Inconsistencies in the Process Specification Language (PSL)

Michael Beeson, Wolfgang Mayer, and Jay Halcomb

February 4, 2011

1 Inconsistencies in the PSL outer core

PSL consists of an "outer core" theory (containing about ninety axioms), plus about fourteen "extensions" and more than 60 "definitional extensions". We worked with the PSL outer core, ignoring the extensions, since (a) we thought that the outer core should be adequate for formalizing some simple examples, and (b) it was used in the examples given us by Conrad Bock of NIST, and (c) we ran into plenty of trouble using the outer core, without thinking about the extensions.

The core (the first 20 or so axioms of the outer core) says that everything is either an activity, activity occurrence, timepoint, or object, and is not more than one of these things, and contains a few fundamental axioms about these things. It also introduces the (irreflexive) relation *before* between timepoints and asserts the existence of a first timepoint *inf*− and a last timepoint *inf*+, and says that between any timepoint and *inf*+ there is another timepoint. This axiom means that PSL has no finite models, except those in which there is only one timepoint. (Strangely, there is no axiom $inf+ \neq inf-$.)

To ensure reproducibility, it is important to specify what version of PSL is used in experiments. As of July, 2010, the most recent version posted on the NIST website [**?**] is the May 2008 revision. We will refer to that version as PSL2008. We worked with a revision we call PSL2010, supplied to us by Conrad Bock of NIST [**?**, **?**].

In order for PSL to be used effectively, there must be no built-in contradictions, i.e. PSL 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, so the entire theory is satisfied by the one-element model containing a single timepoint. Although one can verify this by hand, Mayer has confirmed it by computer, using the Paradox model finder.

However, this observation does not preclude the possibility that the theory becomes inconsistent as soon as one adds some constants for specific activities, objects, and activity occurrences. If we add some constants, 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. Mayer [**?**] found that this was not the case; there were several errors in the axioms that resulted in inconsistency as soon as there are two activity occurrences! Specifically, as soon as one introduces three constants and the axiom $min\_precedes(b, c, a)$, Mayer found contradictions using Prover9.

Mayer 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 discovered. Specifically:

1. Axiom 7 of the Theory of Atomic Activities needed an extra hypothesis $a_1 \neq a_2$.
2. Axiom 11 of the Theory of Complex Activities had an incorrect negation and needed an extra hypothesis.
3. Axiom 6 of the Theory of Activity Occurrences, for which there is no obvious correction.

Regarding this last axiom, Mayer comments:

> Unfortunately, our understanding of the complex interplay between PSL axioms is not complete enough to determine how this problem should be resolved. Our repair is to weaken the axiom such that all complex subactivity occurrences must form connected regions on a branch of the activity tree.

However, there are examples in [**?**] that do not satisfy this condition, so presumably this is at best a temporary repair.

Of the changes proposed by Mayer (in January 2008) to fix these problems, only the first one (Axiom 7 of the Theory of Atomic Activities) was made in the May 2008 revision to the PSL posted at NIST. Hence, PSL2008 is (still) inconsistent. What about PSL2010? The fix for (1), made in PSL2008, is still there. The fix for (2), not made in PSL2008, is still not made in PSL2010. As for (3), that axiom has been drastically revised in PSL2010.

Mayer also found other difficulties; for example, Axiom 8 of the Theory of Atomic Activities is actually a tautology, and would need changes to correctly express the comment. This change has also not been made in PSL2008 and not in PSL2010, either. Mayer proposed a revision of Axiom 3 of the Theory of Activity Occurrences; in PSL2008 and PSL2010, that axiom has indeed been revised, but not in the way Mayer proposed.

After making these changes, Mayer also eliminated the time-point axioms and the successor axioms that require infinite models, producing a subtheory of PSL that does admit the construction of a finite ground model in the three-constant case described above.

We also conducted experiments regarding the consistency of the PSL outer core. We found PSL2010 Outer Core to be inconsistent with some very simple assumptions about two activity occurrences. Specifically, we assumed:

```
activity(B).
activity_occurrence(OCC_A).
activity_occurrence(OCC_B).
earlier(OCC_A,OCC_B).
occurrence_of(OCC_B,B).
occurrence_of(ROOT_OCC,ROOT_ACT).
root_occ(OCC_A,ROOT_OCC).
subactivity(B,ROOT_ACT).
next_subocc(OCC_A,OCC_B,ROOT_ACT).
```

After finding the proof of contradiction, we extracted from the proof the axioms of PSL Outer Core that are actually used in the proof. There are only 15 of them. The proof is 71 steps long, but that includes listing the axioms and clausifying them, so it is really not a very long proof. If we delete the unused axioms, then Prover9 finds the proof instantly: see the proof file NIST-psl_outer_core1.contra1.proof. The reader can check the input file at NIST-psl_outer_core1.contra1.in.

This was how we learned that Axiom 11 in the Theory of Complex Activities has not been revised, since this axiom, one of the causes of Mayer's inconsistencies, is used in our proof of inconsistency as well.

The situation as of December, 2010, is as follows: PSL2008 and PSL2010 are inconsistent with simple ground axioms that should be consistent with a correct version of PSL. The known contradictions, however, all involve Axiom 11 of the Theory of Complex Activities; if that axiom is corrected as suggested by Mayer, it is then unknown whether the resulting theory is or is not consistent with simple ground axioms.

In our opinion, a correctly formulated version of PSL should have intuitively comprehensible ground models, built from constants for distinct activities and activity occurrences. According to our intuition, with two atomic activities A and B, both subactivities of one root activity `ROOT_ACT`, and one occurrence (specified by constant symbols `OCC_A`, `OCC_B`, and `ROOT_OCC`) of each of the three activities, the ground model should consist of terms for activity occurrences built up by the binary function symbol *successor* from the constants; the predicate $earlier(x, y)$ is interpreted as the transitive closure of the relation $\exists z(successor(z, x) = y)$. We interpret $legal(x)$ to mean that x is an occurrence of A or of B. There would be only three activities, given by the three constants mentioned above, and a countable infinity of timepoints, given by terms involving the function symbols *beginOf* and *endOf*. It seems that the only objects required to exist are subtrees of the occurrence tree. In this model, $min\_precedes(x, y, \texttt{ROOT\_ACT})$ holds only if $earlier(x, y)$.

In addition, Axiom 11 of the Theory of Complex Activities, which was discussed above in connection with the Mayer inconsistencies, fails in our model because of the *not* that Mayer said should be removed and the hypothesis he says is missing.

Perhaps the contradiction is due to errors in the formulation of Axiom 11, but we have

difficulties in understanding the intended meanings of both Axiom 8 and Axiom 11, and hence we cannot say definitely whether the problems can be fixed by minor changes to the axioms. What is missing is a proof of consistency of PSL with simple axioms such as those above 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.

After being informed of contradictions in PSL found by Mayer and by us, Bock and Gruninger defined a "minimal PSL", which they hoped would be free of contradictions. This is a collection of PSL axioms taken from the PSL sub theories (pslcore, complex, occtree and actocc), and some "lemmas" which are theorems of PSL. All axioms that would require infinite models are omitted. We could verify using Mace4 that this subset plus axioms that there exist some activity occurrences does have finite models. This subset can be found in the file psl-1016-subset-618-lemmas-flat.in.

Meantime, we came to the conclusion that, while the occurrence tree is fundamental to PSL's ontology, PSL does not contain enough axioms to reason about finite trees well (let alone reason about the infinite occurrence tree).

We also formulated appropriate axioms about finite trees, expressed in the language of PSL, as described below in a section on Finite Tree PSL (FTPSL). Adding the finite tree axioms to Bock and Gruninger's minimal PSL, we were able to find a finite model, establishing consistency of this subset.

The semantics of PSL are formulated using the notions and vocabulary of trees. These are activity trees and occurrence trees, with activity trees intended to be represented as subsets of occurrence trees. However, as it stands, no subset of PSL (PSL2008, PSL2010, or the minimal subset of PSL) can consistently demonstrate the existence of any nontrivial occurrence tree, as such. This is because PSL lacks valid and logically proper definitions for the notions of siblings of complex occurrences and ancestors of complex occurrences (same_grove). Hence, in particular PSL cannot consistently represent the compositionality of occurrences - which is a fundamental flaw.

In order to represent compositionality of occurrences, PSL must allow the construction of finite occurrence trees representing occurrences of both primitive and complex activities. While the occurrence of complex activities potentially might be represented as subsets (not simply as individual nodes of occurrences of primitive activities) of an occurrence tree, PSL is unable to do this, due to its current inconsistencies.

Further, to be usable PSL must be able to (consistently) construct a temporal ('earlier'), finite total ordering of the primitive occurrences of a single complex activity. The domain of this ordering should include (in the extension of a total order, 'earlier') occurrences of both complex and primitive activities. FTPSL is able to do this, while PSL is not for reasons we discuss next.

In a semantical vein, PSL has definitions for subtree, for isomorphic occurrences of an activity, for sibling occurrences of an activity, and for 'same grove' occurrences. We will take up each in the next section. These definitions are:

```
all s1 all s2 all a (sibling(s1, s2, a) <->
exists s3 (next_subocc(s3, s1, a)  & next_subocc(s3, s2, a))
| root(s1, a) & root(s2, a) & (initial(s1) & initial(s2)
| exists s4 exists a1 exists a2 (s1 = successor(a1, s4)
& s2 = successor(a2, s4)))) # label("complex:Def.05").

all s1 all a1 all a2 (subtree(s1, a1, a2) <-> root(s1, a1) &
exists s2 exists s3 (root(s2, a2) & min_precedes(s1, s2, a1)
& min_precedes(s1, s3, a1) & -min_precedes(s2, s3, a2)))
# label("complex:Def.04").

all occ1 all occ2 all s1 all s2 (same_grove(occ1, occ2) <->
exists a (occurrence_of(occ1, a) & occurrence_of(occ2, a)
& root_occ(s1, occ1) & root_occ(s2, occ2) & (initial(s1)
& initial(s2) | exists s4 exists a1 exists a2 (s1 = successor(a1, s4)
& s2 = successor(a2, s4))))) # label("actocc:Def.05").

all s1 all s2 all a (iso_occ(s1, s2, a) <->
exists a1 exists a2 exists a3 (atomic(a1) & atomic(a2) & atomic(a3)
& subactivity(a3, a)  & occurrence_of(s1, conc(a1, a3))
& occurrence_of(s2, conc(a2, a3))
& all a4 (subactivity(a4, conc(a3, a1)) & subactivity(a4, conc(a3, a2))
& subactivity(a4, a) ->  -subactivity(a3, a4))))  # label("actocc:Def.01").
```

1.1 Subtree contradiction (Mayer) and problematic definition

For this work we used Mayer's proposed revisions to PSL. See PSL 2010 MayerRev.in. Without any additional existential assumptions, this theory produces a two element model in Mace4; see PSL 2010 MayerRev.model.

Mayer discovered a contradiction in PSL involving the notion of subtree, Complex Ax 11, and Complex Definition 4. when an activity occurrence is the root occurrence of two different activities. Mayer noted that under simple ground assumptions a contradiction involving the notion of 'subtree' results from Complex:Ax11 and the definition of subtree. Mayer proposed to fix this by revising Complex:Ax11. Using Mayer's revisions a contradiction no longer results in PSL2010 from instantiating the definition of subtree. However, even with Mayer's revision it cannot be proven that some subtrees always exist. Mayer also noted that the definition of 'subtree' is not reflexive in PSL. This is suspicious and calls into question the PSL notion of tree.

"Definition 4. The activity tree for a1 with root occurrence s1 contains an activity tree

for a2 as a subtree if and only if every atomic subactivity occurrence in the activity tree for a2 is an element of the activity tree for a1, and there is an atomic subactivity occurrence in the activity tree for a1 that is not in the activity tree for a2." (NIST PSL website).

```
all s1 all  a1 all  a2 (subtree( s1, a1, a2)  <->
root( s1,  a1) & exists  s2 exists  s3 (root( s2,  a2)
& min precedes( s1, s2, a1)
& min precedes( s1,  s3,  a1)
& -min precedes( s2, s3, a2)))    # label("complex:Def.04").
```

The logical meaning of the informal text is obscure and does not seem to express the formalization. In particular, the formal axiom makes no mention of atomicity. To test the subtree definition we created ground PSL assertions for 7 activities and 7 occurrences of those activities, called MGM7. This file may be viewed at: MGM7.in

Using MGM7, the subtree condition subtree(OCC_A,ACT_ABCD,ACT_AB) is unprovable using PSL2010MayerRev when OCC_A is specified to the root of both ACT_AB and ACT_ABCD. This is because `all x all a -(min_precedes(x,x,a))` is a theorem of PSL, as no occurrence of an activity is earlier than itself. Thus it is impossible simultaneously that `root(OCC_A, ACT_AB) & min_precedes(OCC_A,OCC_A,ACT_ABCD)`, as required.

In PSL 2010, even with Mayer's revisions, root occurrences of activities are not required to be unique in the sense that

```
all s1 all s2 all s3 all a all o (activity(a) & activity_occurrence(s3)
 & activity_occurrence(s2) & root(s2,a) & root(s3,a) -> s2 = s3))
  # label("root-unique:New").
```

This is the case although root_occs (as in `root_occ(occ1,occ2)`) are provably unique in PSL2010. While an additional assumption of the uniqueness of root occurrences of activities is consistent with PSL without ground terms (see PSL 2010 MayerRev roots unique.model), when this axiom is introduced with ground models such as MGM7, it leads to a contradiction (see PSL 2010 MayerRev MGM7 roots unique.proof).

If we assume that occurrences of roots of activities are unique, then PSL proves the assertions:

```
all x all y all act1 all act2  ((subtree(x,act1, act2)
& root(x,act1)  & root(y,act2)) -> -(x=y)).


all s1 all a1 all a2 ((subtree(s1, a1, a2)
& root(s1, a2) )->  - root(s1,a1)).
```

PSL has the axiom:

```
all s all a (root(s,a) ->
 -(exists s2 (activity_occurrence(s2) & min_precedes(s2,s,a)))))
# label("complex:Axm.05").
```

Adding the assumption `-(exists x min_precedes(x,OCC_A,ACT_ABCD))` leads to another contradiction. See PSL 2010 MayerRev MGM7 nominprecedes.proof

These considerations indicate a fundamental and unresolved ambiguity in PSL.

Earlier we remarked that to be usable PSL must be able to consistently construct a temporal ('earlier') finite total ordering of the primitive occurrences of a single complex activity. An axiom which expresses a necessary condition (the 'totality' of a total ordering) is:

```
(all a1 all a2 all a all x all y (atomic(a1) & atomic(a2) &
subactivity(a1,a) & subactivity(a2,a) & occurrence_of(x,a1)
& -(a1 = a2) & occurrence_of(y,a2) -> min_precedes(x,y,a)
| min_precedes(y,x,a))) # label("complex:NewAxm").
```

However, the addition of this axiom to PSL2010MayerRev yields a contradiction using MGM7. See PSL 2010 MayerRev MGM7 TOofAtoms.in and PSL 2010 MayerRev MGM7 TOofAtoms.proof.

1.2 same_grove contradiction

The definition of same_grove is not logically proper as two outermost quantifiers having occurrences in the definiens do not occur in the definiendum.

```
all  occ1 all occ2 all s1 all s2 (same_grove(occ1, occ2) <->
exists _a (occurrence_of(occ1, a) & occurrence_of(occ2, a)
& root_occ(s1, occ1) & root_occ(s2, occ2) & (initial(s1) &
initial(s2) | exists s4 exists a1 exists a2
(s1 = successor(a1, _s4) & s2 = successor(a2, s4)))))
# label("act_occ:Def.05").
```

See PSL 2010 MayerRev MGM3 same grove.in and PSL 2010 MayerRev MGM3 same grove.proof. However, when a correction is made (as below) by moving the offending quantifiers to the definiens, the corrected axiom quickly implies a contradiction under the assumption that same_grove occurrences exist.

```
all occ1 all occ2 (same_grove(occ1, occ2) <->
all s1 all s2 exists a (occurrence_of(occ1, a)
&  occurrence_of(occ2, a) & root_occ(s1, occ1) & root_occ(s2, occ2)
```

```
& (initial(s1) & initial(s2)  |
exists s4 exists a1 exists a2 (s1 = successor(a1, s4)
& s2 = successor(a2, s4)))))
# label("actocc:Def.05Rev").
```

See PSL 2010 MayerRev MGM3 same grove1.proof.

1.3 sibling contradiction

The definition of sibling implies a contradiction when instantiated.

```
(all  s1 all  s2 all  a (sibling( s1, s2, a) <->
(exists  s3 (next _subocc( s3, s1, a) &
next_subocc( s3, s2, a))) | root( s1, a) & root( s2, a)
& (initial( s1) & initial( s2) | (exists  s4 exists  a1 exists  a2
( s1 = successor( a1, s4) &  s2 = successor( a2, s4))))))
# label("complex:Def.05").
```

See: PSL 2010 MayerRev MGM3 sibling.in and PSL 2010 MayerRev MGM3 sibling.proof. Note that the following is a theorem of PSL2020 (and of minimal PSL):

```
all s1 all s2 all s3 all a all o (next_subocc(s1, s2, a)
& next_subocc(s1, s3, a)  & occurrence of(o, a)
& subactivity_occurrence(s3, o)  & subactivity_occurrence(s2, o)
-> s2 = s3)  # label("MIN:Lemma 5: nextsubocc-unique:").
```

1.4 iso_occ contradiction

Assuming the existence of isomorphic occurrences, the definition of iso_occ implies a contradiction, via the aggregation function 'conc' in PSL in the presence of the axioms for atomic.th, even as revised by Mayer.

```
all  s1 all  s2 all  a (iso _occ( s1,  s2,  a) <->
exists  a1 exists  a2 exists  a3 (atomic( a1)  & atomic( a2)
& atomic( a3)  & subactivity( a3,  a)  &
occurrence _of( s1, conc( a1,  a3))  &
occurrence _of( s2, conc( a2,  a3))  &
all  a4 (subactivity( a4, conc( a3,  a1))  &
subactivity( a4, conc( a3,  a2))  & subactivity( a4,  a) ->
-subactivity( a3,  a4))))  # label("act occ:Def.01").
```

See PSL 2010 MayerRev MGM3 iso occ.in and PSL 2010 MayerRev MGM3 iso occ.proof.

1.5 conc contradiction

The PSL theory of 'conc' (atomic.th) is inconsistent in PSL2010 despite Mayer's revisions. Note also that it can't distinguish the ordering of activities because there is no way to differentiate occurrences of 'conc(a1,a2)' from 'conc(a2,a1)' using earlier, since conc(a1,a2) is identical to conc(a2,a1).

See PSL 2010 MayerRev MGM3 conc.in and PSL 2010 MayerRev MGM3 conc.proof.

1.6 successor contradiction

PSL successor axioms imply contradictions, both in the presence of atomic.th and without it. For the former see PSL 2010 Simp MGM5 conc1 contra.in and PSL 2010 Simp MGM5 conc1 contra.proof.

For a contradiction without atomic.th but in the presence of the statement (exists x exists y exists z (activity(z) & activity_occurrence(x) & activity_occurrence(y) & arboreal(x) & arboreal(y) & occurrence_of(x,z) & occurrence_of(y,z) & sucessor(z,x) = y)), see: PSL_2010_simp_successor_contra2.in and PSL_2010_simp_successor_contra2.proof.

2 Fixing PSL

The plan of this paper was to test the two hypotheses (mentioned in the introduction) about PSL and about modern theorem-provers. Our tests exposed defects of PSL and of the combination of PSL with theorem-proving, forcing us to the negative conclusions stated in the last section. We naturally investigated whether these defects could be repaired. In this section we indicate some partial results along this line and some directions for further work.

2.1 Removing axioms leading to infinite models

When we wish to compare what happened "on the shop floor" with a process specification, we need to be able to find a proof (if what happened had the desired result), or a counter-model (to show that it did not necessarily have the desired result). We emphasize that one cannot hope to *prove* that the desired result was not forthcoming. Suppose, for example, that we botched the table-making operation somehow. Could we hope to prove that after the last operation there is no table? No! It might have appeared on the workbench *deus ex machina*. One cannot prove that a table cannot just appear suddenly, because there are no axioms to prevent such a thing. Instead, we want to be able to produce a *ground model*, in which only the reported activity occurrences took place, and show that, *in that model*,

there is no table. That is what we mean when we say that this sequence of operations did not (suffice to) produce a table.

It is therefore a fundamental defect of PSL that it has only infinite models. Moreover, there appears to be no very good reason for that. There are two ideas in PSL that lead to infinite models:

1. Between any two timepoints there is another timepoint.

2. For every activity occurrence, there are more activity occurrences (formed using the *successor* function symbol).

In other words, the occurrence tree contains all possible activity occurrences (technically it even contains some impossible ones!), rather than just those that actually occurred. If we limit the intended model to those activity occurrences that actually occurred, we should get a more tractable and more applicable theory.

2.2 Typing

The concepts that PSL attempts to formalize are naturally typed: there are timepoints, activities, activity occurrences, and some others that PSL lumps together as "objects". Specifically, the required objects include *fluents* (properties), parts, supplies, and possibly agents. Moreover, activities and activity occurrences can be complex, atomic, or primitive. PSL has been expressed in first-order logic, and uses unary predicates to partially express these types. This procedure is not conducive to efficient use of a theorem-prover. PSL would be more naturally expressed in a many-sorted first order logic. Such a logic has different "sorts" of variables; the word "type" from programming languages is not used unless there is a type of functions or sets of objects of a given type, which in PSL is not required.

Theorem-provers do not work directly with many-sorted logic, but that is not a serious limitation, because it can be handled by a technique known as "implicit typing." In this technique, each function symbol or predicate should have a unique "signature" or "prototype"; that means that the sorts of its arguments are specified, and in the case of a function symbol, the sort of its value. Then one can check that one can unambiguously delete unary predicates needed for typing only, and reconstruct them if desired. For example, if we want to quantify over all activity_occurrences, we just write $\forall x P(x)$ instead of $\forall x(activity\_occurrence(x) \rightarrow P(x))$. It can be shown that all the deductions made by a theorem-prover with the typing predicates removed will still be valid when the typing predicates are restored. This was apparently first observed by [**?**] and rediscovered by the first author, who then extended the result to a more complicated logic [**?**]. It is worth noting that for this method to work, one needs a theorem-prover that supports "multiple equality predicates", because one needs a different equality predicate for each sort, to fulfill the hypothesis that the predicate symbols have unique signatures. As far as we know only

Otter supports multiple equality symbols. We did not conduct experiments with implicitly-typed versions of PSL, since Otter is regarded by its author as replaced by Prover9, and Prover9 does not support multiple equality symbols. Nevertheless, implicit typing is a valuable technique; for example, there are many logic puzzles that involve, for example, five men living in five houses of five different colors, each having a different pet and a different favorite food, and you are given some ground facts about some of these items and asked to find who owns the iguana and lives in the green house and likes lasagna. One can do those problems with implicit typing and often one cannot do them without it; one drowns in a sea of mistyped terms.

Therefore we recommend that theorem-provers be modified to support multiple equality symbols, and PSL's successor should be expressed in many-sorted predicate calculus (with sorts as indicated above, i.e. more than the four sorts of PSL).

2.3 FTPSL: a theory of finite trees

The occurrence tree is the fundamental notion of PSL, yet PSL has few axioms bearing upon trees and those which it has are inconsistent with other PSL axioms, as we have shown above. If we modify PSL as indicated above so that it has finite models, then occurrence trees will be finite trees, which are more tractable than the infinite occurrence trees of PSL. One way to improve PSL, then, is to modify it so that occurrence trees will be finite, and supplement it by appropriate axioms about finite trees. A good theory of finite trees, which has the needed definitions, is presented in Backofen et al [**?**]. This theory of finite trees has an attractive property: it is decidable, and it uses inductive schema.

We translated the tree axioms of [**?**] 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 comport with the directionality of PSL axioms regarding the notion of subactivity_occurrence).

We are able to demonstrate, using Mace4, that FTPSL plus simple ground axioms constructs an appropriate occurrence tree for the seven occurrences of MGM7, has finite models, and is therefore consistent. See the files FTPSLRev1.model, FTPSLRev1 MGM7 occ.in, and FTPSLRev1 MGM7 occ.model.

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, succ(act,occ) = occ1)). Whether such an approach can be extended to encompass 'holds as well', we do not know.

2.4 Add some kind of reflection principle

One should be able to state purely logical facts, independent of activity occurrences, and then infer that those facts *hold* at arbitrary elements of the occurrence tree. For example,

the definition of $isTable(T)$ just says that T has the form $table(t, \ell_1, \ell_2, \ell_3, \ell_4))$ where t is a tabletop and the ℓ_k are legs, and t is attached to each ℓ_k. Yet, even if we know that after the last activity occurrence $s(4)$ each of those individual facts holds, we cannot conclude that $isTable(T)$ holds, because there is no reflection principle in PSL. To make the *MakeTable* example work, we had to formulate the definition of *isTable* using *holds*, as if what constitutes a table depended on when you ask. This is awkward, and that awkwardness would be compounded in more complex (and natural) examples.

2.5 Expand the scope of *holds*

PSL does not permit one to say that something holds after an occurrence of a compound activity. Nor does it permit one to say that a logically compound statement holds anywhere, even after an atomic activity. These are both serious defects if one wants to reason about industrial-strength processes.

3 Conclusions

We set out to test the hypotheses that

- PSL is adequate for industrial process definition.
- Existing theorem-provers are adequate to support reasoning about process descriptions in PSL.

We found that PSL (either in the ISO standard form or with all changes made so far by NIST, or with any changes of similar magnitude) is not adequate for industrial process description, for the following reasons:

- It contained and (after correction) still contains inconsistencies (with simple ground axioms added).
- Its ontology is not clear enough to permit the construction of ground models to demonstrate consistency with simple process descriptions.
- It does not permit one to state that complex properties will hold after complex activities are performed.
- It lacks a reflection principle connecting ordinary logical reasoning with assertions that facts "hold" after activity occurrences.
- It does not permit finite models with nontrivial activity occurrences, precluding the use of automated model-finders.
- It lacks a good theory of finite trees, although occurrence trees are fundamental to PSL.

- It is not formulated in a way "friendly" to automated deduction; specifically with respect to its preference for the use of predicates instead of functions, and with respect to its lack of enough "typing" axioms, and the lack of a distinction between fluents and physical objects.

Our experiments with automated theorem-provers helped us to discover the 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 theory of PSL, and in particular, a substantive theory of finite trees in PSL which represents compositionality of complex occurrences.