Ontology Formalisms: What is Appropriate for Different Applications?

Craig Schlenoff
National Institute of Standards and Technology
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899
301-975-3456

craig@schlenoff.com

ABSTRACT

Ontologies can take many forms. There are ontologies that are extremely formal (e.g., using first order logic), and there are ontologies that are less formally defined (e.g., ontologies in the relational databases or dictionaries). Nonetheless, all of these can be considered ontologies and are appropriate in different situations.

In this paper, I will present a view of levels of ontology formalizations and then describe three efforts that have applied ontologies to solve real-world problems. I will show where each of these efforts fall on the formalization spectrum and show why that level of formalization is appropriate for that application.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods Language – *predicate logic, relation systems, representation languages, representations*

General Terms

Design, Experimentation, Standardization, Languages, Theory

Keywords

Ontologies, Formalization, Robotic, Knowledge Representation

1. INTRODUCTION

Ontologies can take many forms. There are ontologies that are extremely formal (e.g., using first order logic), and there are ontologies that are less formally defined (e.g., ontologies in the relational databases or dictionaries). Nonetheless, all of these can be considered ontologies and are appropriate in different situations.

Similarly, ontologies can play different roles. They can be used for common access to information, for search, as exchange languages and for reasoning.

In this paper, I will present one view of different levels of ontology formalizations and then describe some efforts that have applied ontologies to solve real-world problems. I will then show where each of these efforts fall on the formalization spectrum and show why that level of formalization is appropriate for that application. Section 2 describes the formalization scale that will be using for this paper and Section 3 gives an overview of how ontologies have been used in real-world applications. Section 4 describe the details of three projects that have used ontologies and how they fit into the classifications described in Sections 2 and 3. Section 5 concludes the paper.

PerMIS'09, September 21-23, 2009, Gaithersburg, MD, USA.
ACM 978-1-60558-747-9/09/09

2. LEVELS OF ONTOLOGY FORMALISM

For the purpose of this paper, I will loosely define an ontology as a knowledge representation that can be captured at different levels of formality, ranging from terms in a glossary or dictionary up to formal logic-based descriptions. Admittedly, this definition of an ontology is much broader than the commonly-accepted view. Often, people think of an ontology as a more formal representation; often one that can be reasoned over in an automated fashion.

To describe the level of formalisms of an ontology, I will use the scale in Figure 1. This scale shows examples of ontologies listed from least formal (left side of the figure), to more formal (right side of the figure). The black diagonal line in the middle of figure shows the point at which ontologies can be reasoned over. One can run a reasoning engine over everything to the right of the line but cannot over the formalisms to the left of the line. This figure was not created by the author; it is often used in the literature but the author was unable to find the origin of it.

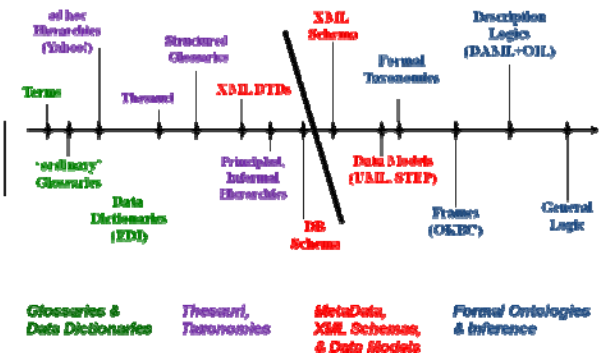

Figure 1: Levels of Ontology Formalism

The items in green can be thought of as standard glossaries or dictionaries, similar to the ones that you may have in your house. The items in purple are thesauri, taxonomies, and hierarchies. In this realm, one is starting to organize and categorize information.

These structures start to exhibit some superclass/subclass types of relationships, and can be used for application such as navigating web pages. The red items start providing a lot more structure to data, and provide a much richer set of relationships, such as part-of, contains, spatial relations, etc. These structures are often used as specification for software, exchange languages, and ontology-based search. The items in blue can be thought of as formal ontologies and are often represented in logic-based languages such as the Knowledge Interchange Format (KIF) or description logics. The advantage of these types of formalisms is that they allow for inferencing. This allows one to discover additional information that is not formally represented and also allows one to identify inconsistencies in the knowledge that is represented. For the remainder of this paper, I will be using this formalization scale to characterize existing efforts in ontology development for robotics and related applications.

3. APPLYING ONTOLOGY: THE BIG PICTURE

Over the past two decades, ontologies have found a role in many different applications. Four ways in which ontologies have been used include:

- Common access to information
- Ontology-based search
- Exchange language
- Reasoning

Each of these is discussed in more detail below.

3.1 Common Access to Information

It is often the case that multiple applications need access to the same information. This type of information could be a material database, a specification of a part to be manufactured, or terrain characteristics of an environment that an autonomous vehicle must traverse. Instead of requiring an application to encode this information in its own internal format, an ontology can provide this information in a neutral format that these different applications can reference. By not duplicating this information is numerous different applications, the ontology can allow the information to be represented only once, providing only a single source when information needs to be updated and ensuring that there is consistency between the information in different applications. In addition, the ontology provides a common set of vocabulary that all of the applications that access the ontology can reference. This will ensure that when information exchange between the applications needs to occur, mappings between concepts can be easily done based on the vocabulary in the ontology. The Road Network Database and the Intelligent Systems Ontology, discussed later in the paper, are examples of ontologies developed for common access to information.

3.2 Ontology-Based Search

It is not uncommon that a given concept can have two terms that correspond to it. For example, when a person goes to a web site and wants to buy a helmet that will protect their head at a construction site, they may type in either “hard hat”, “protective helmet”, “hard helmet”, or possibly other terms. Depending what the person types is, often different results will be displayed. This is because search engines often work on the term that is entered as opposed to the concept that is intended.

Ontologies can help to address this issue by representing concepts by what they mean (as opposed to the terms that are used to represent them) and then mapping search terms and underlying information in product databases to those ontological concepts. This is not only true with products… it can also be done with web pages or any other item that needs to be searched.

There are no specific examples of this type of ontology application in the paper but please contact the author if you would like to learn more about how this was applied in private industry.

3.3 Ontologies as an Exchange Language

Information often needs to be shared among different applications. This information is usually generated in one application and then needs to be sent to another application. An example of this is in the manufacturing domain where a person may use a process planning system to create a part in one application and then needs to send that information to a scheduling or production planning system to allow the part to be made. The problems with point-to-point translators between each pair of application are well documented, and these result in a very large amount of translators that need to be developed. Also, as a new version of the application is released, all of the translators that are written either to or from that application need to be updated.

Ontologies have shown to be valuable in serving as a neutral representation to allow for the exchange of information between different applications. The ontology provides a common superset of all of the information structures that need to be exchanged between the applications. By having this common interchange structure, a given application would only have to write a translator to and from the ontology and then would be able to exchange information with any other application that has done the same.

STEP [1] (STandard for the Exchange of Product model data) is perhaps the most widely used ontology for this purpose. In this paper, I describe the Process Specification Language (PSL) which is a more formal ontology that is used to exchange process data among applications.

3.4 Ontologies for Reasoning

When represented formally, ontologies have the ability to reason over information and provide additional information that was not previously formally represented. For example, when an autonomous vehicle is driving down a road and is presented with multiple paths, each of which that has an obstacle in its way, the ontology can reason about the expected damage that could occur by hitting each of the obstacles based on their known characteristics and those of the vehicle. Then a proposed path can be presented to a planner to determine how the vehicle should proceed. An ontology for navigation planning is discussed later in this paper which shows how ontologies can be used for this purpose.

4. ONTOLOGY EXAMPLES

In this section, I will describe existing and past efforts that have used ontologies for real-world applications. For each effort, I will characterize it with respect to its level of formality as described in Section 2 and what role it is playing as described in Section 3. I will start with ontologies that are considered to be less formal and then proceed to more formal ontologies.

4.1 The Road Network Database

For an autonomous vehicle to navigate a road network, it must be aware of and must respond appropriately to any object it encounters. This includes other vehicles, pedestrians, debris, construction, accidents, emergency vehicles … and the roadway itself. The road network must be described such that an autonomous vehicle knows, with great precision and accuracy, where the road lies, rules dictating the traversal of intersections, lane markings, road barriers, road surface characteristics, and other relevant information.

The purpose of this section is to provide an overview of the Road Network Database [2], which is to provide the data structures necessary to capture all of the information necessary about road networks so that a planner or control system on an autonomous vehicle can plan routes along the roadway at any level of abstraction. At one extreme, the database should provide structures to represent information so that a low-level planner can develop detailed trajectories to navigate a vehicle over the span of a few meters. At the other extreme, the database should provide structures to represent information so that a high-level planner can plan a course across a country. Each level of planning requires data at different levels of abstraction, and as such, the Road Network Database must accommodate these requirements. In this section, I explore the contents of the Road Network Database and describe why it was represented in a database format as opposed to a more formal ontology.

The fundamental components of the Road Network Database are described below. This is not an exhaustive list, but instead is meant to give the reader an idea of the type of structures that are represented in the database.

- **Junctions** – A junction is a generic term referring to two or more paths of transportation that come together or diverge, or a controlled point in a roadway. Examples of roadway paths that could cause a junction are lanes splits, forks in the road, merges, and intersections.
- **Intersections** - Intersections are a type of junction in which two or more separate roads come together.
- **Lane Junctions** - A lane junction is a location in a junction in which two or more lanes of traffic overlap.
- **Road** – A road is a stretch of travel lanes in which the name of the travel lanes does not change. An example is “Main Street” or “Route 95.”
- **Road Segment** - A road segment is a uni-directional stretch of roadway bounded by intersections. A road segment is roughly analogous to a “block”.
- **Road Element** - A road element is a uni-directional stretch of roadway bounded by any type of junction. Unlike road segments, road elements can be bounded by merging lanes, forks in the road,
- **Lane Cluster** - A lane cluster is a set of uni-directional lanes (with respect to flow of traffic) in which no physical attribute of those lanes change over the span of the lane segment. Unlike a road element, lane clusters are not required to be bounded by junctions.
- **Lane** - A lane is a single pathway of travel that is bounded by explicit or implicit lane marking.
- **Lane Segment** - A lane segment is the most elemental portion of a road network captured by the database structure. Lane segments can be either straight line or constant curvature arcs. One or more lane segments compose a lane
- **Junction Lane Segments** - A junction lane segment is a constant curvature path through a portion of a lane junction.

As stated earlier, the data structures are designed to accommodate a control system that may contain planners with various levels of abstraction. The planners, their descriptions, and the data structures which best correspond to their level of responsibility are shown in Table 1.

Table 1: Planner to Data Structure Mapping

Planner Name	Planner Description	Appropriate Data Structures
Destination Planner	Plans the sequence of route segments to get to commanded destination goal. Outputs MapQuest[1]-like directions Plans on the order of 1 to 2 hrs into the future	Roads Road Segments Intersections
Drive Behavior Planner	Develops low-level behaviors for negotiating intersections and deciding when to change lanes. Plans on the order of 100 secs into the future. Plans up to 500 m	Lane Clusters Lanes Intersection
Elemental Maneuver Planner	Carries out real-time maneuvers to slow down, stop, speed up, and change lateral position. Plans on the order of 10 secs into the future Plans up to 50 m distances	Lanes Lane Segments

[1] The name of commercial products or vendors does not imply NIST endorsement or that this product is necessarily the best for the purpose.

This information is represented in a relational database. An example of the detailed information that was represented for a road segment can be seen in Table 2. The corresponding picture of what a road segment may look like is shown in Figure 2.

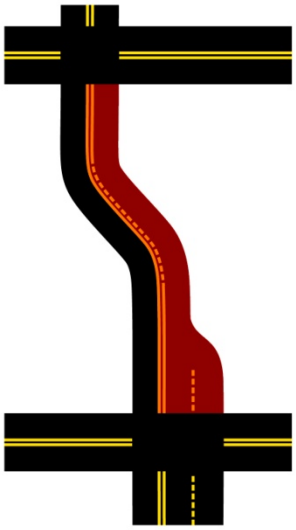

Figure 2: Sample Road Segment

A road segment is a uni-directional stretch of roadway bounded by intersections. A road segment is composed of one or more road elements and zero or more junctions. There are one or more road segments in a road. Unlike road elements, road segments are only bounded by intersection, not any type of junction. A road segment within a road must always be rendered in the same direction as the road. Road segments are used in the planning and control system to provide MapQuest-like directions to the vehicle to allow for route planning.

This Road Network Database is represented as a database schema (on the left side of the formalization figure shown in Section 2). The reason why a more informal representation was chosen was because the database was meant to serve for common access to information (as described in Section 3). It was not anticipated that any reasoning would need to be performed on the data structures so a more formal type of representation (e.g., logic) was not needed. Conversely, since the database was expected to provide common access to information, more informal types of representations (glossaries, data dictionaries, informal hierarchies) were not used since they did not provide the level of specificity needed and provide too high a level of ambiguity in the meaning of the terms that were represented.

Table 2: Road Segment Database Representation

Attribute	Data Type	Value Restriction	Point To	Description
ID	Integer	Any whole number greater or equal to one		A unique identifier for this entry in this table
World_ID	Integer		World.ID	A pointer to an element in the World table that indicates with which world this entry is associated. A road segment may only be associated with a single world. See 4.5.1. for information about worlds.
Description	Text			A textual description of this field for human understanding
Road_ID	Integer		Road.ID	A pointer to the element in the Road table in which the road segment is a part of.
Start_Point_Adjacent_Intersection_ID	Integer		Intersection.ID	A pointer to the element in the Intersection table which precedes the road segment.
End_Point_Adjacent_Intersection_ID	Integer		Intersection.ID	A pointer to the element in the Intersection table which follows the road segment.
Segment_Length	Double			Measured in meters. The length of the road segment measured from center point to center point. This should be derived from the length of the road elements which compose it.
Road_Segment_Class	Integer		RoadSegmentClass.ID	A pointer to an element in the RoadSegmentClassLookup table which contains the class of road segment which applies to this road segment.

4.2 Ontologies for Autonomous Navigation

The field of autonomous vehicles has reached a level of maturity such that it could greatly benefit from leveraging the latest technologies in the area of reasoning over knowledge representations and ontologies.[2] The use of ontologies and automated inference is a natural fit for representing and reasoning about world models (the internal knowledge representation) for autonomous vehicles. The goal for the effort described in this section is to apply ontologies to improve the capabilities and performance of on-board route

[2] The 2004 AAAI Spring Symposium series includes a workshop on this the topic: "Knowledge Representation and Ontology in Autonomous Systems". See: http://www.aaai.org/Symposia/Spring/2004/sssparticipation-04.pdf

planning for autonomous vehicles. More specifically, to apply ontologies to determine the extent to which a given object is an obstacle to a given vehicle in a given situation [3].

There are many potential benefits of introducing an ontology (or set of ontologies) into an autonomous vehicle's knowledge base. One is the potential for reuse and modularity. For example, a general theory of obstacles could apply to a broad range of autonomous vehicles. In addition, ontologies provide a mechanism to allow for a more centralized approach to represent and reason about environmental information. Different modules in an autonomous vehicle would query the ontology, rather than having the information scattered among the modules. This has a corresponding benefit in cheaper and more reliable maintenance. Finally, there is the potential for increased flexibility of response for the autonomous vehicle. Methods that rely on pre-classification of certain kinds of terrain in terms of their traversability [4;5] are important, but do not support reasoning about objects in a more dynamic context.

I start with the simple scenario illustrated in Figure 3. Our vehicle (labeled OV) is in the left lane of a four-lane, two-way, undivided highway. An object is detected in our lane. The goal is to formulate an optimal route plan that takes into account the potential damage from a collision with the object. The main role of the ontology component is [initially] to provide assessments of collision damage. I will take into account not only damage to the vehicle, but also damage to the payload and to the object, itself. This information is used to plan a route that either goes around the object, or collides with it.

A number of parameters may be varied in this scenario. These include: the type of vehicle being controlled, the speed at which the vehicle is traveling, the payload being carried, and type of object in the path that may be an obstacle. For example, if the object is a newspaper in the middle of the roadway, then the ontology component will conclude that no damage will occur and the planner will conclude that the best course of action is to maintain the current lane (because changing lanes always accumulates additional risk over maintaining your lane).

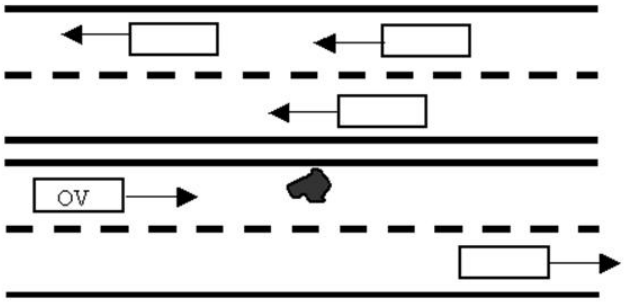

Figure 3. Simple Driving Scenario

However, if the object were a large cinder block, significant damage would be likely and the final route should be quite different. The ontology component is equipped with knowledge about many kinds of vehicles, objects, and the kind of damage that can arise from different collisions. This is used to determine the damage that would be caused by a collision.
The ontology includes objects, vehicles and situations with associated inference rules. Specifically, the ontology contains different types of objects that one expect to encounter in various environments, along with their pertinent characteristics and relationships to other objects. Initially the effort is focusing on on-road driving, so categories of objects such as other vehicles, pedestrians, animals, debris, speed bumps, etc. are represented. Each one of the objects that fall under these categories has a set of characteristics that describe them and help us to understand the damage that may be caused by colliding with them. For example, a certain type of debris may have a set of dimensions, a weight, a density, a velocity, etc. The rules determine the 'degree of obstacleness', which is ultimately expressed in terms of a cost.

The ontology and its associated reasoning engine provides as an output, a damage assessment in the event of a collision between our vehicle and a given object based upon:

- The type of autonomous vehicle;
- The type of object being collided with;
- The closing speed of our vehicle with the object;
- The integrity of our vehicle, i.e., what damage has already occurred to our vehicle, if any.

Based on this information, the ontology provides a damage classification pertaining to:

- The vehicle's integrity (initially only assigning damage to the bumper, wheels, and overall vehicle, but will eventually include other components of the vehicle).
- The obstacle's integrity
- The vehicle payload's integrity

In order to provide the damage classifications, the expressiveness of the ontology must be such that it represents concepts such as:

- The type of vehicle that is being autonomously controlled and its pertinent characteristics;
- The objects that are being encountered in the environment and their pertinent characteristics;
- The payloads that the vehicle is carrying and their pertinent characteristics;
- Severity classifications of damage;
- Damage types;
- Terrain information (initially fixed as paved roads);
- Collisions (e.g., a certain type of vehicle with a certain type of object).

For the initial work, the levels of collision damage shown in Table 3 are assumed.

Table 3: Levels of Collision Damage

	Vehicle	Object	Payload
None	No damage to vehicle	No damage to object	No damage to payload
Minor	Damage to vehicle will not affect vehicle performance	Damage to object will not affect object overall integrity	Damage to vehicle will not affect payload
Moderate	Moderate probability of vehicle damage, maintenance required	Damage to object will affect object integrity, but will not result in object destruction	Moderate probability of payload loss
Severe	Major loss of functionality/ integrity of vehicle likely	Major destruction of object	Major payload loss
Catastrophic	Vehicle loss	Object destruction	Payload loss

There are many approaches that could be used to estimate the actual collision damage. These include:

- Numerical simulation tools which model the physics of weight, materials, shapes, density, momentum etc. to compute impact damage;
- Probabilistic models;
- Fuzzy logic;
- Symbolic logic.

Not one of these techniques is likely to be adequate in all circumstances. The current work focuses on the symbolic logic approach. The hypothesis is that even when logic-based inference is not sufficient, the core ontology of objects and characteristics will remain useful as a conceptualization and vocabulary for expressing rules and procedures for estimating damage.

For the initial experiments, a small ontology was constructed using OilEd [6]. Using a description logic [7] tool has two advantages for us. First, the classifier detects logical errors in the ontology, which greatly increases confidence that the ontology is correct. Second, it is very fast at doing inference. This is important because the planner needs to query the ontology component up to a few hundred times a second to get damage estimates for the many nodes being explored in the search space.

A class called *Situation* was defined which has various characteristics or attributes, each modeled by functional relations with *Situation* as the domain. The key characteristics of a *Situation* that will determine the damage classification are the vehicle, the payload and the object with which the vehicle may collide. These functional relations are called *hasVehicle*, *hasPayload*, and *hasPotentialObstacle*, respectively. Attributes were also used to define the damage categories in Table 3. For example, the class *VehicleIntegrityMinor* is defined to be the class of all *Situations* such that the value of the functional relation *hasVehicleIntegrity* attribute is *Minor*.

A simple ontology of physical objects was constructed that including various types of vehicles and other objects such as bricks, newspapers etc. that may be in the vehicle's environment. These objects have characteristics such as weight, speed, density, etc. that are important in determining the damage category. Initially, some qualitative categories for measuring these characteristics were created, such as low, medium and high for weight, or density.

Finally some axioms were created which specify how to classify a given situation in terms of the categories in Table 3. Here is a simple example:

A *Situation* such that

- The value of the *hasPotentialObstacle* relation is restricted to be of type *SmallDenseObject.*

  &
- The value of the *hasVehicle* relation is restricted to be of type *Car*

is a subclass of *VehicleIntegrityModerate*.

Some fictitious situations were created to test these axioms. For example, the situation whose *hasPotentialObstacle* relation is a brick, and whose *hasVehicle* relation is a Toyota Corolla will be classified by this rule under *VehicleIntegrityModerate.* This is inferable because a brick is a *SmallDenseObject* (by virtue of its weight and size), and a Toyota Corolla is a subclass of *Car*.

This Autonomous Navigation Ontology is represented in description logic (near the right side of the formalization figure shown in Section 2). The reason why a more formal representation was chosen was because the ontology was developed to allow reasoning, which requires that the underlying representation be more formal. The effort clearly falls into the "Ontology for Reasoning" section described in Section 3. Full first order logic could have been chosen in this effort, but it was felt that it was overkill for the fairly simple examples that were anticipated.

4.3 The Process Specification Language

The Process Specification Language (PSL) [8] is addressing the software interoperability issue by creating a neutral, standard language for process specification to serve as an interlingua to integrate multiple process-related applications throughout the manufacturing life cycle. This interchange language is unique due to the formal semantic definitions (the ontology) that underlie the language. Because of these explicit and unambiguous definitions, information exchange can be achieved without relying on hidden assumptions or subjective mappings.

Existing approaches to process modeling lack an adequate specification of the semantics of the process terminology, which leads to inconsistent interpretations and uses of the information. Analysis is hindered because models tend to be unique to their applications and are rarely reused. Obstacles to interoperability arise from the fact that the legacy systems that support the functions in many enterprises were created independently, and do not share the same semantics for the terminology of their process models.

For example, consider Figure 4 in which two existing process planning applications are attempting to exchange data. Intuitively the applications can share concepts; for example, both *material* in Application A and *workpiece* in Application B correspond to a common concept of *work-in-progress*. However, without explicit definitions for the terms, it is difficult to see how concepts in each application correspond to each other. Both Application A and B have the term *resource*, but in each application this term has a different meaning. Simply sharing terminology is insufficient to support interoperability -- the applications must share their semantics.

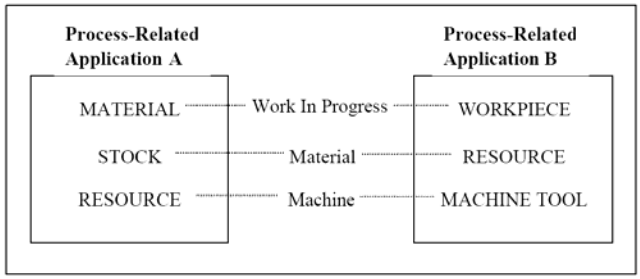

Figure 4: The Need For Semantics

A rigorous foundation for process design, analysis, and execution therefore requires a formal specification of the semantics of process models. One approach to generating this specification is through the use of ontologies. A major goal of PSL is to reduce the number of translators to *O(n)* for *n* different ontologies, since it would only require translators from a native ontology into the interchange ontology.

Within this work, the term “ontology” refers to a set of sentences in first-order logic, comprising a set of foundational theories and sets of definitions written using the foundational theories. In providing such an ontology, one must specify three notions:

- Language
- Model theory
- Proof theory (axioms and definitions)

A language is a set of symbols (lexicon) and a specification of how these symbols can be combined to make well-formed formulae (grammar/syntax). The lexicon consists of logical symbols (such as connectives, variables, and quantifiers) and non-logical symbols. For PSL, the non-logical part of the lexicon consists of expressions (constants, function symbols, and predicates) that refer to everything needed to describe processes.

The underlying language used for PSL is KIF[9] (Knowledge Interchange Format). Briefly stated, KIF is a formal language developed for the exchange of knowledge among disparate computer programs. KIF provides the level of rigor necessary to define concepts in the ontology unambiguously, a necessary characteristic to exchange manufacturing process information using the PSL Ontology.

The primary component of PSL is its terminology for classes of processes and relations for processes and resources, along with definitions of these classes and relations. Such a lexicon of terminology along with some specification of the meaning of terms in the lexicon constitutes what this effort is calling an ontology.

The model theory of PSL provides a rigorous mathematical characterization of the semantics of the terminology of PSL. The objective is to identify each term with an element of some mathematical structure, such as a set or a set with additional structure (e.g. a complete partial order); the underlying theory of the mathematical structure then becomes available as a basis for reasoning about the terms of the language and their relationships.

The proof theory of PSL provides axioms for the interpretation of terms in the ontology. It is useful to distinguish two types of sentences in this set of axioms: core theories and definitions. A core theory is a set of distinguished predicates, function symbols, and individual constants, together with some axiomatization. Distinguished predicates are those for which there are no definitions; the intended interpretations of these predicates are defined using the axioms in the core theories. For these terms, one needs to describe the set of models corresponding to the intuitions that one has for them. Axioms are then written that are sound and complete with respect to the set of models. That is, every interpretation that is consistent with the axioms is a model in the set, and any model in the set is an interpretation consistent with the axioms. These axioms constitute the foundational theories of the ontology. The set of models form the semantics (or model theory) of the ontology.

All other terms in the ontology are given definitions using the set of primitive terms. These definitions are known as conservative definitions since they do not add to the expressive power of the core theories, that is, anything that can be deduced with the definitions, can be deduced using the core theories alone. All definitions in an ontology are specified using the core theories; any terminology that does not have a definition is axiomatized in some core theory. Since all other terms are defined using these primitives, the set of models for them can be defined using the models of the core theories for the primitives. One can therefore assign semantics to the definitions using the classes of models that have already been specified for the core theories.

The challenge is that some framework is needed for making explicit the meaning of the terminology for many ontologies that reside only in people's heads. Any ideas that are implicit are a possible source of ambiguity and confusion. For PSL, the model theory provides a rigorous mathematical characterization of process information and the axioms give precise expression to the basic logical properties of that information in the PSL language. So when one speaks about semantics for PSL, it is in reference to

the axiomatization of core theories and definitions for the PSL terminology.

The focus of the ontology is not only on the terms, but also on their definitions. An infinite set of terms can be included in the ontology, but they can only be shared if everyone agrees on their definitions. It is the definitions that are being shared, *not simply* the terms. A simple definition with the PSL ontology is shown below:

Definition An activity is-occurring-at a timepoint p if and only if p is betweenEq the activity's begin and end points.

```
(defrelation is-occurring-at (?a ?p) :=
    (exists (?occ)
    (and  (occurrence ?occ ?a)
          (betweenEq (beginof ?occ) ?p (endof ?occ)))))
```

A simple axiom within the PSL ontology is shown below:

Axiom. An object can participate in an activity only at those timepoints at which both the object exists and the activity is occurring.

```
(forall (?x ?a ?t)
    (=>  (participates-in ?x ?a ?t)
         (and       (exists-at ?x ?t)
                    (is-occurring-at ?a ?t))))
```

This Process Specification Language is represented in full first order logic (all the way to the right side of the formalization figure shown in Section 2). The reason for this is two-fold:

1. Precise semantics are needed to ensure that complete and unambiguous information exchange occurs between two applications,
2. Reasoning must be performed over the concepts in the ontology to ensure that mappings between the applications ontology and PSL are complete and correct.

The effort clearly falls into the “Ontology as an Exchange Language” section described in Section 3.

5. CONCLUSION

In this paper, different levels of ontology formalization are discussed and an overview of the ways in which ontologies have been used in practice over the past couple decades is described. Three examples are also provided of very different ontologies that have been developed to solve real-world problems in the autonomous vehicle and manufacturing systems integration domains. It is also explained why the formalisms that were used were the most appropriate for their intended purpose.

There are many other ontology efforts which could have been used as examples, including an ontology for searching products on private company’s web site, an ontology for classifying robot capabilities to allow a first responder to find the best robot for a disaster site, and an ontology that was developed to classify autonomous vehicle behaviors so that the right behavior can be chosen when confronted with specific environmental conditions. The three that were chosen were done so because they provide a good spectrum of the types of formalism that can be used when developing an ontology. If the reader is interested in hearing about these efforts, please don’t hesitate to contact the author.

6. REFERENCES

[1] S. Brooks and R. Greenway, "Using STEP to integrate design features with manufacturing features," in *Computers in Engineering Conference* New York, NY: 1995, pp. 579-586.

[2] C. Schlenoff, S. Balakirsky, T. Barbera, C. Scrapper, J. Ajot, E. Hui, and M. Paredes, "The NIST Road Network Database: Version 1.0," National Institute of Standards and Technology (NIST) Internal Report 7136,2004.

[3] C. Schlenoff, S. Balakirsky, M. Uschold, R. Provine, and S. Smith, "Using Ontologies to Aid in Navigation Planning in Autonomous Vehicles," *Knowledge Engineering Review*, vol. 18, no. 3, pp. 243-255, 2004.

[4] J. J. Donlon and K. D. Forbus, "Using a Geographic Information System for Qualitative Spatial REasoning about Trafficability," in *Proceedings of the Qualitative Reasoning Workshop* Loch Awe, Scotland: 2003.

[5] R. M. Malyyankar, "Creating a Navigation Ontology," in *Proceedings of the Workshop on Ontology Management, AAA!-99* Orlando, FL: 1999.

[6] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens, "OilEd: a reason-able ontology for the semantic web," in *Proc. of the Joint German Austrian Conference on AI, number 2174 in Lecture Notes In Artificial Intelligence* Springer-Verlag, 2001, pp. 396-408.

[7] F. Baader, D. McGuinness, D. Nardi, and F. Patel-Schnedier, *Description Logic Handbook: Theory, Implementation and Application* Cambridge University Press, 2002.

[8] C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. Lee, "The Process Specification Language (PSL) Overview and Version 1.0 Specification," in *NISTIR 6459, National Institute of Standards and Technology* 2000.

[9] M. Genesereth and R. Fikes, "Knowledge Interchange Format," in *Stanford Logic Report Logic-92-1* Stanford University: 1992.