Reasoning about Partially Ordered Web Service Activities in PSL

Michael Gruninger and Xing Tan

Semantic Technologies Laboratory
Department of Mechanical and Industrial Engineering
University of Toronto

Abstract. Many tasks within semantic web service discovery can be formalized as reasoning problems related to the partial ordering of subactivity occurrences in a complex activity. We show how the first-order ontology of the Process Specification Language (PSL) can be used to represent both the queries and the process descriptions that constitute the underlying theory for the reasoning problems. We also identify extensions of the PSL Ontology for which these problems are NP-complete and then explicitly axiomatize classes of activities for which the various reasoning problems are tractable.

1 Introduction

Ontologies that represent complex activities (such as composite web services and manufacturing process plans) are required for many applications of automated reasoning. We typically need to specify the occurrence and ordering constraints over the different subactivities; such constraints may either be explicitly represented or they may be entailed by other properties, such as preconditions and effects.

Within domains of semantic web service discovery, resources are associated with composite web service plans, which are partially ordered sequences of processes. In general, such process plans may also be nondeterministic (that is, involve different choices of sequences of processes). At any point in a web service plan, there are multiple activities that can possibly occur next. Furthermore, different web service plans may have processes in common so that an object may participate in an activity that is part of multiple plans. This scenario motivates four queries that are relevant for the discovery and verification of semantic web service plans, and which we will formalize later in this paper:

1. Is it possible for one activity in a web service process to occur before some other activity?
2. Is one activity in a web service process required to occur before some other activity?
3. Given the occurrence of some activity in a web service process, what activities are required to occur later?

4. Given the occurrence of some activity in a web service process, what activities can possibly occur next?

In this paper, we will focus on the formalization of these four queries as first-order entailment problems which are related to occurrences of complex activities and the ordering constraints on those occurrences. To achieve this objective, we use a first-order ontology in which complex activities and their occurrences are elements of the domain, so that web service discovery can be expressed as entailment problems that are solved by inference techniques that are sound and complete with respect to models of the ontology. Inference is done using the axioms of the ontology alone, without resorting to extralogical assumptions or special algorithms used by interpreters.

In particular, we will use the PSL Ontology to axiomatize the constraints in the antecedents, as well as the queries in the consequents, of the entailment problems.[1] Furthermore, we use the model theory of the PSL Ontology to provide correctness theorems for the specification of the queries and the process descriptions for certain classes of activities. Finally, we will define extensions of the PSL Ontology in which two of the entailment problems are NP-complete, and additional extensions in which these entailment problems are tractable. In this way, tractable classes of the problems are explicitly axiomatized within the ontology itself, and the relationships between different assumptions can themselves be determined by first-order theorem proving.

2 PSL Ontology

As a modular set of theories in the language of first-order logic, the Process Specification Language (PSL) [6, 7] has been designed to facilitate correct and complete exchange of process information;[2] whereas it has been adopted by the Semantic Web Services Language (SWSL) Committee of Semantic Web Services Initiative (SWSI) [3] to specify the model-theoretic semantics of Semantic Web Services Ontology (SWSO) ([3], [8]), one of the two major components within Semantic Web Services Framework (SWSF) [2].

Within the PSL Ontology, all core theories are consistent extensions of a theory referred to as PSL-Core, which introduces the basic ontological commitment to a domain of activities, activity occurrences, timepoints, and objects that participate in activities. Additional core theories capture the basic intuitions for the composition of activities, and the relationship between the occurrence of a complex activity and occurrences of its subactivities.

In order to formally specify a broad variety of properties and constraints on complex activities, we need to explicitly describe and quantify over complex

[1] Given the entailment problem $T \models \phi$, we say that T is the antecedent and ϕ is the consequent.

[2] PSL has been accepted as an International Standard (ISO 18629) within the International Organisation of Standardisation. The full set of axioms in the Common Logic Interchange Format is available at `http://www.mel.nist.gov/psl/ontology.html`

[3] `http://www.swsi.org`

activities and their occurrences. Within the PSL Ontology, complex activities and occurrences of activities are elements of the domain and the *occurrence_of* relation is used to capture the relationship between different occurrences of the same activity.

A second requirement for formalizing the queries is to specify composition of activities and occurrences. The PSL Ontology uses the *subactivity* relation to capture the basic intuitions for the composition of activities. Complex activities are composed of sets of *atomic* activities, which in turn are either *primitive* (i.e. they have no proper subactivities) or they are concurrent combinations of primitive activities.

Corresponding to the composition relation over activities, *subactivity_occurrence* is the composition relation over activity occurrences. Given an occurrence of a complex activity, subactivity occurrences are occurrences of subactivities of the complex activity.

Within the PSL Ontology, concurrency is represented by the occurrence of concurrent activities rather than concurrent activity occurrences. We use the following relation to generalize the notion of occurrence to include any atomic activity that is a subactivity of the activity that occurs:

$$(\forall s, a)\, atocc(s, a) \equiv (\exists a_1)\, atomic(a_1) \wedge occurrence\_of(s, a_1) \wedge subactivity(a, a_1)$$

Finally, we need some way to specify ordering constraints over the subactivity occurrences of a complex activity. The PSL Ontology uses the $soo\_precedes(s_1, s_2, a)$ relation to denote that subactivity occurrence s_1 precedes the subactivity occurrence s_2 in occurrences of the complex activity a.

The models of the axioms of the PSL Ontology have been characterized up to isomorphism [7]. A fundamental structure within these models is the occurrence tree, whose branches are equivalent to all discrete sequences of occurrences of atomic activities in the domain. Elements of the occurrence tree are referred to as *arboreal* occurrences.

Although occurrence trees characterize all sequences of activity occurrences, not all of these sequences will intuitively be physically possible within a given domain. We therefore consider the subtree of the occurrence tree that consists only of possible sequences of activity occurrences, which we refer to as the legal occurrence tree. The $legal(o)$ relation specifies that the atomic activity occurrence o is an element of the legal occurrence tree.

The basic structure that characterizes occurrences of complex activities within models of the ontology is the activity tree, which is a subtree of the legal occurrence tree that consists of all possible sequences of atomic subactivity occurrences; the relation $root(s, a)$ denotes that the subactivity occurrence s is the root of an activity tree for a. Elements of the tree are ordered by the *soo_precedes* relation; each branch of an activity tree is a linearly ordered set of occurrences of subactivities of the complex activity. In addition, there is a one-to-one correspondence between occurrences of complex activities and branches of the associated activity trees.

In a sense, an activity tree is a microcosm of the occurrence tree, in which we consider all of the ways in which the world unfolds in the context of an occurrence of the complex activity. Different subactivities may occur on different branches of the activity tree – different occurrences of an activity may have different subactivity occurrences or different orderings on the same subactivity occurrences (see the examples[4] in Figures 1 through 5). This distinction plays a key role in the specification of the entailment problems in this paper.

3 Formalization of the Entailment Problems

We can use the PSL Ontology to specify the queries (informally posed in the introduction) as the consequents of first-order entailment problems. The antecedent of the entailment problems will consist of the following sets of sentences:

- T_{psl} : the axioms of the PSL Ontology, together with the following three sentences:
  - Activity closure (primitive and complex)
    $(\forall a)\, primitive(a) \equiv (a = A_1) \vee ... \vee (a = A_n))$
    $(\forall a)\, \neg atomic(a) \equiv (a = P_1) \vee ... \vee (a = P_m))$
    where $A_1, ..., A_n, P_1, ..., P_n$ are constants denoting activities.
  - Legal Occurrence Assumption [5]
    $(\forall o, a)\, occurrence\_of(o, a) \wedge atomic(a) \supset legal(o)$
- $\Sigma_{pd}(P_i)$: the process description for the complex activity P_i, which specifies the relationship between occurrences of the activity and its subactivities.

In this section, we first focus on the queries in the consequents of the problems, and then define the classes of activities and process descriptions that constitute the antecedents of the problems.

3.1 Queries

The query in the consequent of one of our entailment problems is a first-order sentence that is satisfied by properties of the activity trees within the models of T_{psl} and process descriptions. We can apply the model theory of the PSL Ontology to provide characterizations of the activity trees for each query that we consider, demonstrating the correctness of the sentence with respect to the intended properties of the activity trees. In the motivating scenarios from the introduction, process plans and composite web services are represented in the PSL Ontology as complex activities. The four particular queries that we formalize in this paper focus on the relationship between occurrences of complex activities and their subactivities.

[4] In these examples, we adopt the convention that o_i^a denotes an occurrence of the activity a. In all of the examples, we use the primitive activities *register*, *hotel*, *airplane*, *payment*, and *train*.

[5] We use this assumption in the complexity analysis to focus on the intractability that arises solely from occurrence and ordering constraints, independently of preconditions and effects.

To formalize the first query, we want a sentence that determines whether the subactivity A_1 can possibly occur before the subactivity A_2, whenever the complex activity P occurs; such a sentence is characterized by the following result:

Lemma 1. *Suppose $\mathcal{M}$ is a model of $T_{psl} \cup \Sigma_{pd}(P)$.*
$\mathcal{M} \models (\forall o)\, root(o, P) \supset (\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_2, P) \wedge soo\_precedes(o_1, o_2, P)$
iff any activity tree for the complex activity P contains a branch in which a subactivity occurrence o_1 of A_1 precedes a subactivity occurrence o_2 of A_2.

The activity tree in Figure 1 contains a branch in which the subactivity *hotel* occurs before *airplane* and also contains a branch in which the subactivity *airplane* occurs before *hotel*. In the same activity tree, there does not exist a branch in which the *payment* subactivity occurs before the *register* subactivity. In Figure 2, there does not exist any branch containing occurrences of both subactivities *airplane* and *train*. The following lemma characterizes the sentence that is satisfied when the subactivity A_1 is required to occur before the subactivity A_2 in occurrences of the complex activity P:

Lemma 2. *Suppose $\mathcal{M}$ is a model of $T_{psl} \cup \Sigma_{pd}(P)$.*
$\mathcal{M} \models (\forall o, o_1, o_2)\; root(o, P) \wedge occurrence\_of(o_1, A_1) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_1, P) \wedge soo\_precedes(o, o_2, P) \supset \neg soo\_precedes(o_2, o_1, P)$
iff for any branch $\mathbb{B}$ in any activity tree for the complex activity P, either

1. *every occurrence of the subactivity A_1 in $\mathbb{B}$ precedes every occurrence of the subactivity A_2 in $\mathbb{B}$, or*
2. *$\mathbb{B}$ does not contain occurrences of both A_1 and A_2.*

For example, in Figure 4, the subactivity *hotel* occurs before *payment* on every branch of the activity tree. On the other hand, there is no branch in Figure 2, that contains occurrences of both *train* and *airplane*.

The third query from the introduction determines whether occurrences of the subactivity A_1 are followed by later occurrences of the subactivity A_2 in occurrences of the complex activity P. This sentence is characterized by the next result:

Lemma 3. *Suppose $\mathcal{M}$ is a model of $T_{psl} \cup \Sigma_{pd}(P)$.*
$\mathcal{M} \models (\forall o, o_1)\; root(o, P) \wedge occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \supset (\exists o_2)\; occurrence\_of(o_2, A_2) \wedge soo\_precedes(o_1, o_2, P)$
iff every occurrence of the subactivity A_1 is the initial element of a subtree of an activity tree for P that contains an occurrence of the subactivity A_2.

Figure 2 illustrates this query, where every occurrence of the subactivity *train* is followed by an occurrence of the subactivity *payment*.

The final query determines which activities can possibly occur next, given the occurrence of some activity A_1 in a process plan P. The following lemma characterizes the sentence that defines this query:

Lemma 4. *Suppose $\mathcal{M}$ is a model of $T_{psl} \cup \Sigma_{pd}(P)$.*
$\mathcal{M} \models (\forall o, o_1)\, root(o, P) \wedge occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \supset$
$(\exists a, o_2)\, occurrence\_of(o_2, a) \wedge soo\_precedes(o_1, o_2, P) \wedge$
$\neg((\exists o_3)\, soo\_precedes(o_1, o_3, P) \wedge soo\_precedes(o_3, o_2, P))$
iff no occurrence o_1 of the subactivity A_1 is a leaf of an activity tree for P.

In any proof of the above sentence with answer extraction, the variable a binds to one of the successors of the occurrence of A_1 in an activity tree for P. In Figure 2, the next subactivity to occur after o_2^{hotel} is either *airplane* or *train*.

3.2 Classification of Activities

One set of sentences within the antecedent of the entailment problems is the extension of the ontology with restricted classes of activities. Within the PSL Ontology, complex activities are classified with respect to symmetries of their activity trees. Concretely, these are axiomatized by mappings between the different branches of an activity tree or between different activity trees. In this section we introduce the model-theoretic definitions for the classes of activity trees that play a prominent role in this paper; their first-order axiomatization can be found in the PSL Ontology.

Definition 1. *An activity tree τ in a model of T_{psl} is permuted iff for every two branches $\mathbb{B}_1, \mathbb{B}_2 \subseteq \tau$, there exists a bijection $\varphi : \mathbb{B}_1 \to \mathbb{B}_2$ such that for any activity occurrence $\mathbf{o} \in \mathbb{B}_1$ and activity $\mathbf{a}$,*

$$\langle \mathbf{o}, \mathbf{a} \rangle \in \mathbf{occurrence\_of} \Leftrightarrow \langle \varphi(\mathbf{o}), \mathbf{a} \rangle \in \mathbf{occurrence\_of}$$

Figure 1 shows an example of a permuted activity tree; there is a bijection that maps the subactivity occurrences o_2^{hotel}, $o_3^{airplane}$, $o_6^{payment}$ in the branch $\mathbb{B}_1$ to the subactivity occurrences o_5^{hotel}, $o_7^{airplane}$, $o_4^{payment}$, respectively, in the branch $\mathbb{B}_2$. Since the occurrence $o_1^{register}$ is an element of every branch, it is mapped to itself. Intuitively, each branch of a permuted activity tree is a different permutation of the same set of subactivity occurrences; in the example, the same activities (*register*, *hotel*, *airplane*, and *payment*) occur on each branch, although they occur in a different order. On the other hand, the activity tree in Figure 2 is not permuted, since there is no mapping between the branch containing $o_3^{airplane}$ and o_8^{train}.

Definition 2. *An activity tree τ in a model of T_{psl} is folded iff there exists a branch $\mathbb{B}_1 \subseteq \tau$ such that for any branch $\mathbb{B}_2 \subseteq \tau$ there exists a surjection $\varphi : \mathbb{B}_2 \to \mathbb{B}_1$ such that for any activity occurrence $\mathbf{o} \in \mathbb{B}_1$ and activity $\mathbf{a}$,*

$$\langle \mathbf{o}, \mathbf{a} \rangle \in \mathbf{atocc} \Rightarrow \langle \varphi(\mathbf{o}), \mathbf{a} \rangle \in \mathbf{atocc}$$

With folded activity trees, the mappings between branches of the activity tree allow occurrences of atomic subactivities to be mapped to occurrences of concurrent subactivities. Figure 3 shows an example of a folded activity tree;

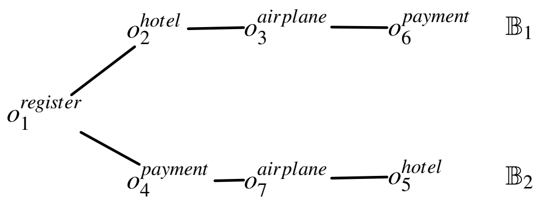

Fig. 1. Example of a permuted activity tree.

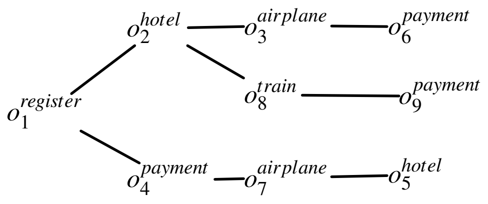

Fig. 2. Example of a nonpermuted activity tree.

the two subactivity occurrences $o_2^{register}$ and $o_4^{payment}$ on the branch $\mathbb{B}_2$ are mapped to the subactivity occurrence $o_6^{(register+payment)}$, which is an occurrence of the atomic activity whose primitive subactivities *payment* and *register* are concurrent.

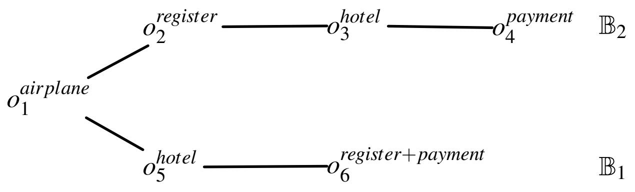

Fig. 3. Example of a folded activity tree.

Each activity tree can be associated with a partial ordering that is preserved by the mappings between branches, so that activity trees can be classified with respect to the relationship between this ordering and branches of the trees. This leads to two subclasses of permuted and folded activity trees that are particularly relevant to the specification of manufacturing process plans and semantic web services.

Definition 3. *Within a model of T_{psl}, a permuted activity tree is a strong poset activity tree iff there exists a partial linear such that there is a there is a one-to-one correspondence between its linear extensions and branches of the tree.*

Figure 4 is an example of a strong poset activity tree. The subactivities *hotel* and *airplane* are incomparable in the partial ordering (since the ordering of the occurrences of these two activities is not preserved on each branch), so there are two branches in the activity tree, corresponding to the two possible linear extensions.

Definition 4. *Within a model of T_{psl}, a folded activity tree is a concurrent poset activity tree iff there is a one-to-one correspondence between branches of the tree and the weak orderings of some set.*

Figure 5 is an example of a concurrent poset activity tree. The subactivities *hotel* and *register* are incomparable in the partial ordering, so there are two branches in the activity tree, corresponding to the two possible linear extensions; there is also a branch in the activity tree containing the occurrence of the activity in which *hotel* and *register* are concurrent. Note that any concurrent poset tree contains a subtree that is a strong poset tree.

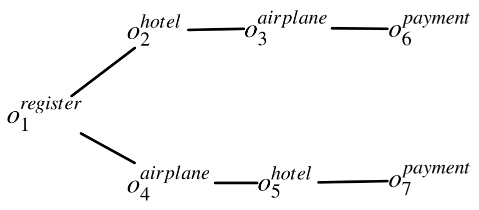

Fig. 4. Example of a strong poset activity tree.

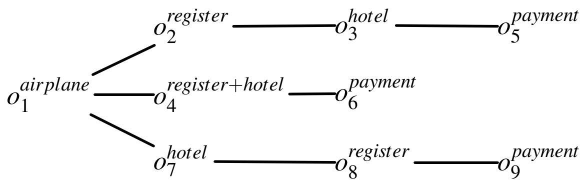

Fig. 5. Example of a concurrent poset activity tree.

Strong poset and concurrent poset activity trees capture the intended semantics of constructs that are present in a wide variety of approaches to process modelling, including OWL-S [11], UML activity diagrams [4], and IDEF3 [10]. In particular, strong posets are equivalent to the AnyOrder control construct in OWL-S and the AND-junctions of IDEF3, while concurrent posets are equivalent to the Split control construct in OWL-S and to forks in UML diagrams. Nevertheless, none of these constructs can be axiomatized in their respective formalisms, either because the underlying language lacks the expressiveness (as with OWL-S) or the formalism lacks a language with a formal model-theoretic semantics. For the same reasons, the relevant queries are not definable in these formalisms.

On the other hand, the class of strong posets has a rather elegant first-order axiomatization in the PSL Ontology that is based on the following property – within a strong poset, there exists a one-to-one mapping between siblings and children in the activity tree that preserves the *occurrence_of* relation, for any two elements that are incomparable in the partial ordering. Thus, the activity tree in Figure 1 is not a strong poset activity tree since the next subactivity occurrence after o_2^{hotel} is not an occurrence of the subactivity *payment*.

Concurrent posets have a similar axiomatization, with the additional condition that for any two siblings in the activity tree there exists another sibling that is an occurrence of the concurrent activity that is composed the activities associated with the siblings.

3.3 Process Descriptions

A process description is an axiomatization of the set of activity trees for an activity within models of the PSL Ontology. The syntactic form of the process description is tightly constrained by the classes of activities in the ontology.

Theorem 1. *A complex activity P has a set of finite permuted activity trees iff its process description $\Sigma_{pd}(P)$ is logically equivalent to a sentence of the form*

$$(\forall o)\ occurrence\_of(o,P) \supset$$
$$[(\exists o_1,...,o_n)\ occurrence\_of(o_1,A_1) \wedge ... \wedge occurrence\_of(o_n,A_m) \wedge$$
$$subactivity(A_1,P) \wedge ... \wedge subactivity(A_m,P) \wedge \mathcal{O}(o_1,...,o_n,P) \wedge$$
$$((\forall s)\ arboreal(s) \supset subactivity\_occurrence(s,o) \equiv ((s=o_1) \vee ... \vee (s=o_n)))]$$

where $\mathcal{O}(o_1,...,o_n,P)$ is a boolean combination of soo_precedes literals whose only variables are $o_1,...,o_n$.

Proof. $\Rightarrow$:

Suppose that the complex activity P has finite permuted activity trees.

Since the activity trees are finite, all of their branches are finite. Furthermore, since there is a bijection between the branches in the tree, all branches have the same cardinality n. Thus, each branch consists of n occurrences of subactivities of P, so that we have

$$T_{psl} \cup \Sigma_{pd}(P) \models (\forall o)\ root(o,P) \supset [(\exists o_1,...,o_n)\ occurrence\_of(o_1,A_1) \wedge ... \wedge$$

$occurrence\_of(o_n, A_m) \wedge subactivity(A_1, P) \wedge ... \wedge subactivity(A_m, P) \wedge$
$\mathcal{O}(o_1, ..., o_n, P)$

Each of these is an arboreal subactivity occurrence of an occurrence of P, and all arboreal subactivity occurrence of an occurrence of P are elements of a branch of an activity tree. We therefore have

$T_{psl} \cup \Sigma_{pd}(P) \models (\forall o)\ root(o, P) \supset [(\exists o_1, ..., o_n)\ occurrence\_of(o_1, A_1) \wedge ... \wedge occurrence\_of(o_n, A_m) \wedge ((\forall s) arboreal(s) \supset subactivity\_occurrence(s, o) \equiv ((s = o_1) \vee ... \vee (s = o_n)))]$

$\Leftarrow$:

Suppose that P has the process description Σ_{pd}.

This process description entails that every occurrence of P has exactly n occurrences of atomic subactivities of P. Therefore, each branch of an activity tree for P has occurrences of the same set of atomic subactivities, so that we can define a bijection between any two branches of the activity tree. Hence, the activity tree is permuted.

For example, the process description for the activity P_1 in Figure 1 is

$(\forall o)\ occurrence\_of(o, P_1) \supset [(\exists o_i, o_j, o_k, o_l)\ occurrence\_of(o_i, register)$
$\wedge occurrence\_of(o_j, hotel) \wedge occurrence\_of(o_k, airplane) \wedge$
$occurrence\_of(o_l, payment) \wedge subactivity(register, P_1) \wedge subactivity(hotel, P_1) \wedge subactivity(airplane, P_1) \wedge subactivity(payment, P_1) \wedge soo\_precedes(o_i, o_j, P_1) \wedge soo\_precedes(o_j, o_k, P_1) \wedge (soo\_precedes(o_l, o_k, P_1) \equiv \neg soo\_precedes(o_k, o_l, P_1))$
$\wedge((\forall s)\ arboreal(s) \supset subactivity\_occurrence(s, o) \equiv ((s = o_i) \vee (s = o_j) \vee (s = o_k) \vee (s = o_l)))]$

For folded activity trees, we have a similar, albeit weaker, result:

Theorem 2. *If the complex activity P has a set of finite folded activity trees, then its process description $\Sigma_{pd}(P)$ entails a sentence of the form*

$(\forall o)\ occurrence\_of(o, P) \supset [(\exists o_1, ..., o_n)\ atocc(o_1, A_1) \wedge ... \wedge atocc(o_n, A_m) \wedge subactivity(A_1, P) \wedge ... \wedge subactivity(A_m, P) \wedge \mathcal{O}(o_1, ..., o_n, P) \wedge ((\forall s) arboreal(s) \supset subactivity\_occurrence(s, o) \equiv ((s = o_1) \vee ... \vee (s = o_n)))]$

where $\mathcal{O}(o_1, ..., o_n, P)$ is a boolean combination of soo_precedes and equality literals whose only variables are $o_1, ..., o_n$.

Proof. Suppose that the complex activity P has finite folded activity trees.

Since the activity trees are finite, all of their branches are finite. Furthermore, since there is a surjection from the branches in the tree into a unique maximal branch, this branch has the maximum cardinality n. Thus, each branch consists of at most n occurrences of subactivities of P.

By the definition of folded activity trees, for each element **s** of a branch of an activity tree that is an occurrence of the subactivity **a** of P, there exists an element $\mathbf{s}'$ of the maximal branch such that

$$\langle \mathbf{s}', \mathbf{a} \rangle \in \mathbf{atocc}$$

We therefore have

$T_{psl} \cup \Sigma_{pd}(P) \models (\forall o) root(o, P) \supset [(\exists o_1, ..., o_n) atocc(o_1, A_1) \wedge ... \wedge atocc(o_n, A_m) \wedge subactivity(A_1, P) \wedge ... \wedge subactivity(A_m, P)$

Each element of a branch is an arboreal subactivity occurrence of an occurrence of P, and all arboreal subactivity occurrence of an occurrence of P are elements of a branch of an activity tree:

$T_{psl} \cup \Sigma_{pd}(P) \models (\forall o) root(o, P) \supset [(\exists o_1, ..., o_n) atocc(o_1, A_1) \wedge ... \wedge atocc(o_n, A_m) \wedge ((\forall s)\ arboreal(s) \supset subactivity\_occurrence(s, o) \equiv ((s = o_1) \vee ... \vee (s = o_n)))]$

The additional conditions in the definitions of strong poset activities and concurrent poset activities also impose restrictions on their process descriptions.

Theorem 3. *If a complex activity P has finite strong poset or concurrent poset activity trees, then the ordering formula $\mathcal{O}(o_1, ..., o_n, P)$ in its process description is logically equivalent to a conjunction of soo_precedes literals.*

For example, the process description for the activity P_2 in Figure 4 is

$(\forall o)\ occurrence\_of(o, P_2) \supset [(\exists o_i, o_j, o_k, o_l)\ occurrence\_of(o_i, register)$
$\wedge occurrence\_of(o_j, hotel) \wedge occurrence\_of(o_k, airplane) \wedge$
$occurrence\_of(o_l, payment) \wedge subactivity(register, P_1) \wedge subactivity(hotel, P_1) \wedge$
$subactivity(airplane, P_1) \wedge subactivity(payment, P_1) \wedge soo\_precedes(o_i, o_j, P_2) \wedge$
$soo\_precedes(o_i, o_k, P_2) \wedge soo\_precedes(o_j, o_l, P_2) \wedge soo\_precedes(o_k, o_l, P_2)$
$\wedge((\forall s)\ arboreal(s) \supset$
$subactivity\_occurrence(s, o) \equiv ((s = o_i) \vee (s = o_j) \vee (s = o_k) \vee (s = o_l)))]$

4 Complexity of Reasoning Problems

We can now consider the computational complexity of the entailment problems, under the assumptions that the process descriptions axiomatize the activity trees in the classes that we have presented above. In particular, we introduce additional assumptions to specify extensions to the PSL ontology, and then determine the complexity of the entailment problems in these extensions.

Definition 5. *The Permuted or Folded Occurrence Assumption (PFOA) is the sentence* [6]*:*

[6] *permuted*(o), *folded*(o), *strong_poset*(o), and *concurrent_poset*(o) are the relations defined within the PSL Ontology to axiomatize the corresponding classes of activity trees.

$(\forall o, a)\ occurrence\_of(o, a) \wedge \neg atomic(a) \supset (permuted(o) \vee folded(o))$

The Strong or Concurrent Poset Assumption (SCPA) is the sentence:
$(\forall o, a) occurrence\_of(o, a) \wedge \neg atomic(a) \supset (strong\_poset(o) \vee concurrent\_poset(o))$

It can be shown that $T_{psl} \models SCPA \supset PFOA$.

The following results show that these two assumptions are close to the boundary between tractability and intractability for the entailment problems that we have defined.

Theorem 4. *Suppose the complex activity P has only finite activity trees. Determining*

$T_{psl} \cup \Sigma_{pd}(P) \cup PFOA \models (\forall o)\ root(o, P) \supset (\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_2, P) \wedge soo\_precedes(o_1, o_2, P)$

is NP-complete.

Proof. By Theorem 1 and 2, there are n existentially quantified activity occurrence variables in $\Sigma_{pd}(P)$. By $PFOA$, any branch of an activity tree contains at most n atomic activity occurrences, and the maximum number of branches in any activity tree is equal to the number of weak orderings on a set of n points. Thus, the problem is in NP, since by Lemma 1 we need to check whether the branch contains a subactivity occurrence of A_1 that precedes a subactivity occurrence of A_2.

For folded activity trees, the proof can be found in [14], which first provides a straightforward reduction from an instance I of $Isat$ problem [1] in Interval Algebra (represented as a set of precedence and/or concurrency restrictions between endpoints of intervals of the instance) into $f(I)$, an instance of the problem of determining the existence of a complex activity P (composed of subactivity occurrences with corresponding *soo_precedes* and/or *conc* constraints) occurrences. A new subactivity occurrence o_i that precedes any other occurrences o_j is then added to construct a new complex activity P'. It is obvious I is satisfiable iff $soo\_precedes(o_i, o_j, P')$.

For permuted activity trees, since all of the occurrence variables denote distinct subactivity occurrences and the ordering formulae in the process description is a boolean combination of *soo_precedes* literals, NP-completeness follows from a straightforward reduction from 3SAT.

Thus, this entailment problem (whose query was characterized in Lemma 1) is intractable even when we restrict the activities to have permuted or folded activity trees.

If we strengthen the assumption so that we consider only strong poset or concurrent poset activity trees, then we obtain an extension of the theory in which the entailment problem is tractable.

Theorem 5. *Suppose the complex activity P has only finite activity trees. There exists an $O(n^2)$ algorithm to determine*

$T_{psl} \cup \Sigma_{pd}(P) \cup SCPA \models (\forall o)\ root(o,P) \supset (\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_2, P) \wedge soo\_precedes(o_1, o_2, P)$

where n is the number of existentially quantified activity occurrence variables in $\Sigma_{pd}(P)$.

Proof. Suppose that $\Sigma_{pd}(P)$ contains n activity occurrence variables and m *soo_precedes* literals. By Theorem 3, we can construct a directed graph $G = \langle V, E\rangle$, where V is the set of subactivity occurrence variables occ_i, and $(occ_i, occ_j) \in E$ iff the literal $soo\_precedes(occ_i, occ_j, P)$ is in $\Sigma_{pd}(P)$. $\Sigma_{pd}(P)$ is consistent (and hence there exists an occurrence of P) iff there exists a linear ordering on the vertices in V. This can be found using a topological sort algorithm, whose complexity is $O(n + m)$, where the upper bound for m is $n(n-1)/2$ (for a complete graph). Now, let Φ be the existential conjunction that is the consequent of the query. We can define another process description

$$\Sigma_{pd}(P') = \Sigma_{pd}(P) \wedge \Phi$$

Similarly, we know that checking the consistency of $\Sigma_{pd}(P')$ (which is equivalent to the existence of P') can also be solved in $O(n^2)$ time (as n stays unchanged). And it is straightforward to see that the existence of occurrence of P' implies that it is possible that there exists a subactivity occurrence of A_1 before a subactivity occurrence of A_2 in an activity tree for P.

Note that the algorithm requires a process description with a fixed set of subactivity occurrence variables and ordering constraints that are equivalent to a conjunction of *soo_precedes* literals. Although the first condition is satisfied by all permuted and folded activities, only strong poset and concurrent poset activities have process descriptions that satisfy the second condition.

We can also consider the query that we characterized in Lemma 2:

Theorem 6. *Suppose the complex activity P has only finite activity trees. Determining*

$T_{psl} \cup \Sigma_{pd}(P) \cup PFOA \models (\forall o, o_1, o_2)\ root(o,P) \wedge occurrence\_of(o_1, A_1) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_1, P) \wedge soo\_precedes(o, o_2, P) \supset \neg soo\_precedes(o_2, o_1, P)$

is NP-complete.

Proof. By Lemma 2, the sentence is logically equivalent to

$(\forall o)\ root(o,P) \supset \neg(\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \wedge occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_2, P) \wedge soo\_precedes(o_1, o_2, P)$

Since the activity trees for P are either permuted or folded, we know that the same set of activities occur on every branch, so that the above sentence becomes

$(\forall o)\, root(o, P) \supset (\exists o_1, o_2) occurrence\_of(o_1, A_1) \wedge soo\_precedes(o, o_1, P) \wedge$
$occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_2, P) \wedge soo\_precedes(o_1, o_2, P)$
which is equivalent to the sentence in Theorem 4.

Once again, if we use the Strong or Concurrent Poset Assumption, then we have an extension of the PSL Ontology in which the entailment problem is tractable.

Theorem 7. *Suppose the complex activity P has only finite activity trees. There exists an $O(n^2)$ algorithm to determine*

$T_{psl} \cup \Sigma_{pd}(P) \cup SCPA \models (\forall o, o_1, o_2)\, root(o, P) \wedge occurrence\_of(o_1, A_1) \wedge$
$occurrence\_of(o_2, A_2) \wedge soo\_precedes(o, o_1, P) \wedge soo\_precedes(o, o_2, P) \supset$
$\neg soo\_precedes(o_2, o_1, P)$

where n is the number of existentially quantified activity occurrence variables in $\Sigma_{pd}(P)$.

Proof. We can use the algorithm from the proof of Theorem 5 to determine whether there exists branch containing a subactivity occurrence of A_1 before a subactivity occurrence of A_2 in an activity tree for P and a branch containing a subactivity occurrence of A_2 before a subactivity occurrence of A_1 in an activity tree for P. If one of these branches does not exist, then the ordering satisfied by the other branch is satisfied on all branches.

The complexity of the entailment problems characterized in Lemmas 3 and 4 is still open.

5 Summary

We have shown how the PSL Ontology can be used to define the antecedents and consequents for first-order entailment problems related to the partial ordering of subactivity occurrences in occurrences of complex activities. The model theory of the PSL Ontology also allows us to prove the correctness of the definitions of the queries, as well as the correctness of the process descriptions for the classes of activities used within this paper.

It is difficult to define these entailment problems using other process modelling ontologies. Approaches such as the Business Process Modelling Notation (BPMN) lack an ontological foundation. Ontologies such as [13] lack a model theory. The ontologies in [5] and [11] lack axiomatizations in their respective languages, so that we cannot formalize the queries as entailment problems. Ontologies such as [12] and [9] provide axiomatizations, but they lack an explicit and complete characterization of the models of the axiomatizations. These approaches also fail to make the distinction between the axioms in ontology and the classes of sentences in the process descriptions. As a result, it is difficult to define classes of activities such as *permuted* and *strong_poset*, and we are unable to prove the correctness of the process descriptions. Finally, approaches such as [9] are unable to quantify over complex activities and their occurrences, which is required by the entailment problems that we considered.

In addition to providing a model-theoretic characterization of the sentences in the antecedents and consequents of the entailment problems, we have also defined extensions of the PSL Ontology in which the associated entailment problems are NP-complete and stronger extensions in which the problems are tractable. This demonstrates that the PSL Ontology can not only be used to axiomatize the assumptions that guarantee tractability, but it can also be used to reason about the logical relationships among these assumptions.

There are several avenues for future work. First, we want to provide a sharper characterization of the boundary between tractable and intractable extensions of the PSL Ontology by finding the maximal classes of ordered activity trees that contain the strong poset and concurrent poset activity trees and for which the entailment problems are still tractable.

Second, there are many other classes of activity trees and activities in the PSL Ontology which are independent of the folded and permuted activity trees; no work has yet been done to characterize the complexity of the entailment problems with these extensions of the PSL Ontology. This includes the entailment of ordering constraints from precondition and effect axioms.

Finally, we can apply the methodology of defining tractable extensions of the PSL Ontology to reasoning problems including temporal projection, plan verification, and plan recognition.

References

1. Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.
2. Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., martin, D., McIlraith, S., McGuinness, D., Su, J., and Tabet, S. 2005. Semantic Web Services Framework (SWSF) Overview. Version 1.0, Available at: http://www.daml.org/services/swsf/1.0/overview/
3. Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., and Tabet, S. 2005. Semantic Web Services Ontology (SWSO). Version 1.0, Available at: http://www.daml.org/services/swsf/swso/
4. Bock, C., and Gruninger, M. 2005. PSL: A semantic domain for flow models. *Software and Systems Modeling* 4:209–231.
5. Ghallab, M., and McDermott, D. 1998. PDDL: The planning domain definition language v.2. Technical Report Technical Report CVC TR-98-003, Yale Center for Computational Vision and Control.
6. Gruninger, M., and Menzel, C. 2003. The process specification language theory and applications. *AI Mag.* 24(3):63–74.
7. Gruninger, M. 2004. Ontology of the Process Specification Language. In Staab, S., and Studer, R., eds., *Handbook of Ontologies in Information Systems.* Springer-Verlag.

8. Gruninger, M., Hull, R. and McIlraith, S. (2005) A First-Order Ontology for Semantic Web Services, *Proceedings of W3C Workshop on Frameworks for Semantic in Web Services*, Innsbruck, Austria (9th - 10th June 2005)
9. Levesque, H., Reiter, R.; Lesperance, Y.; Lin, F.; and Scherl, R. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:92–128.
10. Mayer, R.; Menzel, C.; Painter, M.; deWitte, P.; Blinn, T.; and B.Perekath. 1995. Information integration for concurrent engineering: IDEF3 process description capture method report. Technical Report Technical Report AL-TR-1995, Knowledge Based Systems Incorporated.
11. McIlraith, S.; Son, T.; and Zeng, H. 2001. Semantic web services. *Intelligent Systems* 16(2):46–53.
12. Pease, A., and Niles, I. 1998. IEEE standard upper ontology: A progress report. *The Knowledge Engineering Review* 17:65–70.
13. Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubn Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel: Web Service Modeling Ontology, *Applied Ontology*, 1(1): 77 - 106, 2005.
14. Tan, X. 2008. Computational properties of PSL problems. Technical report, Semantic Technologies Laboratory, Department of Mechanical and Industrial Engineering, University of Toronto.