PSL as a Foundational Ontology for the Industrial Ontologies Foundry

Michael GRÜNINGER [a] and Megan KATSUMI [a]
[a] *Department of Mechanical & Industrial Engineering, University of Toronto, Canada*

Abstract. In today's global economy, manufacturing enterprises must employ increasingly effective and efficient information systems. Such systems should result in the seamless integration of manufacturing applications and exchange of manufacturing process information between applications. The Industrial Ontologies Foundry (IOF) seeks to create a set of core ontologies that spans the entire domain of digital manufacturing. In this paper we explore the adequacy for the Process Specification Language (PSL) to serve as the foundational ontology that can be used to axiomatize the ontologies within IOF. We provide conservative definitions for the Top 20 Classes within the IOF signature, and show how PSL formalizes the competency questions for the IOF ontologies.

Keywords. ontology design, upper ontologies, foundational ontologies

1. Introduction

The goal of the Industrial Ontologies Foundry[1] is the creation of a set of core ontologies that spans the entire domain of digital manufacturing. This set of ontologies will be non-proprietary and are expected to serve as the foundation from which other domain-dependent and/or application ontologies can be derived in modular fashion across all industrial domains and manufacturing specializations. The scope spans all industrial domains and types of manufacturing (batch/continuous process, discrete high/low volume, make-to-order), all operational areas of a manufacturing enterprise (design, engineering, manufacturing, sourcing, supply-chain management), and all stages of the product life cycle from inception through end of life.

One approach to the design of the IOF Ontologies is to use a foundational ontology as the basis for the axiomatization of the intended semantics of the classes, relations, and functions in the signature of the IOF Ontologies. The Process Specification Language (PSL) ([5], [9]) was designed to support automated reasoning and semantic integration of software within manufacturing applications. This led to a perspective that stressed planning and scheduling problems over use cases that considered more general interactions of agents within the physical world. More recently, PSL is being incorporated into the TUpper ontology [6] as Part 4 of ISO 21838 (Top Level Ontologies). In particular, TUp-

[1] `https://sites.google.com/view/industrialontologies/home`

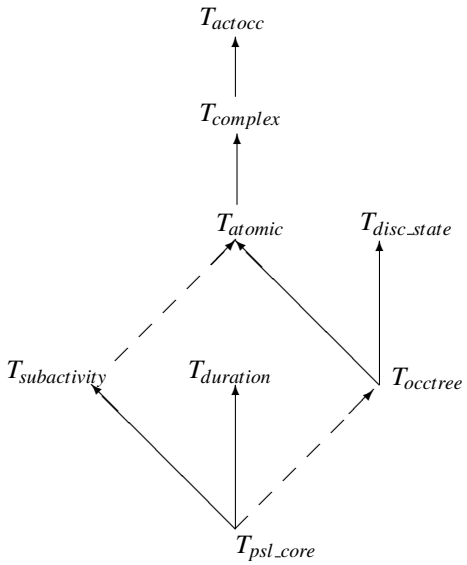

Figure 1. Modules in the PSL Ontology.

per extends PSL with modules for physical objects, location, and units of measure, thus addressing the shortcomings of using PSL alone. In these notes, we summarize various aspects of PSL and TUpper with respect to the IOF selection criteria for a foundational ontology.

2. PSL as a Foundational Ontology

The Process Specification Language (PSL) has been designed to facilitate correct and complete exchange of process information among manufacturing systems Included in these applications are scheduling, process modelling, process planning, production planning, simulation, project management, workflow, and business process reengineering.

PSL is a modular ontology, with each module consisting of at most twenty axioms. Overall, there are approximately 250 axioms spread across 31 modules. (This counts only those CLIF texts that axiomatize primitive concepts, since we have adopted the practice in which defined relation is found in its own CLIF text). The modules of the PSL Ontology[2] can be seen in Figure 1.

The goal for the TUpper Ontology is to support the ontological analysis of relevant existing standards and to integrate the ontologies within those standards ([8]). The scope

[2]The axioms for these modules can be found at
`colore.oor.net/psl_core/psl_core.clif`
`colore.oor.net/psl_occtree/occtree.clif`
`colore.oor.net/psl_subactivity/subactivity.clif`
`colore.oor.net/psl_disc_state/disc_state.clif`
`colore.oor.net/psl_atomic/atomic.clif`
`colore.oor.net/psl_complex/complex.clif`
`colore.oor.net/psl_actocc/actocc.clif`

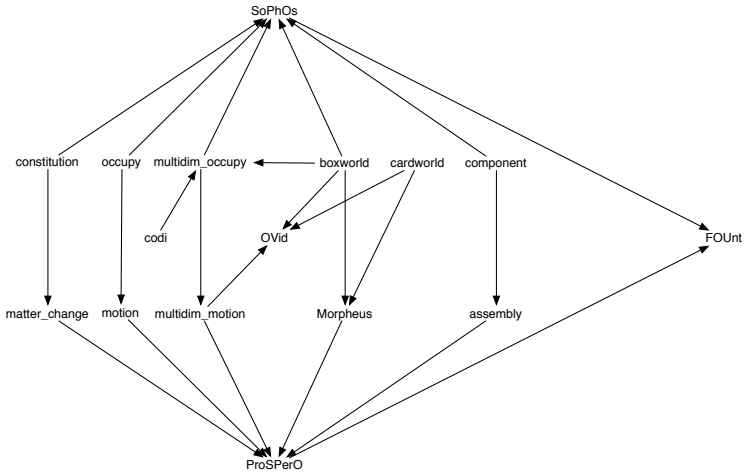

Figure 2. Modules in TUpper.

of the TUpper Ontology is determined by the task of *standards integration* – each of the generic ontologies is needed to support one of the participating standards. Each of the generic ontologies is a reductive module of TUpper. There are no additional bridge axioms in a combined signature across these modules. The verification of TUpper is therefore trivial, being equivalent to the verification of each of the generic ontologies.

TUpper also contains modules for reasoning about state designed using the methodology of [1], so that domain state ontologies and domain process ontologies are specified based on corresponding generic ontologies. For example, the location module T_{occupy} within TUpper leads to the motion ontology (i.e. how location changes), the mereotopology of physical objects leads to the assembly ontology (i.e. how the mereological relationships among components of a physical object change), and the ontology of physical objects T_{sophos} leads to physical process ontology. This latter ontology includes the axiomatization of temporary parthood. The complete set of modules in TUpper are shown in Figure 2.

2.1. Ontological Completeness

As proposed in ISO 21838-4, TUpper (which includes PSL as a module) is a top level ontology since it satisfies the criteria for breadth of coverage.

Space and Time As axiomatized by $T_{pslcore}$, an object exists at a timepoint and an activity occurs at a timepoint. Each activity occurrence has a beginof timepoint at which it starts to occur and an endof timepoint at which it ceases to occur. Similarly, each object has a beginof timepoint at which it starts to exist and an endof timepoint at which it ceases to exist. Timepoints are linearly ordered by the before relation, with endpoints at infinity. The existence of timeintervals is axiomatized in $T_{interval\_psl}$ – each interval corresponds to a pair of timepoints.

T_{region_mt} is the mereotopology that is specified over the set of spatial regions. T_{occupy_root} is the module that is the location ontology in TUpper. It is consistent with the

ontology for more than one material object to occupy exactly the same spatial region at the same time.

Actuality and Possibility Models of $T_{occtree}$ are occurrence trees, which consist of all possible sequences of atomic activity occurrences. The set of activities that actually occur in a model are elements of one branch of the occurrence tree. Models of $T_{complex}$ consist of subtrees of the occurrence tree that correspond to possible occurrences of complex activities. A legal occurrence tree is the subtree of an occurrence tree in which all activity occurrences satisfy precondition axioms that specify the conditions under which an activity can possibly occur. Dispositions are treated via such precondition axioms.

Parts, Wholes, Unity, and Boundaries TUpper adopts mereotopological pluralism – there are multiple distinct parthood and connection relations for different classes of entities. The mereologies on matter and spatial regions are complete extensional mereologies with complementation, as is the mereology of atomic (concurrent) activities. On the other hand, the mereology of complex activities is logically synonymous with the weakest mereology, and does not require the existence of sums or complements. The mereology on components and timeintervals entails Strong Supplementation, but does not require the existence of sums for underlapping parts.

Space and Place T_{occupy} is the module that is the location ontology in TUpper. T_{motion} classifies all activities that can possibly change the location of an object.

Shape is represented using the $T_{boxworld}$ module of TUpper. Holes are specified as physical objects which have shape but which are not constituted by matter.

While $T_{matter} \cup T_{constitution}$ axiomatizes material objects and their constitution by matter, T_{matter_change} classifies all activities that can change the mereology of matter and that can change a material object by changing the matter that constitutes the object.

Quantities and Mathematical Entities The FOUnt module within TUpper axiomatizes the units of measure, constraints on how units can be algebraically manipulated), and the relationship between physical objects and processes and the units of measure.

Processes and Events Processes appear as the class of activities within $T_{pslcore}$. Within the module T_{disc_state}, fluents (states) are achieved or falsified by activity occurrences, and only activity occurrences can change fluents. There are no constraints on the kinds of processes that exist. Each activity occurrence has a duration ($T_{duration}$), which can be zero in the case of an instantaneous activity occurrence (i.e. one in which the beginof timepoint is equal to the endof timepoint).

Constitution $T_{constitution}$ axiomatizes the constitutes relation between an object and its matter. Constitution only holds between objects and matter, and there is no analogue of constitution that holds between processes or nonmaterial entities.

Causality Constraints are definable between activities, as well as between activities and fluents (states). In particular, one activity can achieve the preconditions of another activity, or a fluent may trigger the occurrence of an activity.

2.2. Logical Consistency

The PSL Ontology is verified – the models of all core theories within the ontology have been characterized up to isomorphism ([7]). The logical consistency of the PSL Ontology

follows from the verification of the ontology, in which PSL is shown to be logically synonymous with the union of a set of mathematical theories. For example, T_{psl_core} (the module imported by all PSL subtheories) is synonymous with

$$T_{bounded\_linear\_order} \cup T_{scatter\_graph\_partitioning} \cup T_{graphical\_incidence} \cup T_{present\_bundle}$$

The consistency of PSL-Core follows from the consistency of each of these mathematical theories.

2.3. Multiple Representations

TUpper and PSL have a natural language representation, and they are axiomatized in both Common Logic and OWL-2. However, the OWL-2 is far too weak to satisfy any of the initial requirements or to support any of the motivating scenarios in decision support and semantic integration in the next section of this paper. In particular, any OWL-2 axiomatization will have unintended models, and the existence of such models prevents the specification of reasoning problems for planning and scheduling. It is therefore not clear how the OWL-2 axiomatization of PSL would be used.

3. Use Cases

3.1. Software Integration

Many tasks require correct and meaningful communication and integration among intelligent agents and information resources. A major barrier to such interoperability is semantic heterogeneity: different applications, databases, and agents may ascribe disparate meanings to the same terms or use distinct terms to convey the same meaning. The development of ontologies has been proposed as a key technology to support semantic integration—two software systems can be semantically integrated through a shared understanding of the terminology in their respective ontologies.

A semantics-preserving exchange of information between two software applications requires mappings between logically equivalent concepts in the ontology of each application. The challenge of semantic integration is therefore equivalent to the problem of generating such mappings, determining that they are correct, and providing a vehicle for executing the mappings, thus translating terms from one ontology into another.

The work by [4] summarizes how PSL was used to integrate the following manufacturing software systems.

- SAP ERP ([12]) is an enterprise resource planning solution that among many other capabilities can be used to design manufacturing processes and manage manufacturing operations. Production orders in the ERP system are used to represent the enterprise's demand on manufacturing operations to deliver a certain quantity of a specific product to be available by a give date and time.
- MetCapp ([11]) is a computer-aided process planning system that generates process plans, given a set of workstations, machines, setups, routing sequences, and generic part feature data. A machining capability database provides automated capabilities of cutting tools selection, speed and feed calculations, processing time estimation, and operation sequencing.

- ILOG Scheduler ([10]) consists of an extensible library of C++ classes and functions that implement scheduling concepts such as activities and resources. The library enables the representation of scheduling problems as a collection of scheduling constraints, such as activity durations, release dates and due dates, precedence constraints, resource availability, and resource sharing. These constraints in turn are used as input for ILOG Solver, which can solve the constraints to provide schedules, in which activities are assigned to resources over different time intervals.

A reproduction of this use case with the other foundational ontologies (BFO and DOLCE) for the IOF Ontologies would be another approach to the evaluation of the fitness of the foundational ontologies.

3.2. Next Generation Software

Whereas use cases for semantic integration are based on existing engineering software applications, the IOF Ontologies are also intended to support the design of new manufacturing software systems that automate decision support tasks performed by engineers at all phases of the product lifecycle.

Scenario 1: Manufacturability As products are being designed, software automatically checks to determine whether the product will be manufacturable, that is, whether there exists a process plan whose execution produces objects that satisfy the product's quality constraints. If a design is not manufacturable, potential modifications to the design are recommended.

Scenario 2: Proactive Manufacturing A new approach to achieve this integration has been the notion of proactive computing, in which information systems act in anticipation of future problems, needs, or changes of the user. To be proactive, a computer system must understand the user's context and how it changes over time. Within manufacturing enterprises, this can be facilitated by the recording of process constraints associated with a particular resource as it passes through the set of activities performed within the supply chain. RFID technology can be combined with ontologies (e.g. use a process and time ontology to store information directly on the RFID tags) to create smart items and demonstrate this approach using a motivating scenario of manufacturing process control [3]. As an item flows through a manufacturing process, information about the item can be stored on its tag. This information, along with the ontology's axioms can be used to make decisions about the future possible manufacturing processes that might be performed on the item.

4. Competency Questions

Several automated reasoning problems arise within the motivating scenarios introduced in the preceding section. In this section, we give a brief overview of how PSL can be used to represent these problems so that they can be treated as competency questions relevant for the IOF Ontologies.

4.1. Temporal Projection

The problem of temporal projection is closely related to prediction about how the world changes: given a set of activity occurrences and an initial state, what is the final state?

Problem 1. *Given a partially ordered set of activity occurrences Σ_{occ} and an initial state $\Sigma_{\mathscr{F}}$, determine whether the ground state formula $Q(O)$ holds in after the activity occurrence O:*

$$T_{psl} \cup \Sigma_{occ} \cup \Sigma_{\mathscr{F}} \models Q(O)$$

It is important to understand the significance of this formal definition of temporal projection. Given a sequence of activity occurrences, we must determine *from the axioms in T_{psl} alone* whether the formula $Q(O)$ is entailed after the activity occurrence O. We therefore need to determine the necessary set of axioms in such a process ontology in order to specify the problem of temporal projection and to characterize the solutions to this problem.

What are the minimal ontological commitments to define temporal projection? Since we are given a sequence of activity occurrences, and we must determine what fluents (states) hold after the sequence, we need to axiomatize preconditions (fluents that must hold in order to perform an action) and effects (fluents that hold after performing an action).

4.2. Plan Existence

The problem of plan existences asks whether there exists a sequence of activity occurrences such that a given state holds after the sequence. Such a set of activity occurrences constitutes a plan that achieves the goal state.

Problem 2. *Given a simple state formula $Q(o)$, an initial state $\Sigma_{\mathscr{F}}$ and a set of precondition and effect axioms $\Sigma_{activity}$ determine*

$$T_{psl} \cup \Sigma_{\mathscr{F}} \cup \Sigma_{activity} \models (\exists s)\, Q(s)$$

4.3. Scheduling

Given a set of deadlines, operations, and resources, does there exist a sequence of activity occurrences that achieve a set of goals by the deadlines?

Problem 3. *Let $Q(s)$ be a sentence with no free variables except s. For some temporal ground term T, determine*

$$T_{succ} \cup T_{pre} \cup \Sigma_{\mathscr{F}} \models (\exists s,t)\, S_0 \leq s \wedge Q(s) \wedge start(s,t) \wedge t < T$$

This is equivalent to the existence of a plan that can achieve the goal $Q(s)$ by the deadline T.

4.4. Process Plan Generation

Process plan generation differs from plan existence in two ways. First, the process plan is a complex activity (that is, an activity composed of primitive activities with partial ordering constraints and nondeterminism). Second, the intended effects of occurrences of the process plan must be equivalent to the specification of a product. Process plan generation is the construction of a plan whose occurrences produce an object with a given set of features.

Problem 4. *Given a product specification formula* $\Sigma^{product}(x)$ *for a product* $P(x)$*, find a process description* $\Sigma^{activity}(A)$ *such that*

$$T_{psl} \cup \Sigma^{activity}(A) \cup \Sigma^{product}(x) \models (\forall o,x)\, occurrence\_of(o,A) \wedge P(x) \supset holds(product(x),o)$$

4.5. Plan Verification

Given a partially order complex activity that specifies a plan, is the goal achieved after every occurrence of the plan?

Problem 5. *Given a simple state formula* $Q(o)$ *and a process specification* $\Sigma_{activity}(A)$*, determine*

$$T_{psl} \cup \Sigma_{activity}(A) \models (\forall o)\, occurrence\_of(o,A) \supset Q(o)$$

Closely related to plan verification is the problem of reasoning about the interaction of a plan with external activities (i.e. activities that are performed by actors outside of the plan). Which activities external to the plan can interfere with the plan? Must activities external to the plan occur in order for the plan to occur?

5. Axioms for the Top 20 Classes

The set of terms in the initial IOF signature focus on classes of resources and processes within manufacturing. In this section, we present the axiomatization of the intended semantics of the IOF signature using PSL, as required to support the competency questions specified in the preceding section. Note that the axioms for this initial set of terms in the IOF Ontologies forms a definitional extension of the PSL Ontology; no additional ontologies outside of PSL and TUpper were used.

5.1. Resources

Several classes within the IOF Top 20 terms are related to objects that are resources for manufacturing processes. The challenge is to provide an axiomatization that distinguishes resources from other objects that participate in an activity occurrence. Within the PSL Ontology, resources exist wherever there are concurrency constraints on activities – if two activities cannot be concurrent, there exists a resource that they both require. The different relationships between resources and activities are classified by the roles that the resources play in the activities.

5.2. Material Resources

A resource *r* is consumable by an activity *a* if any other activity that also requires *r* is not possible to perform after a completes its occurrence. Examples of consumable resources include wood in a fire, or raw materials in a manufacturing production process.

Definition 1. *A material resource is a material object that is consumable by an activity.*

$$(\forall x)\ \textit{iof:MaterialResource}(x) \equiv (\exists a)\ MaterialObject(x) \wedge consumable(a,x) \tag{1}$$

5.3. Manufacturing Tool

A resource r is reusable by an activity a if any other activity that also requires r is still possible to perform after a completes its occurrence, in every possible future. A hammer or screwdriver is an example of a reusable resource – as soon as one activity occurs, it is always possible to perform the next activity.

A resource r is weakly reusable by an activity a iff for any other activity that also requires r is still possible to perform after a completes its occurrence, in every possible future situation unless it is prevented. For example, a paintbrush is reusable only if we put it into varsol after use; otherwise, it is not reusable.

Definition 2. *A manufacturing tool is a solid physical object that is reusable by an activity.*

$$(\forall x)\ \textit{iof:ManufacturingTool}(x) \equiv (\exists a)\ solid\_physical\_object(x) \wedge reusable(a,x) \tag{2}$$

5.4. Manufacturing Machine

A resource r is possibly reusable by an activity a iff for any other activity that also requires r is still possible to perform after a completes its occurrence, in some possible future situation. A machine that requires some setup between different activities. After the first activity occurs, it is possible for the other activity, but only if the setup activity occurs first.

Definition 3. *A manufacturing machine is a solid physical object that is possibly reusable by an activity.*

$$(\forall x)\ \textit{iof:ManufacturingMachine}(x) \equiv \tag{3}$$
$$(\exists a)\ solid\_physical\_object(x) \wedge possibly\_reusable(a,x)$$

5.5. Fixture

A resource r is wearable with respect to an activity a1 iff after the occurrence of a1 there is always a situation in every possible future where any other activity that requires r is no longer possible. An example of a wearable resource is a drill bit – in every possible future, there will exist a situation where the bit has worn down to the point where it can no longer be used.

Definition 4. *A fixture is a solid physical object that wears out through use by an activity.*

$$(\forall x)\ \textit{iof:Fixture}(x) \equiv (\exists a)\ solid\_physical\_object(x) \wedge wearable(a,x) \tag{4}$$

5.6. Other Classes of Resources

The PSL Ontology axiomatizes other classes of resources that play a role within planning and scheduling, although they did appear in any of the IOF Top 20 Terms.

A resource r is weakly consumable with respect to an activity a_1 iff after the occurrence of a_1, there always exists a possible future along which any other activity that requires r will never be possible. Consider, for example, a paintbrush – if we do put it into varsol after using it, then any activity that requires the brush will no longer be possible.

A resource r is renewable with respect to an activity a iff for any other activity that also requires r is still possible to perform after a completes its occurrence, in every possible future situation unless it is prevented. An example of a renewable resource is a solar-charged battery – once it is depleted, there will always exist a future situation where the sun recharges the battery so that it can be used again.

5.7. Domain Process Ontologies

PrOSPerO (Process Ontologies for Solid Physical Objects) is the collection of domain process ontologies within TUpper that axiomatize the classes of activities that change objects within manufacturing.

The approach [1] is to classify all possible activities in a domain by characterizing possible all changes in the domain. A domain ontology is translated into a domain state ontology (i.e. relations are mapped to fluents). Activity occurrences correspond to mappings between models of the domain ontology. Activities with respect to possible changes.

With ProSPerO, we treat SoPhOs and Occupy as the domain ontologies, leading to the following domain process ontologies:

- Material removal / addition activities
- Shape-changing activities
- Assembly / disassembly activities
- Motion activities

In particular, there is a domain ontology that axiomatizes the mereological relation for the components of a solid physical object.

Definition 5.

$$(\forall x)\ \textit{iof:Assembly}(x) \equiv \tag{5}$$
$$solid\_physical\_object(x) \wedge (\exists y)\, component\_of(y,x) \wedge (x \neq y)$$

Definition 6.

$$(\forall x)\ \textit{iof:Component}(x) \equiv \tag{6}$$
$$solid\_physical\_object(x) \wedge ((\forall y)\, component\_of(x,y) \supset (x = y))$$

Within the corresponding domain process ontology, assembly and disassembly activities change the $comp(x,y)$ mereological fluent.

Definition 7. *An assembly process achieves the $comp(x,y)$ fluent.*

$$(\forall x)\ \textit{iof:AssemblyProcess}(x) \equiv \tag{7}$$
$$activity(x) \wedge ((\forall o)\, occurrence\_of(o,x) \supset (\exists y,z)\, achieves(o, comp(y,z))$$

Similarly, transportation processes change the location of objects.

Definition 8.

$$(\forall x)\ \textit{iof:TransportationProcess}(x) \equiv \tag{8}$$
$$activity(x) \wedge ((\forall o)\, occurrence\_of(o,x) \supset (\exists y,z)\, achieves(o, loc(y,z))$$

5.8. Manufacturing Processes

A processor activity is an activity which uses some set of resources, consumes or modifies some other set of resources, and produces or modifies a set of objects.

Definition 9.

$$(\forall x)\ \textit{iof:ManufacturingProcess}(x) \equiv processor\_activity(x) \tag{9}$$

5.9. Processes: Composition

Activities can be composed together to construct complex activities, or decomposed into primitive activities. Occurrences of complex activities correspond to sets of occurrences of their subactivities. Different occurrences of complex activities may contain occurrences of different subactivities or different orderings on the same subactivity occurrences. There is a partial ordering of subactivity occurrences for a set of complex activity occurrences.

Definition 10.

$$(\forall x)\ \textit{iof:Task}(x) \equiv primitive(x) \tag{10}$$

5.10. Plans

A set of activity occurrences is a plan in the PSL Ontology if and only if all occurrences of subactivities that agree on state also agree on being subactivity occurrences. (i.e. there exists a set of fluents that are achieved by all occurrences of the activity).

Definition 11. *An IOF plan is an activity all of whose occurrences are plans in the PSL Ontology.*

$$(\forall a)\ \textit{iof:Plan}(a) \equiv ((\forall o)\, root(o,a) \supset plan(o)) \tag{11}$$

5.11. Products

Definition 12. *A product is an object that satisfies a design specification.*

$$(\forall x,o)\, P(x) \supset (prior(product(x),o) \equiv \Phi(x,o) \tag{12}$$

where $P(x)$ is a specific class of objects and $\Phi(x,o)$ is a formula.

5.12. Process Plans

Definition 13. *A process plan is a plan whose goal is a product.*

$$(\forall a)\ \textit{iof:Process\_plan}(x) \equiv \tag{13}$$
$$((\forall o)\ \textit{iof:plan}(a) \wedge \textit{occurrence\_of}(o,a) \supset (\exists x)\ \textit{achieves}(o, \textit{product}(x))$$

5.13. Manufacturing Process Plans

An activity occurrence $occ2$ is the next processor subactivity occurrence after $occ1$ in an activity a iff the output material of $a1$ is the input material of $a2$, and there is no other processor subactivity of a which consumes the output material from $a1$, and which occurs between $a1$ and $a2$. There exists a material flow ordering, which is a partial ordering over the processor subactivity occurrences of a with respect to resource flow. An activity is a resource path iff the subactivity occurrence ordering is equivalent to the material flow ordering.

Definition 14. *A manufacturing process plan is a resource path that is also a process plan for a product.*

$$(\forall a)\ \textit{iof:ManufacturingProcessPlan}(a) \equiv (\exists x)\ \textit{iof:Process\_plan}(a,x) \wedge \textit{resource\_path}(a) \tag{14}$$

6. Summary

We have seen how the PSL Ontology, together with the modules of the TUpper Ontology, can be used to provide conservative definitions that axiomatize the intended semantics for the Top 20 Classes in the IOF signature. Furthermore, PSL has the expressiveness to formally specify the competency questions that arise from motivating scenarios within IOF, and earlier work (e.g. [3]) has demonstrated how PSL can be used to support automated reasoning.

Nevertheless, major challenges remain unresolved. What role does the foundational ontology play? Can multiple foundational ontologies be used together for the axiomatization of the IOF Ontologies? For example, can the mappings between PSL and DOLCE [2] be used to specify axioms that use both ontologies?

More importantly, we still do not know what we need to axiomatize – the IOF community has not yet agreed upon the signature of the IOF Ontologies, let alone the intended semantics of the signature. Without a specification of the intended semantics of the IOF signature, it is difficult to make any claims about the adequacy of an existing ontology to be reused to axiomatize the intended semantics.

Finally, many of the issues that are being encountered in the design of IOF are open research problems in ontological engineering. We must be careful to avoid easy approaches that do not ultimately lead to solutions to these problems.

References

[1] Bahar AAmeri. Using Partial Automorphisms to Design Process Ontologies. In *FOIS*, pages 309–322, 2012.

[2] Carmen Chui and Michael Grüninger. Merging the DOLCE and PSL Upper Ontologies. In *KEOD 2014 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Rome, Italy, 21-24 October, 2014*, pages 16–26, 2014.

[3] M. Gruninger, S. Shapiro, M.S. Fox, and Weppner. Combining RFID with Ontologies to Create Smart Objects. *International Journal of Production Research*, 48:2633–2654, 2010.

[4] Michael Grüninger. The Ontological Stance for a Manufacturing Scenario. *J. Cases on Inf. Techn.*, 11(4):1–25, 2009.

[5] Michael Grüninger. Using the PSL Ontology. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 423–443. Springer Berlin Heidelberg, 2009.

[6] Michael Grüninger, Carmen Chui, and Megan Katsumi. Upper ontologies in colore. In *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano, Italy, September 21-23, 2017*, 2017.

[7] Michael Grüninger, Torsten Hahmann, Ali Hashemi, Darren Ong, and Atalay Özgövde. Modular First-Order Ontologies via Repositories. *Applied Ontology*, 7(2):169–209, 2012.

[8] Michael Grüninger, Torsten Hahmann, Megan Katsumi, and Carmen Chui. A sideways look at upper ontologies. In *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil*, pages 9–22, 2014.

[9] Michael Grüninger and Christopher Menzel. The Process Specification Language (PSL) Theory and Applications. *AI Mag.*, 24(3):63–74, September 2003.

[10] Inc. ILOG. *ILOG Scheduler 6.3 Reference Manual*. 2008.

[11] Institute of Advanced Manufacturing Sciences. *MetCapp Utilities Manual*. 1994.

[12] SAP. *SAP Library*. 2008.