SEMANTIC INTEGRATION THROUGH INVARIANTS

MICHAEL GRÜNINGER AND JOSEPH B. KOPENA

1. INTRODUCTION

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.

Current approaches to semantic integration do not fully exploit the model-theoretic structures underlying ontologies. These approaches are typically based on the taxonomic structure of the terminology ([12], [13]) or heuristics-based comparisons of the symbols of the terminology ([1, 9]). Such techniques are well-suited to working with many ontologies currently under development, most of which define a terminology with minimal formal grounding and a set of possible models that does not contain a rich set of features and properties.

However, automated and correct approaches to semantic integration will require ontologies with a deeper formal grounding so that decisions may be made by autonomous software when comparing ontologies for integration. This article presents an approach toward this goal using techniques based on the development of strong ontologies with terminologies grounded in properties of the underlying possible models. With these as inputs, semi-automated and automated components may be used to create mappings between ontologies and perform translations.

The Process Specification Language (PSL) ([6], [7]) is used in this article to demonstrate this approach to ontology construction and integration. PSL consists of a core ontology which outlines basic objects that exist in the domain, and a multitude of definitional extensions that provide a rich terminology for describing process knowledge. These extensions are based on invariants, properties preserved by isomorphism, which partition the first-order models of the core ontology. Using these invariants, semantic mappings between application ontologies and PSL may be semi-automatically generated. In addition, the direct relationship between the PSL terminology and the invariants improves the ability to verify the generated results. These semantic mappings may then be used to perform integration between applications or ontologies. They may also be used to analyze the application as well as to bootstrap an ontology to those applications which do not have an associated, explicit, formal ontology.

2. An Architecture for Semantic Integration

This section describes the Interlingua Architecture, the basic approach to application integration employed in this work. Semantic integration is then presented in terms of this architecture as the tasks and questions which must be performed and answered.

2.1. **The Interlingua Architecture.**

Informally, semantic mappings express the meaning of a term from one ontology in terms of another ontology; each such mapping may simply link one term to another or may specify a complex transformation. More formally, semantic mappings can be characterized by the notion of definable interpretation ([10]): If $\mathscr{N}$ is a structure in the language $\mathscr{L}_0$ and $\mathscr{M}$ is a structure in the language $\mathscr{L}$, then we say that $\mathscr{N}$ is definably interpretable in $\mathscr{M}$ if we can interpret the symbols of $\mathscr{L}_0$ so that there exists a substructure $\mathscr{M}$ that is isomorphic to $\mathscr{N}$. Semantic mappings are the sentences that axiomatize this interpretation. The techniques that we discuss in this paper semi-automatically generate such semantic mappings by using human input to identify properties of the models that will be preserved by isomorphism.

In current practice, semantic mappings are manually generated directly between application ontologies. However, for software applications operating in open environments such as the Semantic Web, it cannot be assumed that mappings have been generated prior to interaction between applications. In [8], a number of architectures have been proposed to support semantic integration in such an open environment. Each architecture is distinguished by the origins of the semantic mappings, the existence of a mediating ontology, and the degree of agreement that exists among the anticipated community of interacting software.

The *Interlingua Architecture* is adopted within this work, the distinguishing feature of which is the existence of a mediating ontology that is independent of the applications' ontologies and is used as a neutral interchange ontology ([2]). Semantic mappings between application and interlingua ontologies are manually generated and verified prior to application interactions [3]. This process of creating the mapping between the application ontology and the interlingua ontology is identical to the process of creating a mapping directly between two application ontologies, the key difference of this approach being that application ontologies are integrated with the interlingua rather than each other.

The most obvious property of this approach is the dramatic reduction of the number of translators which must be constructed. The manual, point-to-point approach requires on the order of n^2 translators, one for each pairing, while the interlingua approach mandates only one translator per application. In addition to the initial costly development of a translator for each pairing under the point-to-point approach, if one application's ontology is changed, each associated translator must also be updated. Using an interlingua, only the translator to and from the interlingua must be maintained for each application[1]. A demonstration of these properties from the domain of systems for managing manufacturing processes is shown in Figure 1.

Importantly, the point-to-point approach does not work in environments which feature unanticipated software interactions. Interaction can only occur between pairs of software for which a specific translator has been previously developed. Using the interlingua model, a mapping between the application ontology and the interlingua is all that is necessary to interact with the community of software for which mappings to and from the interlingua

[1] See [14] for a more detailed discussion of the tradeoffs between the point-to-point and interlingua approaches.

(a) Point-to-point translation requires on the order of n^2 translators be developed and maintained.

(b) With an interlingua, only n translators must be developed and maintained.

FIGURE 1. Translation pairings for a set of manufacturing process systems.

have also been developed. This eliminates the problem of changes in applications mandating changes to all other systems, and allows existing software to seamlessly interoperate with newly introduced applications, capabilities not possible using manual, point-to-point mappings.

2.2. **Integration and Translation.** Under the Interlingua Architecture, there are two steps in translation: the execution of the mapping from the application ontology to the interlingua and subsequently from the interlingua to the target application's ontology. If the application ontologies and the interlingua ontology are specified using the same logical language, then translation can be accomplished by applying deduction to the axioms of the interlingua ontology in conjunction with the formal mapping rules ([3], [2]). In effect, a direct mapping rule from one application's ontology to the target application's ontology is inferred from the two separate rules. If these mapping rules have been verified to preserve semantics between the application and interlingua ontology, it is guaranteed that this translation between the applications also preserves semantics.

An important question is then whether the existence of the pre-defined mappings between the application ontologies and the interlingua ontology enables the automatic generation of a point-to-point mapping between the applications' ontologies. More formally, if $\mathscr{M}_1$ and $\mathscr{M}_2$ are both definably interpretable in $\mathscr{N}$, is $\mathscr{M}_1$ definably interpretable in $\mathscr{M}_2$? Answering this question is equivalent to the task of semantic integration within the Interlingua Architecture. It is addressed in this work by comparing the mappings between application ontologies and the interlingua.

3. INVARIANT-BASED ONTOLOGY DESIGN

Many ontologies are specified as taxonomies or class hierarchies, yet few provide formal justification for their classification scheme. If we consider ontologies of mathematical structures, we see that logicians classify models by using properties of models, known as *invariants*, that are preserved by isomorphism.

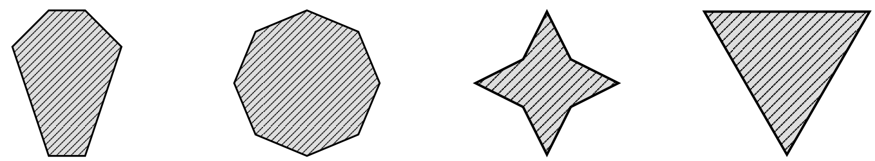
For some classes of structures, invariants can be used to classify the structures up to isomorphism; for example, vector spaces can be classified up to isomorphism by their dimension. For other classes of structures, such as graphs, it is not possible to formulate a complete set of invariants. However, even without a complete set, invariants can still be used to provide a classification of the models of a theory. Figure 2 provides such an

- Is the shape a polygon with $n \geq 3$ sides?
- Is the shape convex?
- Is the symmetry group consisting of rotations and reflections of the shape equivalent to D_n?

(a) Several invariant properties of geometric shapes.

Regular polygons $\equiv$ convex polygons w/ $n \geq 3$ sides and symmetry group $\equiv D_n$.

(b) Definition for the class of regular shapes using the above invariants.



yes, $n = 6$;	yes, $n = 8$;	yes, $n = 8$;	yes, $n = 3$;
convex;	convex;	concave;	convex;
$Symm \not\equiv D_n$	$Symm \equiv D_n$	$Symm \not\equiv D_n$	$Symm \equiv D_n$
$\supset \neg$ regular	$\supset$ regular	$\supset \neg$ regular	$\supset$ regular

(c) Several shapes classified as regular or irregular through comparison to the definition.

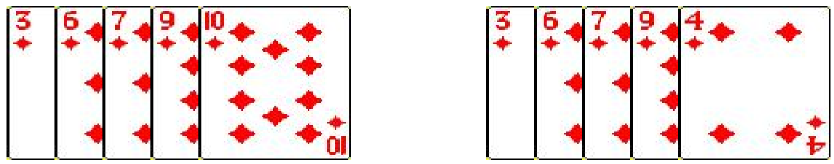
FIGURE 2. The use of invariants in constructing a terminology of geometric shapes. Although not a complete set, these invariants do support the formal definition of terms in the language.

example from the domain of geometric shapes. Some invariants of objects in this domain are given in Figure 2(a). These are used in Figure 2(b) to define the class of regular shapes. Several shapes are classified against this definition and the results given in Figure 2(c).

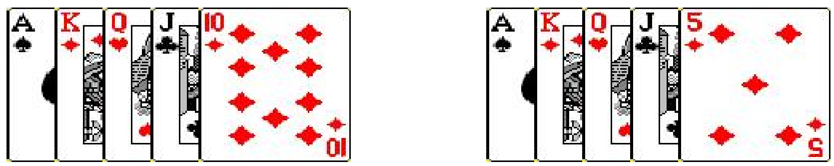
Notice that each question in Figure 2(a) corresponds to an invariant for an object, and each value of the invariant is a possible answer to the question, as in Figure 2(c). We will later use this same correspondence between invaraints and questions to specify semantic mappings for ontologies such as PSL.

Of particular interest in this example is the invariant that is the symmetry group of the object. In this case, symmetry is the preservation of the shape of the object even after we rotate or reflect it along an axis. If we take a triangle and rotate it about its center through an angle of 120°, the resulting figure looks exactly the same as when it started. Similarly, the figure looks the same when reflecting it about a line that contains a vertex and bisects the opposite edge.

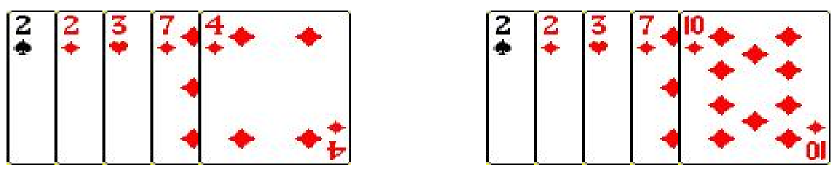
For models of ontologies such as PSL, the symmetries are more abstract, but the basic idea remains – some structure within a model of the ontology will be preserved even after subjecting it to some sort of transformation. The invariants that are used in ontology design are therefore generalizations of symmetry groups.



(a) Any transformation that preserves the suit also preserves a flush. In this example, the 10♢ is exchanged with the 4♢.



(b) No transformation that preserves the suit also preserves a royal straight. In this example, the 10♢ is exchanged with the 5♢.



(c) Some transformations that preserve the suit also preserve a pair. In this example, the 4♢ is exchanged with the 10♢.

FIGURE 3. Poker hands can be classified by considering the sets of transformation that preserve the hand. In this example, we we exchange cards of the same rank, but they are allowed to be of different suits.

To illustrate how invariants are used to provide the classification and terminology of an ontology, we will consider the treatment of preconditions in the PSL Ontology. Preconditions specify the constraints under which activities can possibly occur in some domain. Within the PSL Ontology, occurrence trees characterize all sequences of activity occurrences; however, not all of these sequences will intuitively be physically possible within the domain. Consequently, we need to characterize the subtree of an occurrence tree that consists only of possible sequences of activity occurrences; such a subtree is referred to as a legal occurrence tree, and elements of this subtree are referred to as *legal activity occurrences*.

The most prevalent class of occurrence constraints is that of *markovian* activities, activities whose preconditions depend only on the state prior to their occurrences (e.g., to withdraw money from a bank account, there must be sufficient funds in the account). The class of markovian activities is defined in the PSL definitional extension $state\_precond.def$, a portion of which is given in Figure 4. There are also activities whose preconditions intuitively are not markovian, but depend on the time at which the activity occurs (e.g.,

Transformation	Poker	Preconditions
All transformations of some kind preserve legality	flush	*markov_precond*
A subset of transformations of some kind preserve legality	pair	*partial_state*
No transformations of some kind preserve legality	royal straight	*rigid_state*

TABLE 1. Analogy between the one kind of transformation that preserves legal poker hands and the permutations that preserve legal activity occurrences.

transactions must be completed during office hours), and any process ontology should be able to capture these constraints as well.

The invariant that is associated with markovian preconditions can be illustrated by the symmetries of poker hands. Poker is played with a standard pack of fifty two cards, which are ranked Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2; for each rank, there are four suits – ♠, ♡, ♢, and ♣. Although there are ten possible poker hands, we will focus on three of these hands. A *flush* is a hand where all of the cards are the same suit, e.g., all cards have the ♡ suit. A *royal straight* is the sequence Ace-King-Queen-Jack-10, regardless of the suit. With a *pair*, there are two cards of any rank, matched with three distinct cards.

We can classify poker hands by characterizing which of them are preserved by different kinds of transformations (see Figure 3). In one kind of transformation, we change the suit of a single card, but we must preserve the rank, e.g., change a 3♣ into a 3♡. In another kind of transformation, we change the rank of a single card, but we must preserve the suit, e.g., change a 3♣ into a 7♣. The first kind of transformation will always preserve a royal straight but it will never preserve a flush, while the second kind of transformation will always preserve a flush but never preserve a royal straight. There exist transformations of either kind that will preserve a pair, provided that the rank of the changed card is not the rank of one of the cards in the pair.

The classification of activities with respect to preconditions is analogous to this card game (see Table 1). Rather than preserve poker hands, we want to characterize which permutations of activity occurrences within a model of the PSL Ontology preserve legal occurrences of activities in an occurrence tree. Rather than change cards with the same suit, we consider permutations of activity occurrences within a model that agree on the set of fluents that hold prior to the activity occurrences in an occurrence tree. The invariant in this case is the group of such permutations that preserve the legal occurrences of the activity. If any such permutation will preserve legal occurrence, then the activity is the *markov_precond* class, as axiomatized in Figure 4. With a *partial_state* activity, If only a subset of such permutations will preserve legal occurrences, then there exist additional non-markovian constraints on the legal occurrences of the activity, and this is axiomatized by the *partial_state* class in Figure 4. If no such permutation will preserve legal occurrences, then the constraints on the legal occurrences of the activity are completely non-markovian; this is axiomatized by the *rigid_state* class in Figure 4.

In general, the set of models for the core theories of an ontology are partitioned into equivalence classes defined with respect to the set of invariants of the models. Each equivalence class in the classification of the models of the ontology is axiomatized using a definitional extension of the ontology. Each definitional extension in the ontology is associated

(1)
$$(\forall o_1, o_2)\, state\_equiv(o_1, o_2) \equiv$$
$$(\forall f)\,(prior(f, o_1) \equiv prior(f, o_2))$$

(2)
$$(\forall a, o_1, o_2)\, poss\_equiv(a, o_1, o_2) \equiv$$
$$(poss(a, o_1) \equiv poss(a, o_2))$$

(3)
$$(\forall a)\, markov\_precond(a) \equiv$$
$$((\forall o_1, o_2)\, state\_equiv(o_1, o_2) \supset poss\_equiv(a, o_1, o_2))$$

(4)
$$(\forall a)\, partial\_state(a) \equiv$$
$$(\exists o_1)\,((\forall o_2)\, state\_equiv(o_1, o_2) \supset poss\_equiv(a, o_1, o_2))$$
$$\wedge (\exists o_3, o_4)\, state\_equiv(o_3, o_4) \wedge \neg poss\_equiv(a, o_3, o_4)$$

(5)
$$(\forall a)\, rigid\_state(a) \equiv$$
$$(\forall o_1)(\exists o_2)\, state\_equiv(o_1, o_2) \wedge \neg poss\_equiv(a, o_1, o_2)$$

FIGURE 4. Classes of activities with state-based preconditions from the definitional extension *state_precond.def*. The additional relations are defined to capture the different transformations used to determine the symmetries. Two activity occurrences o_1, o_2 are *state_equiv* iff there exists a permutation of activity occurrences that preserves the fluents that hold prior to the activity occurrences. The two activity occurrences are *poss_equiv* iff there exists a permutation of activity occurrences that preserves legal occurrences of the activity.

with a unique invariant; the different classes of activities or objects that are defined in an extension correspond to different properties of the invariant. In this way, the terminology of the ontology arises from the classification of the models of the core theories with respect to sets of invariants.

4. SEMANTIC MAPPING VIA TRANSLATION DEFINITIONS

As noted in Section 2, the generation of semantic mappings between two ontologies T_1 and T_2 is equivalent to the formal problem of determining whether T_1 is definably interpretable in T_2. Although in general an extremely difficult problem, the invariants used in the classification of the models of the ontologies can also be used to generate semantic mappings. Semantic mappings preserve models—each model of the ontology T_1 is mapped to an isomorphic substructure of a model of the ontology T_2. Since invariants are properties of the models that are preserved by isomorphism, semantic mappings must also preserve the invariants. Therefore, if models of T_1 and T_2 are characterized up to isomorphism by some sets of invariants, then T_1 is definably interpretable in T_2 iff there is a mapping of the invariants of T_1 to the invariants of T_2; a concept in T_1 will be mapped to a concept in T_2 iff the invariants have the same values.

Translation definitions specify the semantic mappings between the interlingua ontology and application ontologies. Following the above discussion, they are generated using the organization of the definitional extensions, each of which corresponds to a different invariant. Every class of activity, activity occurrence, or fluent in an extension corresponds to a different value for the invariant. The consequent of a translation definition is equivalent to the list of invariant values for members of the application ontology class.

Translation definitions have a special syntactic form—they are biconditionals in which the antecedent is a class in the application ontology and the consequent is a formula that uses only the lexicon of the interlingua ontology. For example, the concept of *AtomicProcess* in the OWL-S Ontology [2] ([11]) has the following translation definition with respect to the PSL Ontology:

(forall (?a)
(iff (AtomicProcess ?a)
  (and (atomic ?a)
    ($markov\_precond$?a)
    ($markov\_effects$?a))))

The invariant corresponding to the $markov\_precond$ class was discussed in the preceding section; the invariants corresponding to the $markov\_effects$ and $context\_free$ classes are based on groups consisting of permutations of activity occurrences that preserve effects (i.e., fluents that are achieved or falsified by activity occurrences).

4.1. **Semi-Automatic Generation of Semantic Mappings.** The generation of semantic mappings through the specification of invariant values has been implemented in the PSL project's Twenty Questions mapping tool [3]. Each question corresponds to an invariant, and each value of the invariant is a possible answer to the question. Any particular activity, activity occurrence, or fluent will have a unique value for the invariant; however, if we are mapping a class of activities, occurrences, or fluents from some application ontology, then different members of the class may have different values for the same invariant. In such a case, one would respond to a question by supplying multiple answers. By guiding and supporting users in creating translation definitions without requiring them to work directly with first order logic axiomatizations, the Twenty Questions tool provides a semi-automated technique for creating semantic mappings.

Figure 5 gives a sample question corresponding to the symmetries of fluents and legal activity occurrences; each possible answer corresponds to a different value of the invariant, which is the group of permutations that preserve legal activity occurrences. Following the axiomatizations given in Figure 4 for the classes of activities corresponding to these values, selecting the first answer would generate the translation definition:

```
(forall (?a)
        (iff    (myclass ?a)
                (markov_precond ?a)))
```

Selecting the first two answers would give the translation definition:

```
(forall (?a)
        (iff    (myclass ?a)
                (or     (markov_precond ?a)
                        (partial_state ?a))))
```

In this latter case, some activities in *myclass* will have Markov preconditions while other activities will not.

---

[2] OWL-S is an OWL (Ontology Web Language) ontology for describing Web services, created by a coalition of researchers through the support of the DARPA Agent Markup Language (DAML) program. OWL-S supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form.

[3] Available at `http://ats.nist.gov/psl/twenty.html`.

2. Constraints on Atomic Activity Occurrences based on State

Are the constraints on the occurrence of the atomic activity based only on the state prior to the activity occurrence?

- ☐ Any occurrence of the activity depends only on fluents that hold prior to the activity occurrence.
- ☐ Some (but not all) occurrences of the activity depend only on fluents that hold prior to the activity occurrence.
- ☐ There is no relationship between occurrences of the activity and the fluents that hold prior to occurrences of the activity.

FIGURE 5. One of the Twenty Questions, used to classify activities with state-based preconditions.

4.2. **Validating Semantic Mappings.** The Twenty Questions tool illustrates how the classification of the models of the PSL Ontology determines the syntactic form of the translation definitions. The consequent of the translation definition specifies the values of the invariants that capture the intended semantics of the class of activities that appear in the antecedent of the translation definition. However, this raises the issue of validating the semantic mappings that are generated in this way—how can we determine the correctness of the mappings between an application ontology and the interlingua ontology? If the application ontologies are axiomatized, then we can verify the semantic mappings by proving that they do indeed preserve the models of the ontologies. This can be done in by demonstrating that the class of models of the application ontology is axiomatized by the interlingua, together with the translation definitions.

In practice, the validation of semantic mappings is complicated by the fact that few software applications have explicitly axiomatized ontologies. In such cases, the Twenty Questions tool can also be used to define a formal ontology for the software applications. This is afforded by the assumption of the *Ontological Stance* ([7]), the main tenet of which is that a software application may be modeled as if it were an inference system working on an axiomatized ontology.

The Ontological Stance is an operational characterization of the set of intended models for the application's terminology. In this sense, it should be treated as a semantic constraint on the application—it does not postulate a specific set of axioms, but rather a set of intended models. Given a software application, there exists a class of models $\mathscr{M}^A$ such that any sentence Φ is decided by the application to be satisfiable iff there exists $\mathscr{M} \in \mathscr{M}^A$ such that $\mathscr{M} \models \Phi$.

By answering the questions presented by the Twenty Questions tool, the application designer is capturing the application's set of intended models. Given correct input, the translation definitions generated by the tool together with the interlingua ontology define an explicit axiomatization of the application's previously implicit ontology.

To validate the attributed ontology, the generated translation definitions may be treated as falsifiable hypotheses and tested empirically. By the Ontological Stance, the application decides some sentence Φ to be provable iff $T_{psl} \cup T_{translation} \models \Phi$ where T_{psl} is the set of axioms for the PSL Ontology and $T_{translation}$ is the set of translation definitions that are being verified. In this way, it may be evaluated whether or not the attributed ontology correctly predicts inferences made by the software, and consequently whether or not the translation definitions accurately capture the semantics of the application.

5. Comparison of Semantic Integration Profiles for Integration

The set of translation definitions for all concepts in a software application's ontology defines a *semantic integration profile* for that application. If the interlingua has *m* invariants and each invariant *n* values, then an application profile will have the form:

```
(forall (?a)
(iff (C1-onto ?a)
(and (or (p11 ?a) ... (p1n ?a))
...
(or (pm1 ?a) ... (pmn ?a)))))
```

Each clause in the profile corresponds to a different invariant; each literal `pij ?a)` is a class of objects in the interlingua ontology, all of whose members have the same value of some invariant. For example, suppose Alice's ontology contains a class of activities `(C1-alice ?a)` which has unconstrained preconditions (i.e., they are always possible) and whose effects are either context-free or they depend only on the state prior to occurrences of the activities. Suppose that Bob's ontology contains a class of activities whose preconditions are either unconstrained or markovian and whose effects are context-free. Using the invariants for the PSL Ontology, the Twenty Questions tool would generate the following translation definitions:

```
(forall (?a)
        (iff    (Calice ?a)
                (and    (unconstrained ?a)
                        (or     (markov_effects ?a)
                                (context_free ?a)))))

(forall (?a)
        (iff    (Cbob ?a)
                (and    (context_free ?a)
                        (or     (markov_precond ?a)
                                (unconstrained ?a)))))
```

As noted in Section 2.2, translation between integration targets may be accomplished by applying deduction to the axioms of the interlingua, the semantic mappings, and the input to be translated. Given the above example mappings from the two application ontologies of Alice and Bob into PSL, the following mappings between the two concepts may be inferred:

```
(forall (?a)
(implies  (context_free ?a)
                  (implies  (Calice ?a)
                            (Cbob ?a))))

(forall (?a)
        (implies  (unconstrained ?a)
                  (implies  (Cbob ?a)
                            (Calice ?a))))
```

Thus, if an activity has context-free effects, then Bob's class of activities subsumes Alice's class; if an activity has unconstrained preconditions, then Alice's class of activities subsumes Bob's class.

Such inferred mappings will in general take the form of:

PROFILE-COMPARE($\mathbf{P_a}, \mathbf{P_b}$)
for each $C_a \in \mathbf{P_a}$
do for each $C_b \in \mathbf{P_b}$
  do $\{g_a, g_b\} \leftarrow$ CONCEPT-COMPARE(C_a, C_b)
    OUTPUT(‘$g_a \supset (C_a \supset C_b)$’)
    OUTPUT(‘$g_b \supset (C_b \supset C_a)$’)

CONCEPT-COMPARE(C_a, C_b)
$R_a \leftarrow true; R_b \leftarrow true$
for $i \leftarrow 1$ **to** m
do $s \leftarrow$ VALUES$(C_a, i) \cap$ VALUES(C_b, i)
  if $s \neq \emptyset$
   then $R_a \leftarrow$ CONJUNCTION(R_a, DISJUNCTION(s))
      $R_b \leftarrow$ CONJUNCTION(R_b, DISJUNCTION(s))
   else if VALUES$(C_a, i) \neq \emptyset \wedge$ VALUES$(C_b, i) \neq \emptyset$
      then error “No mapping.”
return $\{R_a, R_b\}$

FIGURE 6. The PROFILE-COMPARE algorithm for determining relationships between ontologies, given the semantic integration profiles.

```
(forall (?a)
(implies  (and (or (p11 ?a) ... (p1n ?a))
...
(or (pm1 ?a) ... (pmn ?a)))
  (implies  (Calicei ?a)
    (Cbobj ?a))))
```

The antecedents of these sentences can be considered to be guard conditions that determine which activities can be shared between the two ontologies. This can either be used to support direct exchange, or simply as a comparison between the application ontologies. In this example, the *alice* can export any *unconstrained* activity description to *bob* and *bob* can export any *context_free* activity description to *alice*; however, *alice* cannot import *markov_precond* activity descriptions from *bob* and *bob* cannot import any *markov_effects* activity descriptions from *alice*.

Although inferred implicitly during translation, these relationships may be explicitly determined by the simple PROFILE-COMPARE algorithm presented in Figure 6. Explicitly inferring these mappings offers several capabilities. If run-time translation efficiency is important, then these point-to-point mapping rules could be generated upon first interaction and then cached as explicit rules to be used in subsequent interactions. A detailed discussion of such tradeoffs and overlaps between point-to-point and interlingua-based integration approaches is presented in [14].

In addition, by explicitly generating such mappings, it may be possible to use simpler inference engines to perform translation, rather than requiring a full first order reasoner to implicitly translate using axioms of the interlingua, the semantic mappings, and the input to be translated. Importantly, such explicit mappings may also be used by the application designers to examine the structure of their application as well as to to evaluate relationships and coverage relative to the interlingua or other ontologies.

6. Open Problems

Several important issues related to semantic integration have not been addressed so far in this work, including:

- **Translation Definitions for Primitive Relations**

  All of the translation definitions generated by the Twenty Questions tool are restricted to semantic mappings using only the definitional extensions of the PSL Ontology; they do not provide general semantic mappings between concepts within the core theories of the ontology.

  Translation definitions are also restricted to mappings between the classes of the application ontology and the PSL Ontology; they do not map relations in the different ontologies. For example, different applications may impose restrictions on the *subactivity* relation in the composition of complex activities—in one ontology, the relation may not be transitive, while in the other ontology, the relation may be isomorphic to a bipartite graph consisting of primitive and nonprimitive activities. Even though both of these relations are definably interpretable within the PSL Ontology, the mappings do not use invariants, and there is no general way of generating a direct mapping between the two ontologies.

  This leads to the following question:

  Under what conditions does the existence of a semantic integration profile guarantee the existence of a definable interpretation of primitive relations with respect to the invariants in the profile?

- **Incomplete Sets of Invariants**

  The approach to semantic integration taken in this paper relies on the existence of a complete set of invariants for the models of the ontology. However, there are theories (e.g. graphs) for which such a set of invariants cannot be found. In such cases, two concepts may have equivalent semantic integration profiles (i.e., equivalent values for the invariants) yet not have isomorphic intended models.

  In some cases, this may require the introduction of new core theories to axiomatize the intended models of the concepts. For example, a theory of resource requirements would be required to distinguish between different classes of manufacturing and logistics activities. Since this does not eliminate the problem if the models of the new core theories also do not have complete sets of invariants, we are faced with the following question:

  Given a theory whose models cannot be completely classified by some set of invariants, how can the translation definitions be augmented by more general relative interpretation axioms?

- **Recognizing Classes from Domain Theories**

  The PSL Ontology makes a distinction between the axioms of the ontology and the axioms of a domain theory that uses the ontology, which are characterized as syntactic classes of sentences that are satisfied elements of the models. For example, traditional precondition axioms are characterized as the class of sentences that are satisfied by *markov_precond* activities, and traditional effect axioms are equivalent to the class of sentences that are satisfied by *markov_effect* activities. On the other hand, many process ontologies used by software applications do not explicitly specify classes of activities, but only specify syntactic classes of process descriptions. A comprehensive account of semantic integration must therefore address the following question:

Is it always possible to automatically determine the profile for a class using only the domain theory associated with elements of the class?

7. Conclusions

This paper has described how model-theoretic invariants of an ontology can be used to specify semantic mappings translation definitions between application ontologies and an interlingua. In particular, examples have been presented using the Process Specification Language (PSL) ontology as the neutral medium in integration.

The sets of models for the core theories of PSL are partitioned into equivalence classes defined with respect to the invariants of the models. Each equivalence class in the classification of PSL models is axiomatized using a definitional extension of PSL. The Twenty Questions tool that is based on these invariants and definitional extensions supports semi-automatic generation of semantic mappings between an application ontology and the PSL Ontology.

This approach can be generalized to other ontologies by specifying the invariants for the models of the axiomatizations. Future work in this area includes developing software to generate mappings based on profiles created with the Twenty Questions tool and application to translation between PSL and other ontologies (such as OWL-S) and translators for existing process modelers and schedulers.

Acknowledgements: This work was supported by the Precision Engineering Project within the Manufacturing Engineering Laboratory at the National Institute of Standards and Technology (NIST); Office of Naval Research (ONR) Grant N00014-01-1-0618; and NIST Grant #70NAN33H1026, funded by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and not necessarily the supporting organizations.

References

[1] Bouquet, P., Serafini, L., Zanobini, S., and Benerecetti, M. (2003) An Algorithm for Semantic Coordination. *Semantic Integration Workshop, International Semantic Web Conference 2003*, 21–26.

[2] Ciocoiu, M., Gruninger M., and Nau, D. (2001) Ontologies for integrating engineering applications, *Journal of Computing and Information Science in Engineering*, 1:45-60.

[3] Ciocoiu, M. 2002 *Ontology-based Semantics*, Ph.D. thesis, Department of Computer Science, University of Maryland, College Park.

[4] Duschka, O.M. and Genesereth, M.R. 1997. Infomaster - an information integration tool. In *Proceedings of the International Workshop on Intelligent Information Integration*. Freiburg, Germany.

[5] Gruninger, M. (2003) Applications of PSL to Semantic Web Services, *Workshop on Semantic Web and Databases*. Very Large Databases Conference, Berlin.

[6] Gruninger, M. (2003) A Guide to the Ontology of the Process Specification Language, in *Handbook on Ontologies in Information Systems*, R. Studer and S. Staab (eds.). Springer-Verlag.

[7] Gruninger, M. and Menzel, C. (2003) Process Specification Language: Principles and Applications, *AI Magazine*, 24:63-74.

[8] Gruninger, M. and Uschold, M. (2003) Ontologies and Semantic Integration, in *Software Agents for the Warfighter*, Information Technology Assessment Consortium, to appear.

[9] Mahdavan, J., Bernstein, P., and Rahm, E. (2001) Generic Schema Matching with Cupid, *Proc. 27th VLDB Conference*, 49–58.

[10] Marker, D. (2000) *Model Theory: An Introduction*. Springer-Verlag.

[11] McIlraith, S., Son, T.C. and Zeng, H. (2001) Semantic Web Services, *IEEE Intelligent Systems*, Special Issue on the Semantic Web. 16:46–53, March/April, 2001.

[12] Noy, N. and Musen, M. (2000) PROMPT: Algorithm and tool for automated ontology merging and alignment, Proceedings of AAAI-2000.

[13] Stuckenschmidt, H. and Visser, U. (2000) Semantic Translation Based on Approximate Reclassification. In Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning, Breckenridge, Colorado.

[14] Uschold, M., Jasper, R. and Clark, P. 1999. Three Approaches for Knowledge Sharing: A Comparative Analysis. In *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling, and Management (KAW'99)*.

INSTITUTE FOR SYSTEMS RESEARCH, UNIVERSITY OF MARYLAND, COLLEGE PARK, MD 20742, gruning@cme.nist.gov

GEOMETRIC AND INTELLIGENT COMPUTING LABORATORY, DEPT. OF COMPUTER SCIENCE, DREXEL UNIVERSITY, 3141 CHESTNUT STREET, PHILADELPHIA, PA 19104, tjkopena@cs.drexel.edu