Formal Ontology in Information Systems
M. Donnelly and G. Guizzardi (Eds.)
IOS Press, 2012

doi:10.3233/978-1-61499-084-0-309

Using Partial Automorphisms to Design Process Ontologies

Bahar Aameri [a]
[a] *Department of Computer Science, University of Toronto, Canada*
bahar@cs.toronto.edu

Abstract. Process ontologies play key roles in semantic integration and decision support systems for applications in manufacturing, enterprise modeling and e-commerce. In this paper, we propose a methodology for the design and verification of domain-specific process ontologies that are extensions of generic process ontologies. This allows us to evaluate the correctness of process ontologies with respect to the class of intended models for their respective domains. Our approach is based on the correspondence between the effects of activities in the process ontology and the partial automorphisms of models of the underlying domain ontology. We then investigate in detail the process ontology for the domain of chains (sets of disjoint linear orderings) using this methodology.

1. Introduction

Although much work has been done on the development of generic process ontologies ([1], [2], [3]), many applications within semantic integration and decision support require process ontologies that axiomatize classes of activities that are specific to a particular domain, such as biology, manufacturing, and e-commerce. Although specific ontologies have been developed in these areas for individual problems, nobody has proposed a general method to design, verify, and evaluate such domain specific ontologies.

Ontology verification includes a characterization of the models of the axiomatization of an ontology up to isomorphism and determining whether or not these models are equivalent to intended models of the ontology [4]. The verification of domain-specific process ontologies requires us to characterize the correctness and completeness of the classes of activities with respect to a particular domain. Correctness is concerned with determining whether the effects of the activities preserve the underlying constraints on possible states within the domain. Completeness addresses the problem of determining whether the process ontology specifies all possible activities i.e. all possible ways of changing states with regard to domain constrains.

The methodology presented in this paper was initially motivated by the problem of axiomatizing qualitative spatial change. Although many efforts have been made in developing ad-hoc representations of spatial change, such as [5] and [6], relatively little have been done on expanding a general approach in this regard. An attempt to generalize the axiomatization of spatial change was by Grenon and Smith [7]; they presented a modular ontology, called SNAP and SPAN, such that a SNAP ontology axiomatizes the depiction of entities at a given time, whereas a SPAN ontology describes entities that

reveal themselves within some interval of time. Intuitively, "*a SNAP ontology is analogous to a snapshot, or to the results of a process of sampling*", while a SPAN ontology is "*analogous to videos time spanning*" [7]. A major concern with this approach is that some axioms of the ontology contain quantification over the ontology itself, which is unusual. Moreover, the correctness and completeness of the ontology has not been verified. The work in [8] proposes another systematic approach for modeling spatial change; their formalization is based on a first-order casual theory that adheres to the semantics of situation calculus and specifies the general template for representing a changing spatial environment. However, like SNAP and SPAN ontology, they do not provide a general methodology for verification of the resulting theory.

It should be emphasized that a domain-specific process ontology is distinct from process descriptions or action theories ([3], [9]). The process ontology contains an explicit axiomatization of a taxonomy of classes of activities, whereas an action theory specifies the effects and preconditions of particular activities within the domain. The relationship between the process ontology and action theories will be discussed in more detail towards the end of the paper. In addition, the axiomatization of the process ontology is independent of whatever approach is taken to the frame problem.

The approach presented here is to begin with an ontology about a specific domain, such as shapes or partial orderings, and then characterize the ways in which activities can possibly change relationships among objects in the domain. By identifying models of the domain ontology associated with states within the process ontology, we can use properties of models to prove correctness of the domain process ontology. Since an activity occurrence changes some fluents while leaving others unchanged, we turn to the notion of partial automorphism, which is a mapping on a model to itself that preserves some substructures. Intuitively, fluents are unchanged by an occurrence of an activity if they correspond to substructures of models of the domain ontology that are fixed by the partial automorphisms. We use partial automorphisms to characterize the preserved substructures that corresponds to the effects of an activity. In particular, we take advantage of the properties of partial automorphisms to specify a complete taxonomy of the domain process and then to prove that every possible activity falls in one of those classes.

We apply the methodology to a domain ontology that axiomatizes the class of chains (disjoint discrete linear orderings). We do this for several reasons. First, the corresponding domain process ontology can be applied to areas such as train marshalling and alternative splicing within gene transcription. Second, chains and their partial automorphisms are well-known and well-studied structures [10,11]. Finally, the underlying domain ontology for Blocks World also axiomatizes the class of chains, and this allows us to demonstrate the correctness and completeness of this approach with respect to a well-studied and well-understood domain.

After a brief overview of the underlying generic process ontology that we are using (the PSL Ontology), we proceed through the different steps within the proposed methodology. The steps can be summarized as follows;

- First we need an axiomatization for the underlying domain ontology which enable us to reason about changes in that domain. Section 3 introduces a definition which gives a theory, called *domain state ontology*, equivalent to the domain ontology, but in the signature of PSL ontology which allows to reason about change.

- In the next step we characterize models of domain state ontology with respect to its underlying domain. The key idea here is to associate activity occurrences and their effects with mappings between models of the domain ontology (Section 4).
- Section 5 shows how special subsets of partial automorphisms (called *scaffold*) can be employed to characterize mappings between models of the domain ontology and hence to characterize activity occurrences that are associated with them.
- Section 6 describes certain properties of models of the underlying domain ontology and illustrate how those properties along with the scaffolds of the models lead to a classification theorem for primitive activities which ensures the completeness of the domain process ontology. It is straightforward then to characterize and axiomatize activity classes with respect to the classification theorem (Sections 7).

2. PSL Ontology

The PSL Ontology [2] is a modular first-order ontology that axiomatizes a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of processes. Two of the theories within the PSL Ontology play key roles in this paper; $T_{occtree}$ (theory of occurrence trees) and T_{disc_state} (theory of fluents and discrete change) [1].

Within the PSL Ontology, an occurrence tree Γ is a partially ordered set of activity occurrences, such that for a given set of activities, all discrete sequences of their occurrences are branches of the tree. An occurrence tree contains all occurrences of all atomic activities; Because the tree is discrete, each activity occurrence in the tree has a unique successor occurrence of each activity. Every sequence of activity occurrences has an initial occurrence which is the root of an occurrence tree.

Within the PSL Ontology, the notion of state is represented by reified fluents. Intuitively, a change in state is captured by fluents that are either achieved or falsified by an activity occurrence. Furthermore, a fluent can only be changed by the occurrence of activities. Thus, if some fluent holds after an activity occurrence, but after an activity occurrence later along the branch it is false, then an activity must occur at some point between that changes the fluent. This also leads to the requirement that the fluents holding after an activity occurrence will be the same fluents that are prior to any successor occurrence, since there cannot be an activity occurring between them.

Here is a brief definition for the relations in PSL that we will use in this paper:

- *arboreal(o)* denotes that activity occurrence $\mathbf{o}$ is an element of occurrence tree.
- *prior(f, o)* denotes that the fluent $\mathbf{f}$ holds before the activity occurrence $\mathbf{o}$.
- *holds(f, o)* denotes that the fluent $\mathbf{f}$ holds after the activity occurrence $\mathbf{o}$.
- $achieves(o, f) \equiv \neg prior(f, o) \wedge holds(f, o)$.
- $falsifies(o, f) \equiv prior(f, o) \wedge \neg holds(f, o)$.
- $changes(o, f) \equiv achieves(o, f) \vee falsifies(o, f)$.

3. Axiomatization of the Domain State Ontology

Many ontologies provide an axiomatization of some set of concepts independently of the notion of change. For example, mereotopologies such as RCC [12] axiomatize notions

[1] We use the axiomatization of the PSL Ontology found at `http://www.mel.nist.gov/psl/psl_ontology/`.

of parthood and connectedness but there is no axiomatization for classes of activities that change such relations. We will refer to such axiomatizations as *domain ontologies*. In this paper we will use chain structures, adopted from [13], as the domain ontology. [13] provided an axiomatization of chain structures T_{chain}[2] and proved that models of T_{chain} are isomorphic to structures in the following class:

Definition 1 *A structure $\mathcal{L} = \langle L, lst \rangle$ is a chain structure iff*

1. $\mathbf{lst}$ *is isomorphic to a set of disjoint discrete linear orderings;*
2. $\langle \mathbf{x} \rangle \in \mathbf{minimal}$ *iff there is no* $\mathbf{y} \in L$ *such that* $\langle \mathbf{y}, \mathbf{x} \rangle \in \mathbf{lst}$,
3. $\langle \mathbf{x} \rangle \in \mathbf{maximal}$ *iff there is no* $\mathbf{y} \in L$ *such that* $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{lst}$,
4. $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{cover}$ *iff* $\langle \mathbf{y}, \mathbf{x} \rangle \in \mathbf{lst}$ *and there is no* $\mathbf{z} \in L$ *such that* $\langle \mathbf{y}, \mathbf{z} \rangle \in \mathbf{lst}$ *and* $\langle \mathbf{z}, \mathbf{x} \rangle \in \mathbf{lst}$,
5. $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{comparable}$ *iff* $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{lst}$ *or* $\langle \mathbf{y}, \mathbf{x} \rangle \in \mathbf{lst}$.

Beginning with the domain ontology, we specify an equivalent theory which will allow us to reason about change.

Definition 2 *State constraints are universal sentences containing only $prior$ and $arboreal$ relations and a unique activity occurrence variable.*

Figure 1 shows state constrains for T_{chain}. Definition 3 describes how to specify state constrains and the corresponding domain state ontology for a domain ontology .

Definition 3 *Let T be a domain ontology and D be a set of state constraints.*
Let $\pi : \mathcal{L}(T) \rightarrow \mathcal{L}(D)$ be a bijection on the set of relation symbols in the signature $\mathcal{L}(T)$ of T and the set of fluent symbols in the signature $\mathcal{L}(D)$ of D. Let Δ be the set of sentences

$(\forall x_1, ..., x_n, o)\ p_i(x_1, ..., x_n) \equiv prior(\pi(p_i(x_1, ..., x_n)), o)$

for each relation symbol $p_i(x_1, ..., x_n)$ in $\mathcal{L}(T)$.
$T_{psl} \cup D$ is the domain state ontology for T iff $T_{psl} \cup \Delta \cup T \models D$ and $T_{psl} \cup \Delta \cup D \models T$.

To specify state constrains for T_{chain}, we map the $\mathbf{lst}$ relation of the domain ontology to the fluent $\mathbf{lt}$ using the translation definition

$$(\forall x, y, o)\ lst(x, y) \equiv prior(lt(x, y), o)$$

The resulting theory $T_{stchain}$ can be found in Figure 1.

Theorem 1 [3] *$T_{psl} \cup T_{stchain}$ is the domain state ontology for T_{chain}.*

The relationship between the domain ontology and the domain state ontology is captured by the following result:

Theorem 2 *Let T be a domain ontology and let D be a set of state constraints.*
The domain state ontology for T is unique up to definably equivalence.
In the other words $T_{psl} \cup D$ is definably equivalent to $T_{psl} \cup T$.

[2] Axioms of T_{chain} can be found at `http://stl.mie.utoronto.ca/colore/ordering/chains.clif`

[3] We omitted proofs for all theorems due to lack of space, but they can be found at:
`http://www.cs.toronto.edu/~bahar/chains/chains_partial_aut_proofs.pdf`

$$(\forall x, o)\ arboreal(o) \supset \neg prior(lt(x, x), o). \tag{1}$$

$$(\forall x, y, z, o)\ (arboreal(o) \wedge prior(lt(x, y), o) \wedge prior(lt(y, z), o)) \supset prior(lt(x, z), o). \tag{2}$$

$$(\forall x, y, z, o)\ (arboreal(o) \wedge prior(lt(x, y), o) \wedge prior(lt(x, z), o)) \supset (prior(lt(y, z), o) \vee prior(lt(z, y), o) \vee y = z). \tag{3}$$

$$(\forall x, y, z, o)\ (arboreal(o) \wedge prior(lt(y, x), o) \wedge prior(lt(z, x), o)) \supset (prior(lt(y, z), o) \vee prior(lt(z, y), o) \vee y = z). \tag{4}$$

$$(\forall x, o)\ (arboreal(o) \wedge \neg prior(minimal(x), o) \supset (\exists y)(prior(cover(x, y), o))). \tag{5}$$

$$(\forall x, o)\ (arboreal(o) \wedge \neg prior(maximal(x), o) \supset (\exists y) prior(cover(y, x), o)). \tag{6}$$

$$(\forall x, o)\ (prior(minimal(x), o) \equiv (arboreal(o) \wedge \neg(\exists y) prior(lt(y, x), o))). \tag{7}$$

$$(\forall x, o)\ (prior(maximal(x), o) \equiv (arboreal(o) \wedge \neg(\exists y) prior(lt(x, y), o))). \tag{8}$$

$$(\forall x, y, o)\ (prior(cover(x, y), o) \equiv (arboreal(o) \wedge prior(lt(y, x), o) \wedge \neg(\exists z)(prior(lt(y, z), o) \wedge prior(lt(z, x), o)))). \tag{9}$$

$$(\forall x, y, o)\ (prior(comparable(x, y), o) \equiv (arboreal(o) \wedge (prior(lt(x, y), o) \vee prior(lt(y, x), o)))). \tag{10}$$

$$(\forall o, x, y)\ prior(lt(x, y), o) \supset (prior(point(x), o) \wedge prior(point(y), o)). \tag{11}$$

Figure 1. Axioms for the Chain State Ontology $T_{stchain}$.

4. Models of the Domain State Ontology

The next step in the design of the ontology focuses on characterizing models of domain state ontology. We show that each state of a domain state ontology can be associated with a model of its corresponding domain ontology, so changing state is equivalent to a mapping from one model of the domain ontology to another. This property can be applied in model-characterization of the domain state ontology.

To start, we use Theorems 2 and 1 to characterize models of $T_{stchain}$.

Definition 4 *Let* $\mathfrak{M}^{stchain}$ [4] *be the class of structures such that* $\mathcal{M} \in \mathfrak{M}^{stchain}$ *iff*

1. *there exists a model* $\mathcal{N}$ *of* T_{psl} *with occurrence tree* Γ *such that* $\mathcal{N} \subset \mathcal{M}$*;*
2. *there exists a mapping* $\mu : \Gamma \to \mathfrak{M}^{chain}$ *such that* $\langle \mathbf{lt}(\mathbf{x}, \mathbf{y}), \mathbf{o} \rangle \in \mathbf{prior}^{\mathcal{M}} \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{lst}^{\mu(\mathbf{o})}$ *and* $\langle \mathbf{point}(\mathbf{x}), \mathbf{o} \rangle \in \mathbf{prior}^{\mathcal{M}} \Leftrightarrow \langle \mathbf{x} \rangle \in \mathbf{point}^{\mu(\mathbf{o})}$.

It is straightforward to prove the following representation theorem for $T_{stchain}$:

[4] We denote structures by calligraphic font: $\mathcal{M}, \mathcal{N}, \ldots$ classes of structures by fraktur font: $\mathfrak{M}, \mathfrak{N}, \ldots$ the domain of a structure $\mathcal{M}$ by M, the extension of relation $\mathbf{R}$ in a structure $\mathcal{M}$ by $\langle \mathbf{a_1}, \ldots, \mathbf{a_i} \rangle \in \mathbf{R}^{\mathcal{M}}$, and domain and range of a mapping φ by $dom(\varphi)$ and $im(\varphi)$.

Theorem 3 *$\mathcal{M} \in \mathfrak{M}^{stchain}$ iff it is isomorphic to a model of $T_{stchain} \cup T_{psl}$.*

In fact, this can be generalized to a characterization of the relationship between models of any domain state ontology and its corresponding domain ontology:

Theorem 4 *Let $T_{psl} \cup D$ be the domain state ontology for the domain ontology T. $\mathcal{M} \in Mod(T_{psl} \cup D)$ iff*

1. *$\mathcal{N} \subset \mathcal{M}$ is a model of T_{psl} with occurrence tree Γ;*
2. *there exists a mapping $\mu : \Gamma \rightarrow Mod(T)$ such that for each relation $\mathbf{p_i}(\mathbf{x_1}, ..., \mathbf{x_n})$ in the signature of T*

$$\langle \pi(\mathbf{p_i}(\mathbf{x_1}, ..., \mathbf{x_n})), \mathbf{o} \rangle \in \mathbf{prior}^{\mathcal{M}} \Leftrightarrow \langle \mathbf{x_1}, ..., \mathbf{x_n} \rangle \in \mathbf{p_i}^{\mu(\mathbf{o})}$$

3. *there exists a mapping $\eta : \Gamma \rightarrow Mod(T)$ such that for each relation $\mathbf{p_i}(\mathbf{x_1}, ..., \mathbf{x_n})$ in the signature of T*

$$\langle \pi(\mathbf{p_i}(\mathbf{x_1}, ..., \mathbf{x_n})), \mathbf{o} \rangle \in \mathbf{holds}^{\mathcal{M}} \Leftrightarrow \langle \mathbf{x_1}, ..., \mathbf{x_n} \rangle \in \mathbf{p_i}^{\eta(\mathbf{o})}$$ [5]

Alternatively, we can consider the set of fluents (commonly referred to as the state) associated with an activity occurrence:

Definition 5 $\Sigma(\mathbf{o}) = \{\mathbf{f} : \langle \mathbf{f}, \mathbf{o} \rangle \in \mathbf{prior}\}$ *and* $\Pi(\mathbf{o}) = \{\mathbf{f} : \langle \mathbf{f}, \mathbf{o} \rangle \in \mathbf{holds}\}$

Since $\Sigma(\mathbf{o})$ is isomorphic to a unique model $\mu(\mathbf{o})$ of T for any activity occurrence $\mathbf{o}$, a model $\mathcal{T}$ of $T_{psl} \cup D$ can therefore be characterized by an injection from the set of states in $\mathcal{T}$ to the set of models of T.

5. Change and Partial Automorphisms

We will be defining classes of activities with respect to how their occurrences change or preserve state.

Definition 6 *A fluent $\mathbf{f}$ is changed by an activity occurrence $\mathbf{o}$ iff $\mathbf{f} \in \Sigma(\mathbf{o}) \Leftrightarrow \mathbf{f} \notin \Pi(\mathbf{o})$. A fluent $\mathbf{f}$ is preserved by an activity occurrence $\mathbf{o}$ iff $\mathbf{f} \in \Sigma(\mathbf{o}) \Leftrightarrow \mathbf{f} \in \Pi(\mathbf{o})$* [6].

As illustrated in the previous section, changing state within a domain state ontology is equivalent to a mapping from one model of the corresponding domain ontology to another So to characterize activity classes, we need a representation for the mappings that can be used to characterize the fluents that are preserved by the corresponding activity occurrences. In the trivial case, an activity occurrence $\mathbf{o}$ that does not change any fluents satisfies $\Sigma(\mathbf{o}) = \Pi(\mathbf{o})$, and this corresponds to the identity mapping between the corresponding models. Activity occurrences that have nontrivial effects correspond to mappings between models that preserve some substructure of the models (i.e. the set of fluents that do not change). To capture this idea, we use the following notion:

[5] PSL Ontology guarantees that if $\mathbf{o_2}$ is any successor occurrence of $\mathbf{o_1}$, then for all fluents $\mathbf{f}$, $prior(\mathbf{f}, \mathbf{o_2}) = holds(\mathbf{f}, \mathbf{o_1})$ and so $\mu(\mathbf{o_2}) = \eta(\mathbf{o_1})$.

[6] Note that in PSL, $changes(\mathbf{o}, \mathbf{f})$ iff $prior(\mathbf{f}, \mathbf{o}) \equiv \neg holds(\mathbf{f}, \mathbf{o})$.

Definition 7 *Let $\mathcal{M}_1, \mathcal{M}_2$ be structures with the same signature.*

An injective mapping φ such that $dom(\varphi) \subseteq M_1$ and $im(\varphi) \subseteq M_2$ is a partial isomorphism iff

$$\langle \mathbf{x_1}, \ldots, \mathbf{x_n} \rangle \in \mathbf{R}^{\mathcal{M}_1} \quad \text{iff} \quad \langle \varphi(\mathbf{x_1}), \ldots, \varphi(\mathbf{x_n}) \rangle \in \mathbf{R}^{\mathcal{M}_2}$$

for some relation $\mathbf{R}$ in the signature. A partial isomorphism $\varphi : \mathcal{M} \to \mathcal{M}$ is a partial automorphism.

This definition of partial isomorphism is more general than the one proposed in [14], which requires that the extensions of all relations in the signature be preserved by the mapping.

For presenting mappings we use *path* notation introduced in [15]. Path notation represents a mapping as a sequence of elements so that if an element a is immediately before b, then $\alpha(a) = b$. The sequence can end at any point either by a close parenthesis or a square bracket. If it ends with a close parenthesis, then the last element of the sequence is mapped to the first. Otherwise it means that the last element is not in the domain of the mapping. For example, $\alpha = \begin{pmatrix} c_1 & c_2 & c_3 \\ c_2 & c_3 & c_1 \end{pmatrix}$ is presented by path notation as $\alpha = (c_1, c_2, c_3)$, while $\alpha' = \begin{pmatrix} c_1 & c_2 \\ c_2 & c_3 \end{pmatrix}$ is presented as $\alpha' = (c_1, c_2, c_3]$. $(c_1][c_2][c_3]$ and $(c_1)(c_2)(c_3)$ denote empty and identity mappings respectively.

As an example of a partial isomorphism, suppose that an activity occurrence $\mathbf{o}$ has three successor occurrences in the occurrence tree Γ. Let the chain structures depicted in Figure 2(c) be models of T_{chain} corresponding to the occurrences of $\mathbf{o}, \mathbf{o_1}, \mathbf{o_2}$ and $\mathbf{o_3}$. We can see that the substructure of $\eta(\mathbf{o})$ that is induced by the set $\{c_1, c_2\}$ is also a substructure of the models $\eta(\mathbf{o_1}), \eta(\mathbf{o_2})$ and $\eta(\mathbf{o_3})$. Note that here $\eta(\mathbf{o}) = \mu(\mathbf{o_1}) = \mu(\mathbf{o_2}) = \mu(\mathbf{o_3})$. Intuitively, this substructure is invariant under the partial isomorphisms (i.e. preserved by mappings between the models) and corresponds to fluents that are not changed by the successors of $\mathbf{o}$.

We can generalize this discussion with the following:

Lemma 1 *Let $\mathcal{M}$ be a model of a domain state ontology.*

For any activity occurrence $\mathbf{o} \in \Gamma$, let $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ be the set of partial isomorphisms between the models $\mu(\mathbf{o})$ and $\eta(\mathbf{o})$.

For any relation $\mathbf{p}(\mathbf{x_1}, ..., \mathbf{x_n})$ in the signature of the domain ontology, the fluent $\pi(\mathbf{p}(\mathbf{x_1}, ..., \mathbf{x_n}))$ is preserved by the activity occurrence $\mathbf{o}$ iff there exists $\varphi \in PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ such that $\mathbf{x_1}, ..., \mathbf{x_n} \in dom(\varphi) \cap im(\varphi)$.

For a particular element in the occurrence tree, there exists a unique set of partial isomorphisms; however, if we are to define classes of activities with respect to their occurrences, we need to characterize all possible partial isomorphisms. One approach is to characterize the invariant substructures by partial automorphisms, which are equivalent to isomorphisms between substructures of $\mu(\mathbf{o})$. Whereas automorphisms describe the global symmetries of a structure, partial automorphisms describe the local symmetries; they are especially useful in cases where a structure does not have any nontrivial automorphisms. For example, one partial automorphism of the model in Figure 2(c) that corresponds to $\mathbf{o_2}$ is $(c_1)(c_2)(c_3][c_4]$.

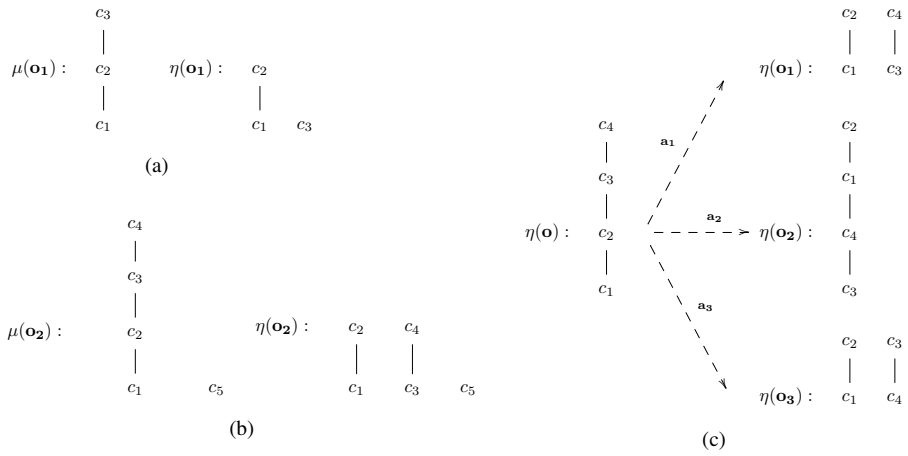

Figure 2. Examples of chain transform processes. Vertical orders represent *lt* fluents; an element of a chain c_i is on another element c_j iff $lt(c_j, c_i)$. $\mu(\mathbf{o})$ denotes the model that is associated with activity occurrence $\mathbf{o}$ prior to its occurrence and $\eta(\mathbf{o})$ denotes the model that is associated with $\mathbf{o}$ after its occurrence.

The set of all partial automorphisms of a structure $\mathcal{M}$ forms a monoid [11], denoted by $PAut(\mathcal{M})$. Partial automorphisms have the advantage that they have been extensively studied and formalized for a wide range of mathematical structures [16,17,18,19]. In particular, the monoid of partial automorphisms of a chain structure is known as $\mathcal{IO}_n$; representation and structure theorems for this class of monoids are discussed in [10].

Definition 8 *Let $\mathcal{M}$ be a model of a domain state ontology.*

For any activity occurrence $\mathbf{o} \in \Gamma$, $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ is the subset of all partial isomorphisms of $\mu(\mathbf{o})$ and $\eta(\mathbf{o}))$ such that $PIso(\mu(\mathbf{o}), \eta(\mathbf{o})) \subseteq PAut(\mu(\mathbf{o}))$.

The intuition is that for any model $\mathcal{M}$ of the domain ontology, $PAut(\mathcal{M})$ encodes the information about all possible mappings that preserve substructures; by the correspondence with activity occurrences, this is equivalent to the specification of all possible ways in which an activity occurrence can change state. Thus, the monoid $PAut(\mu(\mathbf{o}))$ encodes all of the ways that $\mathbf{o}$ changes and preserves fluents. On the other hand, $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ captures the actual way in which $\mathbf{o}$ changes and preserves fluents. Later in the paper, we will define classes of activities with respect to $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$; the completeness of the classification of activities is characterized with respect to $PAut(\mu(\mathbf{o}))$.

If we consider the models in Figure 2(a), then we have

$$\begin{aligned} PAut(\mu(\mathbf{o})) = \{&(c_1](c_2](c_3], (c_1)(c_2](c_3], (c_1](c_2)(c_3], (c_1](c_2](c_3), (c_1, c_2](c_3],\\ &(c_1, c_3](c_2], (c_1](c_2, c_3], (c_2, c_1](c_3], (c_3, c_1](c_2], (c_3, c_2](c_1],\\ &(c_1)(c_2)(c_3], (c_1)(c_2](c_3), (c_1](c_2)(c_3), (c_1)(c_2, c_3], (c_1, c_2](c_3),\\ &(c_1)(c_3, c_2], (c_2, c_1](c_3), (c_1, c_2, c_3], (c_3, c_2, c_1], (c_1)(c_2)(c_3)\}, \end{aligned}$$

The subset of nontrivial mappings which can be associated with $\mathbf{o}$ is

$PIso(\mu(\mathbf{o}, \eta(\mathbf{o})) = \{(c_1](c_2](c_3], (c_1](c_2](c_3), (c_1)(c_2)(c_3]\}$.

Notice that $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ is union of two submonoids α_1, α_2, of $PAut(\mu(\mathbf{o}))$, where $\alpha_1 = \{(c_1](c_2](c_3], (c_1)(c_2)(c_3]\}$ and $\alpha_2 = \{(c_1](c_2](c_3], (c_1](c_2](c_3)\}$.

We therefore need some way of extracting the information about invariant substructures of the model $\mu(\mathbf{o})$ from properties of the monoid $PAut(\mu(\mathbf{o}))$ so that we can characterize the effects of the activity occurrence $\mathbf{o}$. Unfortunately, we cannot use the naive association of individual partial automorphisms or even specific submonoids with activity occurrences. In fact, the preceding example shows that we need to associate each activity occurrence with a set of submonoids of $PAut(\mu(\mathbf{o}))$.

One direction is suggested by the following lemma from [11]:

Lemma 2 *Let $\varepsilon(S)$ be the set of idempotents*[7] *in some semigroup S. For any $e \in \varepsilon(S)$, the set eSe is a submonoid of S with e as the identity element.*

In other words, if we are looking for submonoids that are associated with an activity occurrence $\mathbf{o}$, we should find the identity mappings over the substructures of $\mu(\mathbf{o})$ that remain invariant after $\mathbf{o}$.

Note that some activity occurrences can be associated with more than one idempotent mapping. For example in Figure 2(b) both $\{c_1, c_2\}$ and $\{c_3, c_4\}$ define invariant substructures. So the idempotent mapping can be either $e_1 = (c_1)(c_2)(c_3](c_4](c_5)$ or $e_2 = (c_1](c_2](c_3)(c_4)(c_5)$. By the following lemma (again from [11]), each of these idempotents corresponds to a different submonoid of $PAut(\mu(\mathbf{o}))$.

We know that each idempotent specifies an invariant substructure of $\mu(o)$; this Lemma tells us that maximal sets of idempotents correspond to maximal invariant substructures.

For example, with the structures given in Figure 2(b) we have the maximal idempotents $e_{11} = (1)(2)(3](4](5)$, $e_{12} = (1](2](3)(4)(5]$, $e_{21} = (1](2](3)(4)(5)$, $e_{22} = (1)(2)(3](4](5]$, and the corresponding submonoid of $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$

$G_{11} = \{(1](2](3](4](5], (1, 2](3](4](5), (2, 1](3](4](5), (1)(2)(3](4](5)\}$,
$G_{12} = \{(1](2](3](4](5], (1](2](3)(4)(5]\}$,
$G_{21} = \{(1](2](3](4](5], (1](2](3, 4](5), (1](2](4, 3](5), (1](2](3)(4)(5)\}$,
$G_{22} = \{(1](2](3](4](5], (1)(2)(3](4](5]\}$.

This example leads to the following notion:

Definition 9 *Let $\mathcal{M}$ be a model of a domain state ontology.*

A scaffold $\mathfrak{G}^R_{\mathbf{o}}$ of an activity occurrence $\mathbf{o} \in \Gamma$ is a set consisting of sets of submonoids of $PIso(\mu(\mathbf{o}), \eta(\mathbf{o}))$ such that for all $\mathbb{G}_i \in \mathfrak{G}^R_{\mathbf{o}}$

1. *For all $G_{ij} \in \mathbb{G}_i$,*
   $G_{ij} = (\mathbf{e}_{ij} PAut(\mu(\mathbf{o}))\mathbf{e}_{ij})$
   such that $\mathbf{e}_{ij}$ is the identity element of G_{ij};
2. *The identity element $\mathbf{e}_{ij}$ of each $G_{ij} \in \mathbb{G}_i$ has a maximal domain; that is, there is no other identity mapping $\mathbf{e}'$ that satisfies property 1 and $dom(\mathbf{e}_{ij}) \subset dom(\mathbf{e}')$;*
3. *If $\mathbb{G}_i = \{G_{i1}, \ldots, G_{im}\}$ then*
   $dom(\mathbf{e}_{i1}) \cup \cdots \cup dom(\mathbf{e}_{im}) \subseteq dom(\mu(\mathbf{o}))$ and
   $dom(\mathbf{e}_{i1}) \cap \cdots \cap dom(\mathbf{e}_{im}) = \varnothing$.

[7] An idempotent in a monoid is an element x such that $(x \cdot x) = x$.

Property 1 ensures that elements in $\mathfrak{G}^R_{\mathbf{o}}$ are the maximal submonoids associated with $PAut(\mu(\mathbf{o}))$. Note that $\mathfrak{G}^R_{\mathbf{o}}$ is defined as a *set* of sets of submonoids because an activity occurrence can be specified with more than one set of idempotents, and therefore, with more than one set of submonoids (such as the one shown in Figure 2(b)).

Each idempotent in a submonoid in the scaffold $\mathfrak{G}^R_{\mathbf{o}}$ specifies an invariant substructure of $\mu(\mathbf{o})$; those which are not a subset of any other idempotent specify the maximal invariant substructures. In addition, condition 2 enforces the idempotents of a set (and consequently their corresponding substructures) to be disjoint, while condition 3 guarantees that they cover the domain of $\mu(\mathbf{o})$. In other words, a set of corresponding idempotents of $\mu(\mathbf{o})$, $\mathbb{I} = \{e_1, \ldots, e_n\}$, breaks it into n disjoint invariant substructures $\mathcal{S}_1, \ldots, \mathcal{S}_n$ while $\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n = \mu(\mathbf{o})$.

For the preceding example, we have $\mathfrak{G}^{lt}_{\mu(\mathbf{o})} = \{\{G_{11}, G_{12}\}, \{G_{21}, G_{22}\}\}$ and $\mathfrak{G}^{comparable}_{\mu(\mathbf{o})} = \{\{G_{11}, G_{12}\}, \{G_{21}, G_{22}\}\}$.

Theorem 5 *Let $\mathcal{M}$ be a model of a domain state ontology $T_{psl} \cup D$.*

For any activity occurrence $\mathbf{o} \in \Gamma$ and each relation $\mathbf{R}$ in the signature of $T_{psl} \cup D$, there exists a unique scaffold $\mathfrak{G}^R_{\mathbf{o}}$.

Theorem 6 illustrates the connection between the changes that an activity occurrence makes in the model of the domain ontology and the set of submonoids that are associated with it.

Theorem 6 *Let $\mathcal{M}$ be a model of a domain state ontology $T_{psl} \cup D$. Let $\mathfrak{G}^{\mathbf{r}}_{\mathbf{o}}$ be the scaffold for the activity occurrence $\mathbf{o} \in \Gamma$ and the relation $\mathbf{r}$ in the signature of $T_{psl} \cup D$. For all $G_i \in \mathbb{G} \in \mathfrak{G}^{\mathbf{r}}_{\mathbf{o}}$, $\langle \mathbf{o}, \mathbf{r}(\mathbf{x_1}, \ldots, \mathbf{x_n}) \rangle \notin \mathbf{changes}^{\mathcal{M}}$ iff $\mathbf{x_1}, \ldots, \mathbf{x_n} \in dom(I_{G_i})$.*

In simple words, a submonoid in a scaffold $\mathfrak{G}^{\mathbf{r}}_{\mathbf{o}}$ (associated with an activity occurrence $\mathbf{o}$) only contains elements that $\mathbf{o}$ preserves fluent $\mathbf{r}$ between them.

6. Classifying Activities

For any two activity occurrences $\mathbf{o_1}, \mathbf{o_2}$ in a model of a domain state ontology such that $\mu(\mathbf{o_1}) \cong \mu(\mathbf{o_2})$ ($\cong$ denotes isomorphism), the models $\eta(\mathbf{o_1})$ and $\eta(\mathbf{o_2})$ are isomorphic iff $\mathbf{o_1}$ and $\mathbf{o_2}$ achieve, falsify, and preserve the same types of fluents. In the other words, if $\eta(\mathbf{o_1}) \ncong \eta(\mathbf{o_2})$, then there should be at least one class of activities that $\mathbf{a_1}$ belongs to and $\mathbf{a_2}$ does not ($\mathbf{a_1}$ and $\mathbf{a_2}$ are respective types of activities for $\mathbf{o_1}$ and $\mathbf{o_2}$). Consequently, the axiomatization of the activity classes is based on the criterion that if $\eta(\mathbf{o_1}) \cong \eta(\mathbf{o_2})$, then $\mathbf{o_1}$ and $\mathbf{o_2}$ are occurrences of activities in the same class. In the next phase of the methodology for designing the domain process ontology, we prove a classification theorem for activities by identifying the minimal set of scaffolds for occurrences $\mathbf{o_1}$ and $\mathbf{o_2}$ (of the same activity) which are needed to guarantee that the two models $\eta(\mathbf{o_1})$ and $\eta(\mathbf{o_2})$ are isomorphic.

Since the scaffolds for all primitive relations in the domain ontology must always be included, it is primarily the defined relations that are the focus of the classification theorem. In the case of chains, we have the following:

Theorem 7 *Let $\mathcal{M}$ be a model of a domain state ontology.*

For any activity occurrences $\mathbf{o_1}, \mathbf{o_2} \in \Gamma$ with $\mu(\mathbf{o_1}) \cong \mu(\mathbf{o_2})$, if $\mathfrak{G}^{point}_{\mathbf{o_1}} = \mathfrak{G}^{point}_{\mathbf{o_2}}$, $\mathfrak{G}^{lt}_{\mathbf{o_1}} = \mathfrak{G}^{lt}_{\mathbf{o_2}}$ and, $\mathfrak{G}^{comparable}_{\mathbf{o_1}} = \mathfrak{G}^{comparable}_{\mathbf{o_2}}$, then $\eta(\mathbf{o_1}) \cong \eta(\mathbf{o_2})$.

Intuitively, besides creating or destroying points, there are three ways to change a chain structure. Figure 2(c) shows the possible changes with respect to the point c_4. The activity occurrence o_1 does not change the relation between c_3 and c_4, but they are no longer comparable with c_1 and c_2. The activity occurrences o_2 and o_3 change the *lt* relation for c_4 with c_1, c_2 and c_3; however, while they are still comparable with respect to o_2, c_3 and c_4 are not comparable with c_1 and c_2 with respect to o_3. It is easy to see from this example that in addition to the primitive relations of $T_{stchain}$ (i.e. *point* and *lt*), the defined relation *comparable* must also be considered in characterization of the Chain Transform Process Ontology.

Once we know the minimal set of scaffolds needed to characterize isomorphic models, the next step is to use the relationships between the different scaffolds to provide a model-theoretic definition for each class of activities.

For the Chain Transform Process Ontology, we have the following:

Lemma 3 *For each $G_i \in \mathfrak{G}^{lt}_{\mathbf{o}}$, there exists $H_j \in \mathfrak{G}^{comparable}_{\mathbf{o}}$ such that $\varepsilon(G_i) \subseteq \varepsilon(H_j)$.*

Theorem 7 and Lemma 3 lead to the following definition for the intended models of the Chain Transform Process Ontology. Each class of activities is defined with respect to properties of the scaffolds.

Definition 10 *$\mathfrak{M}^{chainprocess}$ is the class of structures such that $\mathcal{M} \in \mathfrak{M}^{chainprocess}$ iff*

1. *there exists $\mathcal{N} \in \mathfrak{M}^{stchain}$ such that $\mathcal{N} \subset \mathcal{M}$,*
2. *for every $\mathbf{o} \in \Gamma$, $\mathfrak{G}^{point}_{\mathbf{o}} \neq \mathcal{I}$ or $\mathfrak{G}^{lt}_{\mathbf{o}} \neq \mathcal{I}$,*
3. *$\langle \mathbf{a} \rangle \in$ **preserve_point** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\mathfrak{G}^{point}_{\mathbf{o}} = \mathcal{I}$,*
4. *$\langle \mathbf{a} \rangle \in$ **change_point** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\mathfrak{G}^{point}_{\mathbf{o}} \neq \mathcal{I}$,*
5. *$\langle \mathbf{a} \rangle \in$ **create_point** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\mathfrak{G}^{point}_{\mathbf{o}} \neq \mathcal{I}$ and $\mathbf{point}^{\mu(\mathbf{o})} \subset \mathbf{point}^{\eta(\mathbf{o})}$,*
6. *$\langle \mathbf{a} \rangle \in$ **destroy_point** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\mathfrak{G}^{point}_{\mathbf{o}} \neq \mathcal{I}$ and $\mathbf{point}^{\eta(\mathbf{o})} \subset \mathbf{point}^{\mu(\mathbf{o})}$,*
7. *$\langle \mathbf{a} \rangle \in$ **newchain** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, for each $G \in \mathfrak{G}^{lt}_{\mathbf{o}}$ there exists $H \in \mathfrak{G}^{comparable}_{\mathbf{o}}$ such that $\varepsilon(G) = \varepsilon(H)$,*
8. *$\langle \mathbf{a} \rangle \in$ **make_chain** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\langle \mathbf{a} \rangle \in$ **newchain** and $\mathbf{comparable}^{\mu(\mathbf{o})} \subset \mathbf{comparable}^{\eta(\mathbf{o})}$,*
9. *$\langle \mathbf{a} \rangle \in$ **break_chain** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, $\langle \mathbf{a} \rangle \in$ **newchain** and $\mathbf{comparable}^{\eta(\mathbf{o})} \subset \mathbf{comparable}^{\mu(\mathbf{o})}$,*
10. *$\langle \mathbf{a} \rangle \in$ **rearrange** iff for every occurrence $\mathbf{o}$ of the activity $\mathbf{a}$, for each $G \in \mathfrak{G}^{lt}_{\mathbf{o}}$ there exists $H \in \mathfrak{G}^{comparable}_{\mathbf{o}}$ such that $\varepsilon(G) \subset \varepsilon(H)$,*

where $\mathcal{I} = \{\{I_{\mu(\mathbf{o})}\}\}$.

The classes **preserve_point** and **change_point** are defined with respect to whether or not the **point** fluent is changed. The two subclasses **create_point** and **destroy_point** are defined with respect to *how* the fluent is changed (i.e. either achieved or falsified). This is also how the subclasses **make_chain** and **break_chain** of the class **newchain** are defined.

$$((\forall o)\ arboreal(o) \supset ((\exists x, y) changes(o, lt(x, y)) \vee (\exists x) changes(o, point(x))). \quad (12)$$

$$(\forall a)\ preserve\_point(a) \equiv ((\forall o, x)\ occurrence\_of(o, a) \supset \neg changes(o, point(x))). \quad (13)$$

$$(\forall a)\ change\_point(a) \equiv ((\forall o)\ occurrence\_of(o, a) \supset (\exists x)\ changes(o, point(x))). \quad (14)$$

$$(\forall a)\ create\_point(a) \equiv ((\forall o)\ occurrence\_of(o, a) \supset (\exists x)\ achieves(o, point(x))). \quad (15)$$

$$(\forall a)\ destroy\_point(a) \equiv ((\forall o)\ occurrence\_of(o, a) \supset (\exists x)\ falsifies(o, point(x))). \quad (16)$$

$$(\forall a)\ newchain(a) \equiv ((\forall o, x, y)\ (occurrence\_of(o, a) \wedge changes(o, lt(x, y))) \supset changes(o, comparable(x, y))). \quad (17)$$

$$(\forall a)\ make\_chain(a) \equiv ((\forall o, x, y)\ (occurrence\_of(o, a) \wedge achieves(o, lt(x, y))) \supset achieves(o, comparable(x, y))). \quad (18)$$

$$(\forall a)\ break\_chain(a) \equiv ((\forall o, x, y)\ (occurrence\_of(o, a) \wedge falsifies(o, lt(x, y))) \supset falsifies(o, comparable(x, y))). \quad (19)$$

$$(\forall a)\ rearrange(a) \equiv ((\forall o, x, y)\ (occurrence\_of(o, a) \wedge changes(o, lt(x, y))) \supset \neg changes(o, comparable(x, y))). \quad (20)$$

Figure 3. Axioms for $T_{chainprocess}$.

7. Axiomatization of Activity Classes

In the final step of methodology we axiomatize the activity classes specified in the intended models. The axioms in Figure 3 constitute the Chain Transform Process Ontology $T_{chainprocess}$ that axiomatizes the class of structures $\mathfrak{M}^{chainprocess}$. Axiom 12 guarantees that all legal activity occurrences change some fluent (this corresponds to Condition 2 in Definition 10) Axioms 13, 15 and 16 define classes of activities that preserve, create or destroy points. Axiom 17 defines class of activities that change the comparability of two points in addition to their *lt* relation, while Axiom 20 defines activities which preserve the *comparable* relation.

Theorem 8 is the representation theorem for the Chain Transform ontology.

Theorem 8 *$\mathcal{M}$ is a model of $T_{chainprocess} \cup T_{stchain} \cup T_{psl}$ iff it is isomorphic to a structure in $\mathfrak{M}^{chainprocess}$.*

8. Relationship to Basic Action Theories

So far we discussed about building a process ontology based an existing domain ontology and presenting Chain Transform ontology as an example. Using this example, we show that models of the process ontology can be applied to evaluate the ontologies are used in basic action theory for similar domains.

The following theorem shows that *move* and *moveToTable* actions in Reiter's axiomatization of Blocks World [3] are newchain activities.

Theorem 9 *Suppose Σ_{BW} be the set of successor state axioms for Reiter's axiomatization of Blocks World. Then we have:*

$$T_{chainprocess} \cup T_{stchain} \cup T_{psl} \cup \Sigma_{BW} \models$$
$$(\forall x, y) newchain(move(x, y)) \wedge (\forall z) newchain(moveToTable(z)).$$

Note that in our approach, the effects of an activity cannot violate any of the axioms of the domain ontology. For example, no activity can have the effect of splitting an element in a chain. As a result, precondition axioms are simply restrictions on the set of possible activities that are allowed to occur in a particular application.

9. Recap

The methodology for the design and verification of domain-specific process ontologies for a given domain ontology can be summarized as follows:

- specify the axiomatization of a domain state ontology that is definably equivalent to the domain ontology;
- show that the models of the domain state ontology are characterized by an injective mapping between the states in a model of the domain state ontology and the models of the domain ontology;
- characterize models of the domain process ontology with respect to submonoids of partial automorphisms of models of the domain ontology;
- prove a classification theorem for primitive activities, which implies the completeness of the domain process ontology, with respect to submonoids of partial automorphisms of models of the domain ontology;
- axiomatize classes of primitive activities with respect to the classification theorem and prove the representation theorem for the resulted taxonomy which implies correctness of the domain process ontology.

10. Summary and Future Work

In this paper we have introduced a general methodology for designing and verifying process ontologies with respect to the partial automorphisms of models of their underlying domain ontology. For any structure in some class of structures, we can define the unique monoid of partial automorphisms. From this monoid, we can define all possible activities and their effects. The completeness of the axioms is determined by identifying the set of activity classes using sets of submonoids of partial automorphisms. The correctness of the axioms is determined by examining whether they satisfy relations between submonoid of partial automorphisms.

The methodology can be applied either to design new process ontologies for existing domain ontologies or to characterize and evaluate existing process ontologies. It is applicable in any domain that has a mathematical characterization. If a model-characterization

for the domain exists, effect of each activity can be represented with a partial automorphism and the set of all partial automorphisms of a structure forms a monoid.

Our next step will be to explore process ontologies that use the mereotopology of RCC as the domain ontology. Since the models of RCC can be represented by boolean lattices, we can use existing characterizations of the partial automorphisms of boolean lattices to specify axiomatizations for domain process ontologies such as Conceptual Neighborhoods [20].

The approach proposed in this paper is used to axiomatize different classes of primitive activities within a specific domain. Another direction for future work will be to use the theory of complex activities within the PSL Ontology to characterize the sets of plans that can be specified using the domain process ontologies.

References

[1] A. Galton. Experience and History: Processes and their Relation to Events. *Journal of Logic and Computation*, 18(3):323–340, 2008.

[2] M. Gruninger. Ontology of the Process Specification Language. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*. Springer-Verlag, 2003.

[3] R. Reiter. *Knowledge in Action:Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge,MA, 2001.

[4] M. Gruninger, T. Hahmann, A. Hashemi, and D. Ong. Ontology Verification with Repositories. In *Proc. of the 6th FOIS Int. Conference*, pages 317–330, Toronto,Canada, 2010. IOS Press.

[5] F. Dylla and R. Moratz. Exploiting Qualitative Spatial Neighborhoods in the Situation Calculus. In *Spatial Cognition IV, LNAI 3343*, pages 304–322, Berlin,Germany, 2004. Springer-Verlag.

[6] C. Vidal and A. Rodriguez. A Logical Approach for Modeling Spatio-temporal Objects and Events. In *Proc. of ER Workshops, LNCS 3770*, pages 218–227, Berlin,Germany, 2005. Springer-Verlag.

[7] P. Grenon and B. Smith. SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition and Computation*, 4(1):69–104, 2004.

[8] M. Bhatt and S. Loke. Modelling Dynamic Spatial Systems in the Situation Calculus. *Spatial Cognition and Computation*, 8(1):86–130, 2008.

[9] Y. Gu and M. Soutchanski. Reasoning about large taxonomies of actions. In *Proc. 23rd Conf. on Artificial intelligence (AAAI-08)*, pages 931–937, 2008.

[10] O. Ganyushkin and V. Mazorchuk. On the Structure of IO_n. *Semigroup Forum*, 66(3):455–483, 2008.

[11] O. Ganyushkin and V. Mazorchuk. *Classical Finite Transformation Semigroups: an Introduction*. Springer, UK, 2009.

[12] D. Randell, Z. Cui, and A. Cohn. A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation*, pages 165–176, Cambridge, MA, 1992. Morgan Kaufmann.

[13] S. Cook and Y. Liu. A Complete Axiomatization for Blocks World. *Journal of Logic and Computation*, 13(4):581–594, 2002.

[14] H.D. Ebbinghaus and J. Flum. *Finite Model Theory, Second Edition*. Springer, 1999.

[15] S. Lipscomb. Symmetric Inverse Semigroups. In *Mathematical Surveys and Monographs*, volume 46. American Math. Society, Providence,RI, 1996.

[16] V. H. Fernandes. The Monoid of All Injective Orientation Preserving Partial Transformations on a Finite Chain. *Communications in Algebra*, 28(7):3401–3426, 2000.

[17] V. H. Fernandes. The Monoid of All Injective Order Preserving Partial Transformations on a Finite Chain. *Semigroup Forum*, 62(2):178–204, 2001.

[18] J. Chubba, V. S. Harizanov, A. S. Morozov, S. Pingrey, and E. Uffermanc. Partial Automorphism Semigroups. *Annals of Pure and Applied Logic*, 156(2-3):245–258, 2008.

[19] M. Rubin. On the Automorphism Groups of Countable Boolean Algebras. *Israel Journal of Mathematics*, 35(1-2):151–170, 1980.

[20] C. Freksa. Conceptual Neighborhood and its role in temporal and spatial reasoning. In *In Proc. of the IMACS Workshop*, pages 181–187, Toulouse, France, 1991b. North-Holland.