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. This article presents an approach toward this goal using techniques that exploit the model-theoretic structures underlying ontologies. With these as inputs, semiautomated and automated components may be used to create mappings between ontologies and perform translations.

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 (Noy and Musen 2000; Stuckenschmidt and Visser 2000) or heuristics-based comparisons of the symbols of the terminology (Bouquet et al. 2003; Gruninger and Uschold 2003). 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, semiautomated and automated components may be used to create mappings between ontologies and perform translations.

The Process Specification Language (PSL) (Gruninger 2003b; Gruninger and Menzel 2003) 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 semiautomatically 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 that do not have an associated, explicit, formal ontology.

**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 that must be performed and answered.

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 (Marker 2000): If \( \mathcal{N} \) is a structure in the language \( \mathcal{L}_0 \) and \( \mathcal{M} \) is a structure in the language \( \mathcal{L} \), then we say that \( \mathcal{N} \) is definably interpretable in \( \mathcal{M} \) if we can interpret the symbols of \( \mathcal{L}_0 \) so that there exists a substructure \( \mathcal{M} \) that is isomorphic to \( \mathcal{N} \). Semantic mappings are the sentences that axiomatize this interpretation. The techniques that we discuss in this article semiautomatically 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 Gruninger and Uschold (2003), 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 with 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 (Ciocoiu, Gruninger, and Nau 2001). Semantic mappings between application and interlingua ontologies are manually generated and verified prior to application interactions (Ciocoiu 2002). 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 that 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. 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 that 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 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.

**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 (Ciocoiu 2002; Ciocoiu, Gruninger, and Nau 2001). 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 be-
between the applications also preserves semantics.

An important question is then whether the existence of the predefined 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 $M_1$ and $M_2$ are both definably interpretable in $N_K$, is $M_1$ definably interpretable in $M_2$? 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.

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

Notice that each question in figure 2a corresponds to an invariant for an object, and each value of the invariant is a possible answer to the question, as in figure 2c. We will later use this same correspondence between invariants 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 degrees, 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.

To illustrate how invariants are used to pro-
A.
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 \( D_n \)?

B.
Regular polygons \( \equiv \) convex polygons with
\( n \geq 3 \) sides and symmetry group \( \equiv D_n \).

C.

\[
\begin{array}{cccc}
\text{yes, } n = 6 & \text{yes, } n = 8 \\
\text{convex; } & \text{convex; } \\
\text{Symm } \not\equiv D_n & \text{Symm } \equiv D_n \\
\Rightarrow \text{regular} & \Rightarrow \text{regular} \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{yes, } n = 8 & \text{yes, } n = 8 \\
\text{concave; } & \text{convex; } \\
\text{Symm } \not\equiv D_n & \text{Symm } \equiv D_n \\
\Rightarrow \text{regular} & \Rightarrow \text{regular} \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{yes, } n = 3 & \\
\text{convex; } & \\
\text{Symm } \equiv D_n \\
\Rightarrow \text{regular} \\
\end{array}
\]

Figure 2. An Example of the Use of Invariants in
Constructing a Terminology of Geometric Shapes.

A. Several invariant properties of geometric shapes. B. Definition for the class of
regular shapes using the above invariants. C. Several shapes classified as regular
or irregular through comparison to the definition. Although not a complete set,
these invariants do support the formal definition of terms in the language.

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—di-
monds (♦), hearts (♥), spades (♠), and clubs (♣). Although there are ten possible poker
hands, we will focus on three of these hands. A flush is a hand in which all of the cards are the
same suit, for example, all cards have the heart 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, for example, 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, for
example, 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
type 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 activi-
ties in an occurrence tree. Rather than change
cards with the same suit, we consider permuta-
tions 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 oc-
currences of the activity. If any such permuta-
tion will preserve legal occurrence, then the ac-
tivity is the markov_precond class, as axiomat-
tized in figure 4. With a partial_state activity, if

be sufficient funds in the account). The class of
Markovian activities is defined in the PSL
definitional extension state_precond.def, a por-
tion 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 (for example, transac-
tions must be completed during office hours),
and any process ontology should be able to
capture these constraints as well.
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 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.

Semantic Mapping
Via Translation Definitions

As noted previously, 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...
$T_2$ if and only if 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 $T_2$ if and only if 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 (McIlraith, Son, and Zeng 2001) has the following translation definition with respect to the PSL ontology:

\[
\forall \, a \, (
\text{idf}(\text{AtomicProcess} \, a) \land
\text{idf}(\text{atomic} \, a) \land
\text{idf}(\text{markov\_precond} \, a) \land
\text{idf}(\text{markov\_effects} \, a)))
\]

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 (that is, fluents that are achieved or falsified by activity occurrences).

Semiautomatic 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. 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 semiautomated 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:

\[
\forall \, a \, (\text{idf}(\text{myclass} \, a) \land
\text{idf}(\text{markov\_precond} \, a))
\]

Selecting the first two answers would give the translation definition:

\[
\forall \, a \, (\text{idf}(\text{myclass} \, a) \lor
\text{idf}(\text{partial\_state} \, a))
\]

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

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 of$T_1$ that are mapped to the invariants of$T_2$. This consequently implies that all activities, occurrences, and fluents in$T_1$ that have the same value for the invariant will be mapped to activities, occurrences, and fluents in$T_2$. The invariant corresponding to the$\text{markov\_precond}$ class was discussed in the preceding section; the invariants corresponding to the$\text{markov\_effects}$ and$\text{context\_free}$ classes are based on groups consisting of permutations of activity occurrences that preserve effects (that is, fluents that are achieved or falsified by activity occurrences).
variants that capture the intended semantics of
the class of activities that appear in the an-
tecedent of the translation definition. Howev-
er, this raises the issue of validating the seman-
tic 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 applica-
tion ontologies are axiomatized, then we can
verify the semantic mappings by proving that
they do indeed preserve the models of the on-
tologies. This can be done by demonstrating
that the class of models of the application on-
tology is axiomatized by the interlingua, to-
gether with the translation definitions.

In practice, the validation of semantic map-
pings is complicated by the fact that few soft-
ware applications have explicitly axiomatized
ontologies. In such cases, the twenty questions
tool can also be used to define a formal ontol-
ogy for the software applications. This is afford-
ed by the assumption of the ontological stance
(Gruninger and Menzel 2003), the main tenet
of which is that a software application may be
modeled as if it were an inference system work-
ning 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 mod-
els. Given a software application, there exists a
class of models $\mathcal{M}^A$ such that any sentence $\Phi$ is
decided by the application to be satisfiable if
and only if there exists a model $\mathcal{M}$ in $\mathcal{M}^A$ such
that $\mathcal{M}$ entails $\Phi$.

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 ax-
iomatization of the application’s previously im-
plicit ontology.

To validate the attributed ontology, the gen-
eration of translation definitions may be treated as
falsifiable hypotheses and tested empirically.
By the ontological stance, the application de-
cides some sentence $\Phi$ to be provable if and on-
ly if $T_{psl} \cup T_{translation}$ entails $\Phi$ where $T_{psl}$ is
the set of axioms for the PSL ontology and $T_{transla-
tion}$ is the set of translation definitions that are
being verified. In this way, it may be evaluated
whether or not the attributed ontology cor-
rectly predicts inferences made by the software
and, consequently, whether or not the transla-
tion definitions accurately capture the seman-
tics of the application.

---

**Definition 1**

\[
\begin{align*}
&\text{(forall (?o1 ?o2) (iff (state_equiv ?o1 ?o2))} \\
&\text{(forall (?f))} \\
&\text{(iff (prior ?f ?o1) \ (prior ?f ?o2))})
\end{align*}
\]

**Definition 2**

\[
\begin{align*}
&\text{(forall (?a ?o1 ?o2) (iff (poss_equiv ?a ?o1 ?o2))} \\
&\text{(implies \ (and (occurrence_of ?o1 ?a) \ (occurrence_of ?o2 ?a)) \ (legal_equiv ?o1 ?o2)))}
\end{align*}
\]

**Definition 3**

\[
\begin{align*}
&\text{(forall (?a) (iff (markov_precond ?a))} \\
&\text{(forall (?o1 ?o2))} \\
&\text{(implies \ (state_equiv ?o1 ?o2) \ (poss_equiv ?a ?o1 ?o2)))}
\end{align*}
\]

**Definition 4**

\[
\begin{align*}
&\text{(forall (?a) (iff (partial_state ?a))} \\
&\text{(forall (?o1))} \\
&\text{(exists (?o2) \ (implies \ (state_equiv ?o1 ?o2) \ (poss_equiv ?a ?o1 ?o2))))} \\
&\text{(exists (?o3 ?o4))} \\
&\text{(and (state_equiv ?o3 ?o4) \ (not (poss_equiv ?a ?o3 ?o4)))))}
\end{align*}
\]

**Definition 5**

\[
\begin{align*}
&\text{(forall (?a) (iff (rigid_state ?a))} \\
&\text{(forall (?o1))} \\
&\text{(exists (?o2))} \\
&\text{(and (state_equiv ?o1 ?o2) \ (not (poss_equiv ?a ?o1 ?o2)))))}
\end{align*}
\]

---

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 $state_equiv$ if
there exists a permutation of activity occurrences that preserves the fluents that
hold prior to the activity occurrences. The two activity occurrences are $poss_equiv$
if and only if there exists a permutation of activity occurrences that preserves le-
gal occurrences of the activity.

**Comparison of Semantic-Integration Profiles for Integration**

The set of translation definitions for all con-
cepts 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 applica-
tion profile will have the form:

\[
\text{(forall (?a)} \\
\text{(iff (C^i_onto ?a) \ (and (or (p_{i1} ?a ... (p_{in} ?a)))})
\]
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.

Each clause in the profile corresponds to a different invariant; each literal \((p_g ?a)\) 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 \(C_{alice} (a)\), which has unconstrained preconditions (that is, they are always possible) and whose effects are either context free or depend only on the state prior to occurrences of the activities. Suppose that Bob’s ontology contains a class of activities \(C_{bob} (a)\) 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:

\[
\text{forall } (?a) \\
\text{ (iff } (C_{alice} ?a) \\
\text{ (and (unconstrained ?a) \\
\text{ (or (markov_effects ?a) \\
\text{ (context_free ?a)))})})
\]

\[
\text{forall } (?a) \\
\text{ (iff } (C_{bob} ?a) \\
\text{ (and (context_free ?a) \\
\text{ (or (markov_precond ?a) \\
\text{ (unconstrained ?a)))})})
\]

As noted earlier, 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:

\[
\text{forall } (?a) \\
\text{ (implies (context_free ?a) \\
\text{ (implies (C_{alice} ?a) \\
\text{ (C_{bob} ?a)))))}
\]

\[
\text{forall } (?a) \\
\text{ (implies (unconstrained ?a) \\
\text{ (implies (C_{bob} ?a) \\
\text{ (C_{alice} ?a)))))}
\]

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:

\[
\text{forall } (?a) \\
\text{ (implies (and (or (p_{11} ?a) ... (p_{1n} ?a)) \\
\text{ ... (or (p_{m1} ?a) ... (p_{mn} ?a)))}) \\
\text{ (implies (C_{alice} ?a) \\
\text{ (C_{bob} ?a)))))
\]

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, 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 trade-offs and overlaps between point-to-point and interlingua-based integration approaches is presented in Uschold, Jasper, and Clark (1999).

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 evaluate relationships and coverage relative to the interlingua or other ontologies.

Open Problems

Several important issues related to semantic integration have not been addressed so far in this work: translation definitions for primitive relations; incomplete sets of invariants; and recognizing classes from domain theories.

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 article relies on the existence of a complete set of invariants for the models of the ontology. However, there are theories (such as graphs) for which such a set of invariants cannot be found. In such cases, two concepts may have equivalent semantic-integration profiles (that is, 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?

Figure 6. The PROFILE-COMPARE Algorithm for Determining Relationships between Ontologies, Given the Semantic-Integration Profiles.
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 by 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?

Conclusions

This article 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 semiautomatic 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 [Gruninger 2003a]) 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 N00014-0110618; and NIST Grant #70NAN33 H1026, funded by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and not necessarily the supporting organizations.

Notes

1. See Uschold, Jasper, and Clark (1999) for a more detailed discussion of the trade-offs between the point-to-point and interlingua approaches.
2. OWL-S is an Ontology Web Language (OWL) 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.

References

Ciocoiu, M. 2002. Ontology-based Semantics, Ph.D. diss., Department of Computer Science, University of Maryland, College Park, MD.

Michael Gruninger is currently an assistant research scientist in the Institute for Systems Research at the University of Maryland College Park and also a guest researcher at the National Institute for Standards and Technology (NIST). His current research focuses on the design and formal characterization of ontologies and their application to problems in manufacturing and enterprise engineering.

Joseph Kopena is a student researcher at Drexel University, where he is a member of the Secure Wireless Agent Testbed (SWAT) project headed by William Regli and Moshe Kam. His current research revolves around the application of service-based computing to disruption-prone networking environments. His previous research includes knowledge representation for engineering design repositories and low-cost mobile robotics for education.