regarding the generation of exponentially many constants.

4 Tool and Language Developments

Our next step will be to develop a tool for $\mathcal{WSC}|_{fwd}$. We focus on achieving scalability: we expect practical SWS scenarios to involve large sets of Web services, involving huge search spaces (of partial compositions). We will try to overcome this by designing heuristic solution distance and search node filtering techniques. Specifically, we will start from the ideas underlying the Conformant-FF (CFF) [7] planning tool.

CFF represents beliefs in terms of propositional CNF formulas, and uses SAT reasoning to test solutions. We will adapt this to our purposes, simply by adapting the generation of the CNFs. Note that this is possible only in $\mathcal{WSC}|_{fwd}$, not in $\mathcal{WSC}$, for complexity reasons. CFF also introduces heuristic and filtering techniques, based on an abstraction of the planning problem. Namely, for each belief CFF computes an abstract solution using approximate SAT reasoning, and then uses the abstract solution to inform the search. We will apply the same principle for $\mathcal{WSC}|_{fwd}$. This will differ from CFF in the different structure of our CNFs, necessitating different approximate reasoning, and in that we will explore typical forms of background theories (e.g., subsumption hierarchies) to obtain better distance estimates. Further, in difference to us, CFF (and indeed every existing planning tool) does not allow the generation of new constants. We will devise new heuristics for dealing with this. Note that this is critical:

as already indicated, exponentially many constants may be generated in general, so one needs heuristics identifying which are important. Those heuristics need to be clever: if they remove too many constants, then the solutions may be cut out; if they remove too few constants, then the search space may explode.

In the long term, our line of research will be to incrementally enrich the language our tool accepts, accordingly adapting the algorithms. For a start, some generalisations of $\mathcal{WSC}|_{fwd}$ are possible without losing Proposition 1. Most importantly, instead of requiring that *every* effect literal involves a new constant, one can postulate this only for literals that may actually be affected by the background theory. This may be important, e.g., , for dealing with updates on attributes of existing constants. If such a language turns out to not be enough for many practical examples, then we will look into how feasible it is to drop the forward effects restriction and deal with full $\mathcal{WSC}$. Note that, due to Theorem 1, this would require QBF solving for reasoning about beliefs. We further plan to investigate into allowing non-deterministic outcomes of Web services. These would be modelled as operators with lists of alternative effects, each of which may occur. We expect that we can handle this by appropriately extending the CNF formulas underlying beliefs, as well as the associated machinery.

5 Conclusion

We have introduced a formalism for WSC, and identified a special case for which no belief revision is necessary. We will solve some open complexity problems, and develop a tool inspired from CFF. We will incrementally extend the tool to richer languages.

References

1. A. Ankolenkar et al. DAML-S: Web service description for the semantic web. In *ISWC*, 2002.
2. R. Akkiraju, B. Srivastava, I. Anca-Andreea, R. Goodwin, and T. Syeda. Semaplan: Combining planning with semantic matching to achieve web service composition. In *ICWS*, 2006.
3. D. Martin et al. OWL-S: Semantic markup for web services. In *SWSWPC*, 2004.
4. T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A logic programming approach to knowledge-state planning, II: The DLVK system. *AI*, 144(1-2):157–211, 2003.
5. D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag, 2006.
6. E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *AI*, 153(1-2):49–104, 2004.
7. J. Hoffmann and R. Brafman. Conformant planning via heuristic forward search: A new approach. *AI*, 170(6–7):507–541, May 2006.
8. J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.
9. S. Ponnekanti and A. Fox. SWORD: A developer toolkit for web services composition. In *WWW*, 2002.
10. J. Scicluna and J. Hoffmann. Semantic web service composition: Background theories, parallelisation, and business policies. In *ESWC PhD Symposium*, 2007. Submitted.
11. M. Winslett. Reasoning about actions using a possible models approach. In *AAAI*, 1988.