Inconsistencies in the Process Specification Language (PSL)

Michael Beeson
San José State University
San Jose, CA
ProfBeeson@gmail.com

Jay Halcomb
H&S Information Systems
Guerneville, CA
jhalcomb@hsinfosystems.com

Wolfgang Mayer
University of South Australia
Adelaide, Australia
wolfgang.mayer@unisa.edu.au

Abstract

The Process Specification Language (PSL) [6] is a first-order logical theory designed to describe manufacturing or business processes and formalize reasoning about them. It has been developed by several authors over a period of years, yet it is inconsistent with the simplest axioms that preclude non-trivial models. We demonstrate several inconsistencies using an automated theorem prover and attempt to repair the inconsistencies. We conclude that even with our amendments, PSL with its infinite models remains inadequate to represent complex industrial processes. We propose an alternative axiomatization that admits finite models, which may be better suited to automated theorem provers and model finders than the current version of PSL.

1 Introduction

The *Process Specification Language* (PSL) [6] is an ontology designed to formalize reasoning about processes in first-order logic. The ontology has been developed by several authors over a decade or more and has become an ISO standard [1]. PSL is a modular theory where individual modules formalize specific aspects of processes and their activities. The full PSL consists of about a thousand axioms in 75 subtheories that build upon the central *PSL Outer Core* subtheory that includes about ninety axioms in seven modules.[1] Although PSL itself is a domain-independent ontology, it can be tailored to a specific application by adding domain-specific axioms. Statements constructed from predicates and terms defined in the Outer Core constrain the possible models of the theory and therefore can be used to characterize the valid processes and their execution sequences.

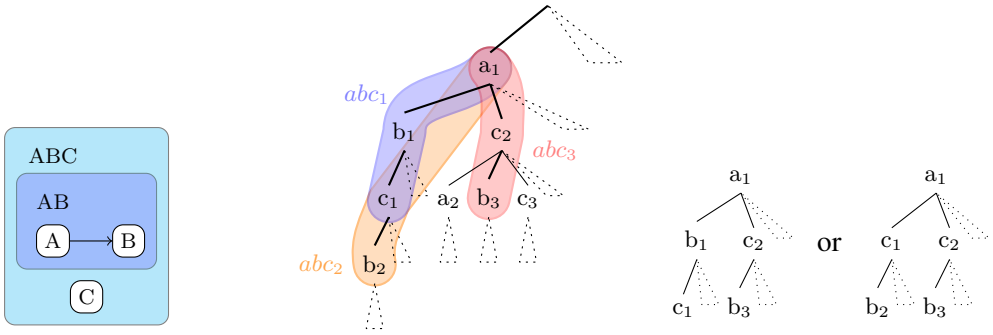
One of the underlying cornerstones of PSL is the idea that applications can be made interoperable by exchanging sentences formalized in PSL in order to share information about process specifications and concrete process executions. Formal theorem proving technology can then be employed to reason about processes and validate concrete executions in order to answer *Competency Questions* [9] of interest to the application domain. However, this requires a consistent theory.

The semantics of PSL are formulated using the notions and vocabulary of trees. These are Activity Trees, representing activities generically, and Occurrence Trees representing particular activity occurrences temporally, with activity trees intended to be subtrees of Occurrence Trees. However, as it stands no subset of PSL can consistently demonstrate the existence of any nontrivial Occurrence Trees. What is missing is a proof of consistency of PSL with simple axioms asserting the existence of a few activity occurrences. Without an intuitively comprehensible model, it will be difficult indeed to use PSL for actual process specifications.

Since its inception, PSL has undergone several revisions. As of April, 2011, the most recent version posted on the NIST website [6] is the May 2008 revision. We will refer to that version as PSL 2008. We also worked with a revision we call PSL 2010, supplied to us by Conrad Bock of NIST [7]. Unless stated otherwise, we use the term "PSL" to refer to PSL 2010.

In this paper we show that the version of PSL 2008 and its revision PSL 2010 contain flaws that lead to inconsistency even for the simplest process instances. We identify the responsible axioms and propose further revisions that resolve the inconsistencies. Although the resulting ontology can consistently

[1] The modules are: PSL Core, Subactivity Theory, Occurrence Tree Theory, Discrete State Theory, Atomic Activity Theory, Complex Activity Theory, and Complex Activity Occurrence Theory.



(a) Process Template (b) PSL Occurrence Tree (c) PSL Activity Trees

Figure 1: Example Process and PSL Trees

represent non-trivial processes, significant challenges remain to be addressed. In particular, it is difficult to obtain possible concrete (finite) models from the theory, whose axioms admit only infinite non-trivial models. We propose an alternative axiomatization that is based on Backofen et al.'s first-order theory of finite trees [2] that may be better suited to automated model construction.

First, we summarize the fundamental modeling primitives of PSL. Second, we describe inconsistencies we found in the course of our experiments and propose revisions to the axioms we believe are responsible for the inconsistency. Third, we outline where PSL may need to be strengthened in order to eliminate unwanted models. Fourth, we summarize our observations gathered from our experiments. Fifth, we introduce our Finite Tree PSL axiomatization before concluding the paper.

2 The PSL Outer Core Ontology

The PSL Outer Core formalizes basic properties of processes and the sequences of activities that may appear in a process execution, along with related objects and time points. The Outer Core is based on the following core concepts: *Activities* specify the actions and events that can appear in a process. Activities are partitioned into atomic activities and complex activities that consist of a number of sub-activities. An *Occurrence* of an activity represents an execution of the activity. Occurrences and activities are related via the binary *occurrence_of* predicate. PSL does not specify any concrete activities and occurrences; these must be introduced when tailoring the ontology to a specific application.

The *Occurrence Tree* contains all (infinite) sequences of occurrences of atomic activities starting from an initial state. This model is inspired by the Situation Calculus [10], where different branches in the situation tree represent alternative "worlds" where different sequences of actions apply. For each atomic activity X, the *successor* function maps each occurrence to its unique successor occurrence of X. Nodes in the tree are associated with timestamps and propositions that describe the process state before and after the occurrence. In addition, the occurrences in the branch of the tree must conform to the *earlier* predicate that embeds the sequence of occurrences in each branch into a totally ordered sequence of time points. One may be tempted to use *successor* or *earlier* to prune the Occurrence Tree; however, this may lead to inconsistency with the axioms of PSL. Instead, occurrences in branches that are admissible for a particular application domain are marked using the *legal* predicate.

Figure 1a shows a visual representation of a process specification where a composite activity *ABC* consists of activities *AB* and *C*, and activity *AB* consists of subactivities *A* and *B*. In any execution of *AB*, an occurrence of *A* must be followed by an occurrence of *B*. Within any occurrence of *ABC*, occurrences of *AB* and *C* may occur in any order. The relationship between complex activities and their subactivities is captured in the *subactivity* relation in PSL. Let activities *A*, *B*, and *C* be atomic, primitive activities that

have no subactivities. A PSL formalization in Prover9 syntax is available at [ABC.in].

The corresponding PSL Occurrence Tree is shown in Figure 1b. We use x_i to denote an occurrence of activity X, where i is an index that distinguishes different occurrences of an activity. Assume further that the activity occurrences $a_1, b_1, b_2, b_3, c_1, c_2$ are the only legal occurrences. Each node in the tree represents an occurrence of an atomic activity, and edges connect the occurrences to their direct successors. Bold edges represent *legal* branches, thin edges illegal branches, and dotted triangular edges denote subtrees that have been omitted for brevity (the tree has infinite depth and arbitrary branching factor). The tree contains two legal sequences of atomic activities: (a_1, b_1, c_1, b_2) and (a_1, c_2, b_3).

Occurrences of complex activities are defined by their atomic activities and are not part of the Occurrence Tree. Three complex activity occurrences $abc_{1,2,3}$ of activity ABC are drawn as hyperedges in Figure 1b. The complex occurrences of *AB* have been omitted. Occurrences of unrelated activities may appear in between activities of a complex occurrence. For example, b_1 is between a_1 and c_1 yet it does not belong to abc_2. The composition relationship between occurrences of complex activities and occurrences of their subactivities is formalized in the *subactivity_occurrence*, *root_occ* and *leaf_occ* predicates in PSL. For example, the complex activity abc_1 in Figure 1b is represented by the facts $subactivity\_occurrence(x, abc_1)$ for $x \in \{a_1, b_1, c_1, abc_1\}$, $root\_occ(a_1, abc_1)$ and $leaf\_occ(c_1, abc_1)$.

Activity Trees for a complex activity specify the possible orderings of atomic activity occurrences within a complex activity. Each Activity Tree is characterized by the *min_precedes* partial order between its occurrences and its *root* and *leaf* occurrences (all axiomatized as predicates). Each complex activity occurrence corresponds to a branch in an activity tree, which orders its subactivity occurrences. By imposing constraints on the activity trees, the models of PSL can be restricted to the complex occurrences that are legal in a particular application domain. For example, the branches in the left activity tree in Figure 1c are represented by $root(a_1, ABC) \wedge min\_precedes(a_1, b_1, ABC) \wedge min\_precedes(b_1, c_1, ABC) \wedge leaf(c_1, ABC) \wedge min\_precedes(a_1, c_2, ABC) \wedge min\_precedes(c_2, b_3, ABC) \wedge leaf(b_3, ABC)$.

A complex activity may have multiple activity trees—one for each occurrence in the Occurrence Tree that initiates a complex occurrence. Figure 1c shows two possible Activity Trees for our example. The tree on the left has branches for abc_1 and abc_3, whereas the right tree has abc_2 and abc_3. Note that the axioms of PSL prohibit us from constructing a single tree with branches for all three complex occurrences, as $leaf(c_1, ABC)$ and $min\_precedes(c_1, b_2, ABC)$ are mutually incompatible.

3 Inconsistencies in the PSL Outer Core

In order for PSL to be used effectively, it must be consistent in a suitable sense. This "suitable sense" is not just the logical consistency of the pure theory. Because there are no axioms asserting the existence of any objects, activities, or activity occurrences, the entire theory is satisfied by the one-element model containing a single time point [trivial.model]. Although PSL 2010 without individual constants is consistent, this observation does not preclude the possibility that the theory becomes inconsistent as soon as one adds constants for specific activities, objects, and activity occurrences. If we add some constants (or, equivalently, add additional existential axioms for PSL predicates), then if all is well, we should expect the theory with those constants to have a "ground model", in which the elements of the model correspond to objects "constructed from the constants by the axioms", i.e. given by the values of terms built up from the constants and Skolem functions.

The third author first reported [11] that this was not the case for an earlier version of PSL 2008. There were several errors in the axioms that resulted in inconsistency as soon as there are two activity occurrences! Specifically, introducing three constants and the axiom $min\_precedes(b, c, a)$, he found contradictions using Prover9. He traced these problems ("after a tedious debugging process") to three axioms, and he suggested changes to these axioms, which would prevent the specific inconsistencies he

```
(forall (?a1 ?a2)
    (if (and (subactivity ?a1 ?a2)
             (not (atomic ?a2))
             (not (= ?a1 ?a2)))
        (not (exists (?s)
                (and (activity_occurrence ?s)
                     (subtree ?s ?a2 ?a1)))))
```

Figure 2: Mayer's revisions to Axiom 11 of the Complex Activity Theory.

```
(forall (?s1 ?s2)
    (if (earlier ?s1 ?s2)
        (exists (?a ?s3)
                (and (arboreal ?s3)
                     (generator ?a)
                     (= ?s2 (successor ?a ?s3))))))
```

Figure 3: Revised *successor* Axiom 11.

discovered. Although some of the problematic axioms have been revised in PSL 2010, not all sources of inconsistency have been eliminated.

The first two authors conducted further experiments regarding the consistency of the PSL Outer Core. Unless stated otherwise, we used a simplified version of PSL 2010, where the time-point axioms and the *successor* axioms that require infinite models were removed, and Axiom 11 of the Complex Activity Tree was revised as shown in Figure 2. The input and output files are available at `http://dl.dropbox.com/u/20534396/papers/ATE2011-PSL/index.html` in Prover9, Mace4 or Paradox3 format. The resulting subtheory of PSL admits the construction of a finite non-trivial ground model. The theory and a Mace4 model file can be found at [`PSL_work.in`] and [`PSL_work.model`]. However, even with those revisions, we will show that PSL is still inconsistent when a few existential axioms or individual constants are added to the theory.

We also found PSL 2010 to be inconsistent with some very simple assumptions about two activity occurrences. Specifically, we assumed a scenario where the occurrence a is immediately followed by an occurrence b of activity B within the occurrence ab of a complex activity AB [`PSL.contra1.in`]:

$$activity(B) \wedge subactivity(B,AB) \wedge$$
$$activity\_occurrence(b) \wedge occurrence\_of(b,B) \wedge activity\_occurrence(a) \wedge occurrence\_of(a,A) \wedge$$
$$root\_occ(a,ab) \wedge occurrence\_of(ab,AB) \wedge earlier(a,b) \wedge next\_subocc(a,b,AB)$$

In the proof of inconsistency [`PSL.contra1.proof`], Axiom 11 of the Complex Activity Theory indeed fails because of the superfluous *not* and the missing hypothesis as described below. However, we have difficulties in understanding the intended meanings of both Axiom 8, which formalizes properties of root occurrences of atomic activities, and Axiom 11, and hence we cannot say definitely whether the proposed changes discussed below will resolve the problem in general.

We propose revisions to the theory that will eliminate the specific inconsistencies that we have seen. The revised PSL Outer Core theory in Prover9 format is available at [`PSL_revised.in`]. We have shown that this theory has non-trivial finite models if the axioms requiring infinite models are omitted [`PSL_revised_finite.in`][`PSL_revised_finite.model`]. However, we cannot claim that these revisions will eliminate all inconsistencies that may arise when posing different ground assumptions.

3.1 Complex Activity Theory Axiom 11

Prior to our modifications, Axiom 11 of the Theory of Complex Activities, which relates the activity tree of a complex occurrence to the subtrees corresponding to its subactivities, still has an incorrect negation and needed an extra hypothesis. Figure 2 shows the axiom and the proposed revisions. The formulas in Figure 2 and subsequent figures are written in KIF notation, which uses question marks to indicate variables and LISP-style parentheses. For example, $\exists x P(x)$ would be written as (exists (?x) (P ?x)).

Intuitively, the inconsistency arises because *subactivity* is reflexive (and can be used with atomic activities), but *subtree* is not (and its second argument must be a complex activity).

```
(forall (?a1 ?a2)
    (iff (subtree ?s1 ?a1 ?a2)
         (and (root ?s1 ?a1)
              (exists (?s2)
                  (and (root ?s2 ?a2)
                       (or (= ?s1 ?s2)  (min_precedes ?s1 ?s2 ?a1))
                       (forall (?s3)
                           (if (min_precedes ?s2 ?s3 ?a2)
                               (min_precedes ?s2 ?s3 ?a1)))))))))
```

Figure 4: Revised *subtree* definition.

3.2 Complex Activity Theory Definition 4: Subtree

The contradiction found in the previous section can in part be attributed to the definition of $subtree(o,a_1,a_2)$. Contrary to most other predicates named *sub_* in PSL, *subtree* is irreflexive in the sense that $(\forall o,a)\neg subtree(o,a,a)$. This mix of reflexive and irreflexive predicates can easily lead to confusion, and missing hypotheses, as demonstrated earlier.

With the current definition one cannot establish that a concrete subtree actually exists in a given ground model. For example, if one asserts that A is a subactivity of AB, which is in turn a subactivity of ABC, and that occurrence a_1 of A is a root of AB and ABC, then $subtree(a_1,ABC,AB)$ is not entailed by PSL because $(\forall x,a)\neg min\_precedes(x,x,a)$ is a theorem of PSL.

Furthermore, the axiom is not strong enough to ensure that the subtree is indeed completely embedded into the activity tree of the superactivity. PSL has a model where both trees overlap at the root r of the subtree, yet not all occurrences in the subtree are also in the supertree: $(\exists x,a_1,a_2,r,y)subtree(x,a_1,a_2) \wedge min\_precedes(x,r,a_1) \wedge min\_precedes(r,y,a_2) \wedge \neg min\_precedes(x,y,a_1)$ is satisfiable [`subtree_overlap.tptp`][`subtree_overlap.model`]. This is at best counterintuitive.

We propose to revise this axiom and base the subtree property on the relationship between the *min_precedes* ordering in the sub- and supertree. Our revisions ensure that the activity tree of the superactivity embeds the tree of the subactivity. Figure 4 shows the new axiom. Our definition is reflexive such that $(\exists x,a)subtree(x,a,a)$ is satisfiable in PSL. The new *subtree* definition allows us to omit the extra hypotheses introduced earlier from Figure 2; the negation must still be removed.

However, even with these corrections, inspection of models of simple occurrence trees such as the overlap model above reveals that unintended elements are introduced into the models by PSL's *initial* predicate, which represents the first occurrence of an activity. This points to another critical incompleteness in PSL. *Generator* activities are those which have initial occurrences in the occurrence tree: $(\forall a)generator(a) \rightarrow (\exists s)initial(s) \wedge occurrence\_of(s,a)$. Introduction of these initial occurrences is not explicitly constrained in PSL by any definition, and is only loosely constrained by *earlier* relations and by the undefined PSL *successor* function. Moreover, the *earlier* relation, which orders primitive and complex occurrences in branches of the occurrence tree, seems unable to consistently express the notions of initial and final activity occurrences during finite temporal intervals, or of possibly unique occurrences, like the assassination of Abraham Lincoln. We return to this in discussing Finite Tree PSL.

3.3 Complex Activity Theory Definition 5: Sibling

PSL informally defines two subactivity occurrences s_1 and s_2 of a complex activity occurrence A to be *siblings* iff s_1 and s_2 either have the same immediate predecessor in the activity tree for A, or if both s_1 and s_2 are roots of activity trees for A and both are immediate successors of the same occurrence in the Occurrence Tree. For example b_1 and c_2 are siblings within A in Figure 1b. Intuitively one would expect that $sibling(s_1,s_2,A)$ implies that $s_1 \neq s_2$. However, the axioms allow that $sibling(s_1,s_1,A)$ holds.

The subexpression formalizing the *successor* constraint is incorrect: it admits models where s_1 and

```
(forall (?s1 ?s2 ?a)
  (iff (sibling ?s1 ?s2 ?a)
       (and (not (= ?s1 ?s2))
         (or (exists (?s3)
               (and (next_subocc ?s3 ?s1 ?a)
                    (next_subocc ?s3 ?s2 ?a)))
             (and
                (root ?s1 ?a)
                (root ?s2 a)
                (or (and (initial ?s1) (initial ?s2))
                    (exists (?s4 ?a1 ?a2)
                        (and
                            (= ?s1 (successor ?a1 ?s4))
                            (= ?s2 (successor ?a2 ?s4))
                            (arboreal ?s4)
                            (occurrence_of ?s1 ?a1)
                            (occurrence_of ?s2 ?a2)))))))))
```

Figure 5: Revised *sibling* definition.

```
(forall (?s1 ?s2 ?a)
  (iff (iso_occ ?s1 ?s2 ?a)
       (exists (?a1 ?a2 ?a3)
         (and (atomic ?a1) (atomic ?a2)
              (atomic ?a3)
              (subactivity ?a3 ?a)
              (occurrence_of ?s1 (conc ?a1 ?a3))
              (occurrence_of ?s2 (conc ?a2 ?a3))
              (all (?a4)
                (if
                  (and
                    (subactivity ?a4 (conc ?a1 ?a3))
                    (subactivity ?a4 (conc ?a2 ?a3))
                    (subactivity ?a4 ?a))
                    (or (= ?a3 ?a4)
                        (not (subactivity ?a3 ?a4))))
              ))))))
```

Figure 6: Revised *iso_occ* definition.

s_2 are equal to $successor(x_i, y)$ where x_i and y are arbitrary terms and not necessarily activities and an activity occurrence, respectively. This stems from the axiomatization of the Occurrence Tree, where the arguments of the *successor* function are constrained only if the entire term $successor(x, y)$ is known to be an occurrence of activity x. Therefore, by choosing x to be a term that does not represent an activity, the existential clause can always be satisfied. Figure 5 presents our attempt to resolve both problems.

3.4 Complex Activity Occurrence Theory Definition 1: iso_occ

Two occurrences s_1 and s_2 are said to be "occurrence isomorphic" with respect to a complex activity A if both occurrences contain a common subactivity of A. The axiom has recently been revised to restrict the common subactivity to be "maximal" (such that no common superactivity exists), yet the formalization implies a contradiction with the remaining axioms of PSL if the existence of two occurrence isomorphic activities is asserted [`iso_occ.contra.in`][`iso_occ.contra.proof`]. The deficiency in the formalization stems from overlooking that *subactivity* is reflexive. Figure 6 shows the repaired axiom. We are unsure whether the predicate is intended to be reflexive. If the relation should be irreflexive, an additional clause demanding that $s_1 \neq s_2$ must be added to the definiens. We have not verified if this extra assumption can be consistently made.

3.5 Complex Activity Occurrence Theory Definition 5: same_grove

Two occurrences of a complex activity are said to be in the "same grove" if they are in alternative branches in the Occurrence Tree. More formally, occurrences o_1 and o_2 of activity A satisfy $same\_grove(o_1, o_2, A)$ iff their root occurrences are siblings in the Occurrence Tree. The formal axiom does not make use of the *sibling* predicate defined earlier but duplicates part of it, including the problematic existential quantification discussed earlier. The axiom implies that either all complex activities share a common root occurrence, or no root of a complex occurrence may be a successor in the activity tree [`same_grove.in`][`same_grove.cont.proof`]. We propose to correct this problem by using the *sibling* predicate instead (see Figure 7). Curiously, this definition is almost identical to that in an earlier version of PSL presented by Bock and Grüninger [5].

4 Incompleteness of PSL Outer Core

Although PSL includes axioms that are too strong and results in inconsistency, certain axioms needed to exclude undesirable models are absent or not strong enough. In the following we identify some of these axioms. Because one cannot efficiently enumerate the possible models for given assumptions, we are far from claiming that our analysis has successfully identified all problematic axioms.

```
(forall (?o1 ?o2)
  (iff (same_grove ?o1 ?o2)
       (exists (?a ?s1 ?s2)
         (and (occurrence_of ?o1 ?a)
              (occurrence_of ?o2 ?a)
              (root_occ ?s1 ?o1)
              (root_occ ?s2 ?o2)
              (or (= ?s1 ?s2)
                  (sibling ?s1 ?s2 ?a))))))
```

Figure 7: Revised *same_grove* definition.

```
(forall (?a ?b0 ?b1)
        (if (and (atomic ?a)
                 (atomic ?b0)
                 (atomic ?b1)
                 (subactivity ?a (conc ?b0 ?b1))
                 (not (primitive ?a)))
            (exists (?a0 ?a1)
                    (and (subactivity ?a0 ~~?a~~ ?b0)
                         (subactivity ?a1 ~~?a~~ ?b1)
                         (= ?a (conc ?a0 ?a1))))))
```

Figure 8: Revised distributivity axiom.

4.1 Atomic Activity Theory Axiom 8: Distributivity

As pointed out by Mayer [11], distributivity of the lattice of atomic activities is formalized incorrectly, in that the axiom is actually a tautology. Therefore, it will eliminate models that do not satisfy this property. The correct formalization of distributivity is given in Figure 8.

4.2 Occurrence Tree Axiom 11: successor

Axiom 11 of the Occurrence Tree Theory should express that every non-initial activity occurrence is the successor of another activity occurrence. Unfortunately, the formalization suffers from the same problem as described in Section 3.3, in that the arguments of the *successor* function are not sufficiently constrained. Our solution is to amend the axiom as shown in Figure 3. The definition of the *poss* predicate that represents the legal successor activities in the occurrence tree requires similar amendments.

5 Observations from Experiments

5.1 Scalability

The axiomatization of PSL is not particularly amenable to applying automated theorem proving technology. We have already pointed out that constructing models is hindered by having only infinite models. Although removing some of the axioms yields a theory that has finite models, findings such models remains difficult. Even with state of the art model finders like Mace4 and Paradox3, we could not consistently find models of 13 or more elements (within generous limits of 4GB memory and 12h CPU time). Furthermore, the resulting models may not satisfy all axioms of full PSL (for example, certain tree properties) and must be inspected and/or post-processed to ensure the results are meaningful. Finding proofs of contradiction is equally challenging, due to the large number of clauses generated from Skolem functions. We expect that considerable tuning and adaptation of PSL will be necessary if the theory is to be used for reasoning about non-trivial processes.

5.2 Compositionality

The treatment of composite activities, such as concurrent aggregation and complex activities, is not transparent in PSL and must be anticipated by the modeler.

Occurrences of concurrent activities must be considered explicitly when defining complex activities. As both the Activity Tree and the Occurrence Tree specify an ordering between *atomic* activities, and PSL distinguishes *conc* activities from *primitive* (i.e., non-concurrent) activities, concurrent occurrences must be allowed explicitly in the specification of complex activities. For example, asserting that $\neg((\exists s_1, s_2) occurrence\_of(s_1, A) \wedge occurrence\_of(s_2, B) \wedge min\_precedes(s_2, s_1, NotBA))$ is insufficient to ensure that no occurrence of B is before an occurrence of A within complex activity

NotBA. Instead, one must write $\neg((\exists s_1, s_2, x, y)atomic(x) \wedge atomic(y) \wedge occurrence\_of(s_1, conc(A,x)) \wedge occurrence\_of(s_2, conc(B,y)) \wedge min\_precedes(s_2, s_1, NotBA))$.

Similarly, one must anticipate which activities will eventually be expressed as complex activities. For example, PSL precludes us from writing

$$(\forall x, y, z)occurrence\_of(x,A) \wedge occurrence\_of(y,B) \wedge occurrence\_of(z,AB) \\ \wedge subactivity\_occurrence(x,z) \wedge subactivity\_occurrence(y,z) \rightarrow min\_precedes(x,y,AB)$$

to express that occurrences of A must precede occurrences of B within each occurrence of AB if A or B are complex activities. Instead, we must amend the antecedent with $leaf\_occ(u,x) \wedge root\_occ(v,y)$ and substitute u for x and v for y in the consequent. Furthermore, mixing complex activities with concurrent ones is not admissible in PSL. Writing such explicit specifications is error prone, and the resulting axioms may be difficult to read.

5.3 Complex Activity Occurrences and Fluents

PSL is tailored to expressing constraints on trees in order to define the intended process models for a particular application domain. Although the axioms allow one to impose constraints on complex activities to eliminate impossible occurrences, the axioms are not complete enough to prove that a complex activity actually occurred given a sequence of atomic activity occurrences that conform to the complex activity. This is for a number of reasons. First, PSL does not include *closure axioms* that state that no other activities and occurrences than those in the activity trees occur in a complex activity [5]. Second, the axioms do not entail that a complex activity occurred even if all the occurrences in a branch of an activity tree occurred. These must be added when the ontology is tailored to an application. We have not verified that this can be done consistently.

Moreover, the modeling of objects and state in PSL is incomplete. The first and second authors observed that the axioms pertaining to fluents and their propagation in the Occurrence Tree are not sufficient to handle composite expressions like conjunction and implication. It seems that a clear distinction between fluents and other objects in combination with a powerful *reflection principle* is needed to tackle these challenges. Detailed treatment of such extensions is beyond the scope of this paper.

5.4 Concurrency Model

PSL uses atomic activities formed from the *conc* constructor to represent the aggregate of multiple activities occurring concurrently. For example, $conc(a_1, a_2)$ represents an activity where subactivities a_1 and a_2 are executed concurrently. Occurrences of such aggregate activities can be part of the Occurrence Tree. This model can express some form of concurrency, however, it is limited in that the concurrent activities must have the same duration (because only the occurrence of the aggregate activity in the Occurrence Tree is associated with begin and end timestamps). Activities that run concurrently with a sequence of other activity occurrences cannot be expressed easily, and multiple concurrent instances of the same activity cannot be expressed [5].

PSL Outer Core lacks axioms that relate the properties of the individual activities to that of the concurrent aggregate activity. Suitable axioms must be added when activities are defined, and possible interferences between activities must be anticipated.

6 Finite tree and minimal PSL

The PSL documentation does not directly address questions arising from the differences between modeling finite and infinite occurrence trees, which is a distinction that cannot be captured in FOL; the compactness

theorem fails for the class of finite models of trees, and the problem of FOL validity in the class of all finite models is undecidable [12]. In fact, the class of valid FOL sentences over finite models is not recursively enumerable but is co-recursively enumerable. Therefore, any useful separation of infinite from finite trees will be impossible in the current PSL FOL theory. However, it is unclear what that augmentation might be. Finite model theory is an active field with many open questions, including some bearing directly upon the infamous P$\neq$NP problem. Regular tree languages definable in *FO* and in *FOmod* (a language with counting quantifiers) are discussed by Benedikt and Segoufin [4]. Gradel et al. [8] discuss the more general problem of embedding finite models into "representations" of infinite databases. History and classical results about finite and infinite models are given in Baldwin [3].

After being informed of contradictions in PSL found the authors, Conrad Bock and Michael Grüninger defined a "minimal PSL", which they hoped would be free of contradictions. This is a collection of PSL axioms taken from the PSL Outer Core subtheories (specifically, from pslcore.th, complex.th, occtree.th and actocc.th), and some "lemmas" which are theorems of PSL. All axioms that would require infinite models are omitted. This subset can be found in [`psl-1016-618.in`]. We could verify using Mace4 that this subset does have finite non-trivial models where some activity occurrences exist. Unfortunately, this minimal version of PSL lacks substantive axioms bearing upon tree constructions. We came to the conclusion that, while the Occurrence Tree is fundamental to PSL's ontology, PSL itself does not contain enough axioms to reason about finite trees well (let alone to reason about infinite occurrence trees).

One possible way to improve PSL is to modify it so that occurrence and activity trees will be finite, and to supplement its other theories with appropriate axioms defined from the new tree primitive. If we modify PSL so that it has finite models, then occurrence and activity trees will be finite trees, which are more tractable than the infinite occurrence trees of PSL. A good theory of finite trees, which has the needed definitions, is presented in Backofen et al. [2]. This theory of finite trees has attractive properties: it is decidable and it permits the use of inductive schema.

We have translated the tree axioms of [2] into the language of PSL, calling it *FTPSL* and using *earlier* for the tree ordering. It was necessary to reverse the order of some of the FTPSL predicates as axiomatized by Backofen et al. in order to comply with the directionality of PSL axioms regarding the notion of *subactivity_occurrence*. Besides *earlier*, the primitives of FTPSL are the binary predicates *subactivity_occurrence*, *proper_subactivity_occurrence* and *immediate_subactivity_occurrence*. The translated FTPSL axioms are available at `FTPSLRev1.in`, and a model is at `FTPSLRev1.model`.

We are able to demonstrate, using Mace4, that FTPSL plus simple ground axioms constructs an appropriate occurrence tree for the seven occurrences of example MGM7, has finite models and is therefore consistent [`FTPSLRev1_MGM7.in`][`FTPSLRev1_MGM7.model`]. Adding the transcribed finite tree axioms to Bock and Grüninger's minimal PSL, we were able to find a finite model for that theory and establish consistency [`FTPSLRev1OCCACTV1_PSL1016Lemmas.in`] [`FTPSLRev1OCCACTV1_PSL1016Lemmas.model`]. However, inspection of the model reveals that activities and occurrences are not always desirably coordinated due to a lack of proper typing in the minimal PSL. We have not attempted to correct this, but by using FTPSL as a basis for defining other PSL notions, and by extending the theory to include trees for activities as well as occurrences, it may be possible to define the compositionality of successor axioms for each activity and for their occurrences (by defining, for each *act*, $successor(act, occ) = occ_1$). We have not yet investigated whether this approach can be extended to encompass an adequate treatment of fluents and the *holds* predicate as well.

7 Conclusions

We set out to test the hypotheses that (i) PSL is adequate for industrial process definition, and (ii) existing theorem provers are adequate to support reasoning about process descriptions in PSL. We found that PSL

is not adequate for industrial process description, for the following reasons:

- PSL contained, and after correction still contains, inconsistencies when simple ground axioms are added. This precludes the use of model finders and theorem provers.
- PSL lacks a good theory of finite trees, although trees are fundamental to PSL.
- PSL lacks a reflection principle permitting ordinary logical reasoning which coordinates activities and their occurrences, and hence does not allow one to infer what properties will hold after complex activities occur.
- PSL is not formulated in a way "friendly" to automated deduction; specifically with respect to its preference for the use of predicates instead of functions, insufficient "typing" axioms and no clear distinction between fluents and physical objects.

Our experiments with automated theorem provers helped us to discover these defects of PSL, and showed that the combination of PSL and existing theorem-proving technology is not adequate for industrial-strength process engineering. Certainly we saw many cases in which theorem provers could not reach the desired conclusion; some of those cases were due in our opinion to the inadequacies of PSL. Clearly, the problems which may lie with theorem proving technology regarding PSL (particularly scaling) cannot be assessed without first having a consistent process specification language, and in particular, a substantive theory of finite trees which represents compositionality of complex occurrences.

In our opinion, the reason for the enumerated troubles is that PSL does not have a clear "intended model". However, an effective revision to the theory must *start* with a clear informal description of such a model, and then include axioms that are true in that model and characterize it, so that the ground model of the theory plus simple axioms describing an activity should be unique, except possibly for the order of activity occurrences when those do not matter. Unfortunately, these goals cannot be accomplished by simple "surgical modifications" or additions to PSL, because the present version of PSL is incoherent, as we have demonstrated by finding inconsistencies for which there is no obvious repair. Our revisions in this paper "put some duct tape on PSL", but it is still broken.

Further issues remain to be addressed before any PSL or any finite version thereof can be considered ready for industrial use. Foremost, the consistency of the theory must be established for the possible ground models of interest. While we have shown that our small ground examples have models, formal analysis and verification must be carried out to lift this result to all "admissible" ground extensions of the theory. Further documentation is necessary to make the theory more accessible. We have spent many weeks trying to understand the intended meanings of the axioms, yet considerable uncertainties remain. A thorough discussion and examples showing the best practice use of the ontology would certainly help. Furthermore, a collection of Competency Questions [9] and other sentences that are supposed to be consistent with or entailed by the theory, as well as ones that should *not* be consistent or provable, would help in validating any changes made in the future. Currently, tailoring the theory to a particular application domain is a predominantly manual activity. Automated support in writing and validating additional axioms would certainly facilitate the adoption of PSL. Similarly, tailoring the ontology to suit a specific theorem proving technology would immensely benefit from automated supporting tools to analyze performance bottlenecks, rewrite axioms, and verify consistency and completeness of the resulting theory.

References

[1] TC 184/SC 4. *Process Specification Language*. ISO, Geneva, Switzerland, 2004.

[2] R. Backofen, J. Rogers, and K. Vijay-Shankar. A first-order axiomatization of the theory of finite trees. Technical Report IRCS-95-02, Institute for Research in Cognitive Science, University of Pennsylvania, 1995.

[3] J. Baldwin. Finite and infinite model theory — a historical perspective. *Logic Journal of IGPL*, 8(5):605–628, 2000.

[4] M. Benedikt, and L. Segoufin. Regular tree languages definable in FO and in FO mod. *ACM Transactions on Computational Logic (TOCL)*, 11(1):4, 2009.

[5] C. Bock, and M. Grüninger. PSL: A semantic domain for flow models. *Software and Systems Modeling*, 4(2):209–231, 2005.

[6] C. Bock, and M. Grüninger. PSL. `http://www.mel.nist.gov/psl/download.html`.

[7] C. Bock, and M. Grüninger. PSL outer core 2010. `http://sonic.net/~halcomb/papers/psl_outer_core.clf`. Translated to Prover9 automatically by R. Schulz: `http://sonic.net/~halcomb/papers/psl_outer_core-2010-axms-defs.in`.

[8] E. Gradel, P.G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M.Y. Vardi, Y. Venema, and S. Weinstein. Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series), 2005.

[9] M. Katsumi, and M. Grüninger. Theorem proving in the ontology lifecycle. In *Knowledge Engineering and Ontology Design*, Valencia, Spain, 2010.

[10] H. J. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Electron. Trans. Artif. Intell.*, 2:159–178, 1998.

[11] W. Mayer. On the consistency of the PSL outer core ontology. Technical Report ACRC-KSE-080201, ACRC, University of South Australia, Mawson Lakes, Adelaide, Australia, Feb 2008. Circulated by email.

[12] B. Trahtenbrot. The impossibility of an algorithm for the decision problem for finite domains. In *Doklady Academii Nauk SSSR*, volume 70, pages 569–572, 1950.