THE SEMANTIC WEB:
The Roles of XML and RDF

STEFAN DECKER AND SERGEY MELNIK
Stanford University
FRANK VAN HARMELEN, DIETER FENSEL, AND MICHEL KLEIN
Vrije Universiteit Amsterdam
JEEN BROEKSTRA
AIdministrator Nederland B.V.
MICHAEL ERDMANN
University of Karlsruhe
IAN HORROCKS
University of Manchester

XML and RDF are the current standards for establishing semantic interoperability on the Web, but XML addresses only document structure. RDF better facilitates interoperation because it provides a data model that can be extended to address sophisticated ontology representation techniques.

The World Wide Web is possible because a set of widely established standards guarantees interoperability at various levels. Until now, the Web has been designed for direct human processing, but the next-generation Web, which Tim Berners-Lee and others call the "Semantic Web," aims at machine-processible information.[1] The Semantic Web will enable intelligent services such as information brokers, search agents, and information filters, which offers greater functionality and interoperability than current stand-alone services.

The Semantic Web will only be possible once further levels of interoperability have been established. Standards must be defined not only for the syntactic form of documents, but also for their semantic content. Notable among recent W3C standardization efforts are XML/XML schema and RDF/RDF schema, which facilitate semantic interoperability.

In this article, we explain the role of ontologies in the architecture of the Semantic Web. We then briefly summarize key elements of XML and RDF, showing why using XML as a tool for semantic interoperability will be ineffective in the long run. We argue that a further representation and inference layer is needed on top of the Web's current layers, and to establish such a layer, we propose a general method for encoding ontology representation languages into RDF/RDF schema. We illustrate the extension method by applying it to OIL, an ontology representation and inference language.[2] //AU: Please spell out OIL on first use.//

ONTOLOGIES: DOMAIN CONCEPTUALIZATION

Ontologies can play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications. Generally defined as shared formal conceptualizations of particular domains, ontologies provide a common understanding of topics that can be communicated between people and application systems.

Ontologies are used in e-commerce to enable machine-based communication between buyers and sellers; vertical integration of markets (such as VerticalNet [http://www.verticalnet.com]); and description reuse

```
class-def animal                                    % animals are a class
class-def plant                                     % plants are a class
    subclass-of NOT animal                          % that is disjoint from animals
class-def tree
    subclass-of plant                               % trees are a type of plants
class-def branch
     slot-constraint  is-part-of                    % branches are parts of trees
          has-value tree
class-def leaf
     slot-constraint is-part-of                     % leafs are parts of branches
          has-value branch
class-def defined carnivore                         % carnivores are animals
     subclass-of animal
     slot-constraint eats                           % that eat only other animals
          value-type animal
class-def defined herbivore                         % herbivores are animals
     subclass-of animal
     slot-constraint eats                           % that eat only plants or parts of plants
          value-type plant
          OR (slot-constraint is-part-of has-value plant)
class-def giraffe                                   % giraffes are herbivores
     subclass-of herbivore
     slot-constraint eats                           % and they eat leafs
          value-type leaf
class-def lion
     subclass-of animal                             % lions are also animals
     slot-constraint eats                           % but they eat herbivores
          value-type herbivore
class-def tasty-plant                               % tasty plants are plants that are eaten by
     subclass-of plant                              % both herbivores and carnivores
     slot-constraint eaten-by
          has-value herbivore, carnivore
```

Figure 1. Example ontology defining African wildlife. The hierarchy of concepts is formulated using the OIL syntax for class expressions. / / AU: OK? / /

between different marketplaces. Search engines also use ontologies to find pages with words that are syntactically different but semantically similar.

An ontology typically contains a hierarchy of concepts within a domain and describes each concept's crucial properties through an attribute-value mechanism. Further relations between concepts might be described through additional logical sentences. Finally, constants (such as "January") are assigned to one or more concepts (such as "Month") in order to assign them their proper type.

The OIL ontology consists of slot definitions (slot-def) and class definitions (class-def). Figure 1 is an example ontology (though it omits slot definitions due to space constraints). A slot-def describes a binary relation between two entities. A class-def associates a class name with a class description and consists of the following components (any of which can be omitted):

- *definition type* can either be "defined" or "primitive." For defined types, a class is completely specified in the class definition. For primitive types, the conditions in the class definition are necessary, but insufficient for determining class membership.
- *slot constraint* restricts the possible values a slot can have when applied to an instance of the class.
- *subclass-of* relates the defined class to a list of one or more *class expressions*—class names, slot constraints, or an arbitrarily complex Boolean combination of these. Hence, the class being defined is a subclass of those defined by each class expression.

The main components of a slot constraint are:

- *name*—a string that delineates the slot being constrained.
- *value-type*—a list of one or more class expressions for which the value of the class must be

```
<class-def>
    <class name="plant"/>
    <subclass-of>
        <NOT><class name="animal"/></NOT>
    </subclass-of>
</class-def>
<class-def>
<class name="tree"/>
    <subclass-of>
        <class name="plant"/>
     </subclass-of>
</class-def>
<class-def>
     <class name="branch"/>
     <slot-constraint>
           <slot name="is-part-of"/>
           <has-value>
              <class name="tree"/>
           </has-value>
     </slot-constraint>
</class def>
```

Figure 2. Partial XML serialization of the example ontology in Figure 1. This XML document contains one possible serialization. It introduces a tag for each part of the grammar.

an instance of each class expression in the list if an instance of the defined class has the current slot. //AU: So if there were three expressions in the list, the defined class would have to be all three? (e.g., human, female, brunette)// For example, a class human could have a slot friend with the slot constraint *value-type human*, which means if a human has a friend, the friend must also be a human.

- *has-value*—a list of one or more class-expressions in which every instance of the class defined by the slot constraint must be related via the slot to an instance of each class expression in the list. If the class human has a slot friend with the slot constraint *has-value human*, then every human is the friend of another human. //AU: This example is confusingly similar to the one for *value-type*. Please clarify.//

XML and RDF each have their merits as a foundation for the Semantic Web, but RDF provides more suitable mechanisms for applying ontology-representation languages like OIL to the task of interoperability. //AU: We need a transition to the next sections here. ok?//

XML GRAMMARS

XML is already widely known in the Internet community, and is the basis for a rapidly growing number of software development activities.[3] It is designed for markup in documents of arbitrary structure, as opposed to HTML, which was designed for hypertext documents with fixed structures.

A well-formed XML document creates a balanced tree of nested sets of open and close tags, each of which can include several attribute-value pairs. There is no fixed tag vocabulary or set of allowable combinations, so these can be defined for each application. In XML 1.0 this is done using a *document type definition* to enforce constraints on which tags to use and how they should be nested within a document. A DTD defines a grammar to specify allowable combinations and nestings of tag names, attribute names, and so on. Developments are well underway at W3C to replace DTDs with XML-schema definitions.[4,5] Although XML schema offer several advantages over DTDs, their role is essentially the same: to define a grammar for XML documents.

Figure 2 shows an example serialization of part of the ontology from Figure 1. The basic XML data model is a labeled tree, where each tag corresponds to a labeled node in the model, and each nested subtag is a child in the tree. Of course, this example shows just one possible XML-based syntax for the ontology. XML is foremost a means for defining grammars, and because different grammars can be used to describe the same content, XML allows multiple serializations. The information in the final class definition in Figure 2, for example, could be expressed in an entirely different form as:

```
<class-def>
      <name>branch</name>
      <slot-constraint>
               <name>is-part-of</name>
               <has-value>tree</has-value>
      </slot-constraint>
</class-def>
```

XML is used to serve a range of purposes:

- *Serialization syntax for other markup languages.* For example, the Synchronized Multimedia Integration Language (SMIL)[6] is syntactically just a particular XML DTD; it defines the structure of a SMIL document. The DTD is useful because it facilitates a common understanding of the meaning of the DTD elements and the structure of the DTD.
- *Semantic markup of Web pages.* An XML serialization (such as the example above) can be used

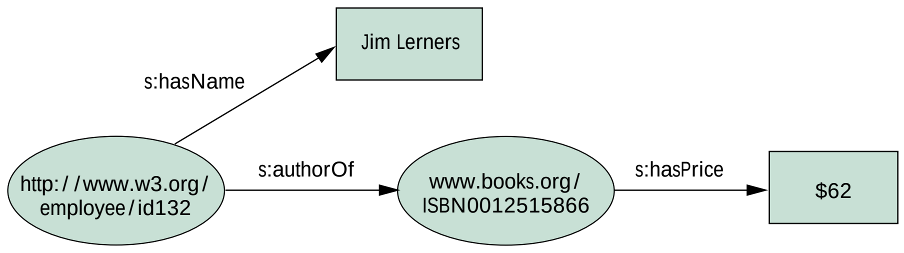

Jim Lerners
s:hasName
http://www.w3.org/employee/id132
s:authorOf
www.books.org/ISBN0012515866
s:hasPrice
$62

Figure 3. RDF graph. //Needs 1-2 sentences to describe the contents of the figure to readers scanning the article. Figures must stand on their own.//

in a Web page with an XSL style sheet to render the different elements appropriately.[7]

- *Uniform data-exchange format*. An XML serialization can also be transferred as a data object between two applications.

It is important to note that a DTD only specifies syntactic conventions; any intended semantics are outside the realm of the XML specification.

RDF: RESOURCE METADATA

The Resource Description Framework is a recent W3C recommendation designed to standardize the definition and use of metadata—descriptions of Web-based resources.[8] However, RDF is equally well suited to representing data.

RDF Foundations

The basic building block in RDF is an object-attribute-value triple, commonly written as $A(O,V)$. That is, an object O has an attribute A with value V. Another way to think of this relationship is as a labeled edge between two nodes: $[O]–A\rightarrow[V]$. //AU: Please clarify the notation here. Is this a vector?// This notation is useful because RDF allows objects and values to be interchanged. Thus, any object can play the role of a value, which amounts to chaining two labeled edges in a graphic representation. The graph in Figure 3, for example, expresses the following three relationships in $A(O,V)$ format:

```
hasName
    ('http://www.w3.org/employee/id1321',
    "Jim Lerners")
authorOf
    ('http://www.w3.org/employee/id1321',
    'http://www.books.org/ISBN0012515866')
hasPrice
    ('http://www.books.org/ISBN0012515866',
    "$62").
```

RDF also allows a form of reification in which any RDF statement can be the object or value of a triple, which means graphs can be nested as well as chained. On the Web this allows us, for example, to express doubt or support of statements created by other people. Finally, it is possible to indicate that a given object is of a certain type, such as stating that "ISBN0012515866" is of the rdf:type *book*, by creating a type arc referring to the *book* definition in an RDF schema:

```
<rdf:Description about="www.books.org/
        ISBN0012515866">
    <rdf:type resource="http://description.org/
        schema/book">
</rdf:Description>
```

It is important to note that RDF is designed to provide a basic object-attribute-value data model for metadata. Other than this intentional semantics—described only informally in the standard—RDF makes no data-modeling commitments. In particular, no reserved terms are defined for further data modeling. As with XML, the RDF data model provides no mechanisms for declaring property names that are to be used.

RDF Schema

Just as XML schema provides a vocabulary-definition facility, RDF schema lets developers define a particular vocabulary for RDF data (such as authorOf) and specify the kinds of object to which these attributes can be applied.[9] In other words, the RDF schema mechanism provides a basic type system for RDF models. This type system uses some predefined terms, such as Class, subPropertyOf, and subClassOf, for application-specific schema. RDF schema expressions are also valid RDF expressions (just as XML schema expressions are valid XML).

RDF objects can be defined as instances of one or

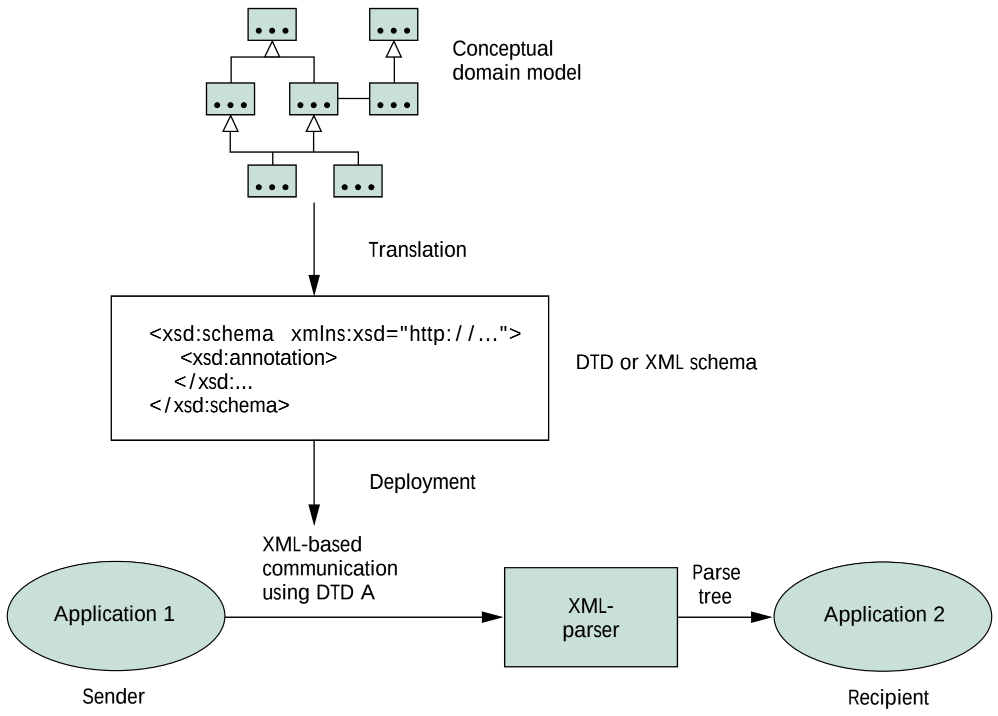


Figure 4. DTD development and point-to-point communication with XML. The conceptual domain model has to be translated to a DTD or XML schema before it can be deployed for … //AU Please clarify the process. Figures must stand on their own.//

more classes using the type property. The subClassOf property allows the developer to specify the hierarchical organization of such classes, and subPropertyOf does the same for properties. Constraints on properties can also be specified using domain and range constructs, which can be used to extend both the vocabulary and the intended interpretation of RDF expressions. This is the mechanism we used to translate an ontology representation language to RDF.

KNOWLEDGE REPRESENTATION

The Web is the first widely exploited many-to-many data-interchange medium, and it poses new requirements for any exchange format:

- *Universal expressive power.* Because it is not possible to anticipate all potential uses, a Web-based exchange format must be able to express any form of data.
- *Syntactic interoperability.* Applications must be able to read the data and get a representation that can be exploited. Software components like parsers or query APIs, for instance, should be as reusable as possible among different applications. Syntactic interoperability is high when the parsers and APIs needed to manipulate data are readily available.
- *Semantic interoperability.* One of the most important requirements for an exchange format is that data be understandable. Whereas syntactic interoperability is about parsing data, semantic interoperability is about defining mappings between terms within the data, which requires content analysis.

Using XML

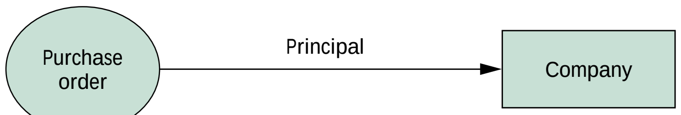
XML fulfills the universal expressive power requirement because anything for which a grammar can be defined can be encoded in XML. It also fulfills the syntactic interoperability requirement because an XML parser can parse any XML data, and is usually a reusable component. When it comes to semantic interoperability, however, XML has disadvantages.

XML's major limitation is that it just describes grammars. There is no way to recognize a semantic unit from a particular domain because XML aims at document structure and imposes no common interpretation of the data contained in the document. Although this limitation is at the heart of the "schema-wars" currently raging at forums such as

Biztalk.com and RosettaNet.org, it is not yet widely recognized.

Fixed vs. flexible communication. Figure 4 shows two applications trying to communicate with each other. Both agree on the use and intended meaning of the document structure given by DTD *A*, but a model of the domain of interest must be built to clarify the kind of data being sent before the data can be exchanged. (This model is usually described in terms of objects and relations, as it is in Unified Modeling Language[10] or entity-relationship modeling.[11]) A DTD or an XML schema is then constructed from the domain model—usually in an ad-hoc way.

Figure 5a shows a simple binary relationship, *principal* (owner), between two concepts, *purchase order* and *company*. Figure 5b shows several possibilities for encoding the relationship in XML. A DTD just describes a grammar, and there are multiple ways to encode any given domain model into a DTD, so no direct connection remains between them. It is impossible to determine from the DTDs the concepts and relation between them, and significantly more encoding options exist when there are, for example, multiple ordered relationships. It is thus difficult to reengineer the domain model from the DTDs. (Note, however, that the relationship depicted in Figure 5a represents a valid RDF model.)



(a)

Encoding DTD	Example XML Instance Data
<!ELEMENT PurchaseOrder (principal)> <!ATTLIST PurchaseOrder id ID #REQUIRED> <!ELEMENT principal (Company)> <!ATTLIST Company id ID #IMPLIED>	<PurchaseOrder id="X"> <principal> <Company id="Y"/> </principal> <PurchaseOrder>
<!ELEMENT principal (PurchaseOrder, Company)> <!ELEMENT PurchaseOrder (#CDATA)> <!ELEMENT Company (#CDATA)>	<principal> <PurchaseOrder>X</PurchaseOrder> <Company>Y</Company> </principal>
<!ELEMENT PurchaseOrder (id, principal)> <!ELEMENT id (#CDATA)> <!ELEMENT principal (Company)> <!ELEMENT Company (id)>	<PurchaseOrder> <id>X</id> <principal> <Company> <id>Y</id> </Company> </principal> </PurchaseOrder>

<!ELEMENT rel EMPTY>
<!ATTLIST rel
src CDATA #REQUIRED
type CDATA #REQUIRED
dest CDATA #REQUIRED>
	<rel
	src="X"
	type="principal"
	dest="Y"/>

<!ELEMENT PurchaseOrderInfo (Company)> <!ATTLIST PurchaseOrderInfo orderID ID #REQUIRED> <!ELEMENT Company (#CDATA)>	<PurchaseOrderInfo orderID="X"> <Company>Y</Company> </PurchaseOrderInfo>

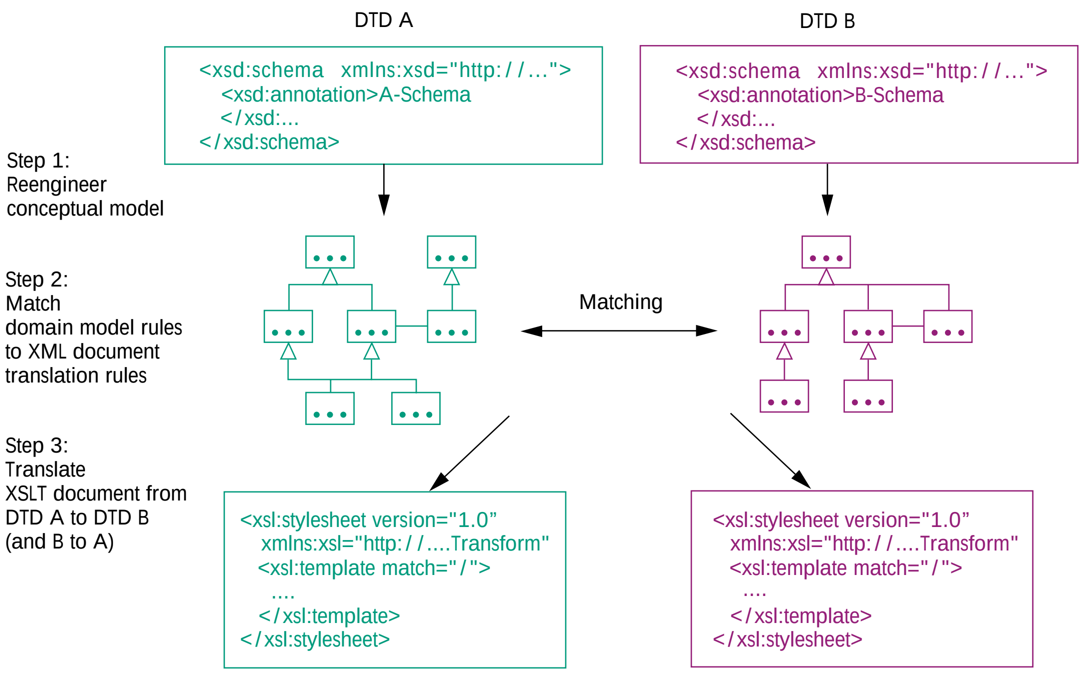
(b)

Figure 5. Binary relationship (a) and XML encoding possibilities (b). //Needs a sentence here to describe the content so the figure stands on its own.//

The advantage of using XML in this case is limited to the reusability of the parsing software components. This is certainly useful, but this scenario deals with a one-on-one communication between parties with an advance agreement. It neglects the reality of the Web, which requires communicating with multiple partners that change frequently.

Not the silver bullet. XML is useful for data interchange between applications that both know what the data is, but not for situations where new communication partners are frequently added. On the Web, new information sources continually become available and new business partners join existing relationships. It is thus important to reduce the costs of adding communication partners as much as possible. As the steps in Figure 6 show, however, using XML and DTDs (or schema) for this operation requires much more effort than necessary.

One domain model cannot be mapped to another because they are both encoded in DTDs. A direct mapping based on the different DTDs is not possible as the task is not to map grammars to each other, but to map objects and relations between domains of interest. Therefore, we must reengineer the original domain models and define the mappings between the concepts and relationships. (Techniques developed in knowledge engineering and


DTD A
DTD B
Step 1: Reengineer conceptual model
Step 2: Match domain model rules to XML document translation rules
Step 3: Translate XSLT document from DTD A to DTD B (and B to A)
Matching
...

```
<xsd:schema  xmlns:xsd="http://...">
  <xsd:annotation>A-Schema
  </xsd:...
</xsd:schema>
```

```
<xsd:schema  xmlns:xsd="http://...">
  <xsd:annotation>B-Schema
  </xsd:...
</xsd:schema>
```

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://....Transform"
  <xsl:template match="/">
    ....
  </xsl:template>
</xsl:stylesheet>
```

```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://....Transform"
  <xsl:template match="/">
    ....
  </xsl:template>
</xsl:stylesheet>
```

Figure 6. Alignment of conceptual models. Several additional steps are necessary in XML to add new communication partners to an existing communication, including the reengineering of the conceptual model used to construct the DTDs.

database research are often helpful for this task.[12,13]) Afterwards, the additional step of defining the mapping between DTDs must be performed.

To exchange XML documents, the domain mappings must be translated using mapping procedures such as XSL Transformations (XSLT) for grammars. This is again a high-effort task and depends on the encoding used to construct the initial DTDs. Additional effort is required in translating the reengineered domain model into an XML DTD and generating mapping procedures for XML documents based on established domain mappings. Using a more suitable formalism than pure XML for data transfer can save much of this additional effort.

Using RDF

RDF's nested object-attribute-value structure satisfies our universal expressive power requirement for an exchange format, although this is not easy to see. Application-independent RDF parsers are also available, so RDF fulfills our syntactic interoperability requirement as well.

When it comes to semantic interoperability, RDF has significant advantages over XML: The object-attribute structure provides natural semantic units because all objects are independent entities. A domain model—defining objects and relationships—can be represented naturally in RDF, so translation steps are not necessary as they are with XML. To find mappings between two RDF descriptions, techniques from research in knowledge representation are directly applicable. Of course, this does not solve the general interoperability problem of finding semantics-preserving mappings between objects, but using RDF for data interchange raises the level of potential reuse of software components much beyond parser reuse, which is all XML offers. Furthermore, the RDF model (and software using the RDF model) can still be used even if the current XML syntax changes or disappears because RDF describes a layer independent of XML.

Ideally, we would like a universal shared knowledge-representation language to support the Semantic Web, but for a variety of pragmatic and technological reasons, this is unachievable in practice. Instead, we will have to live with a multitude of metadata representations. RDF contains as much knowledge-representation technology as can be shared between widely varying metadata languages. Furthermore, the RDF schema language is powerful enough to define richer languages on top of RDF's limited primitives.

ENRICHING RDF

Before showing how RDF can be enriched to define sophisticated data models, we recall Brach-

Modeling Semantics

There are two main approaches to modeling semantics in computer science: *declarative* and *procedural* semantics.

With declarative semantics, an expression E is given meaning by mapping it to another well-understood formalism, or by stating the conclusions or properties that follow from E. The expression can be understood without reference to any specific computational procedure, which is why this approach is dubbed "declarative."

Using procedural semantics, expression E is given meaning by referring to the behavior that some real or virtual procedure (or program, or machine) will exhibit on E. Often the only way to obtain the expression's meaning using procedural semantics is to simply execute the procedure and observe the outcome.

This difference between declarative and procedural semantics loosely coincides with the difference between the XML and RDF approaches to Web-page semantics. As we've argued, an XML expression has no inherent semantics, and its meaning is only determined by the actions that one or more programs undertake on it (whether tag-nesting is interpreted as part-of, or as a subtype-of, for instance). An RDF expression, on the other hand, has a specific declarative semantics (such as the intended meaning of subClassOf), and this is specified independently of any RDF processor; that is, all RDF processors must conform to the intended semantics.

Together with the W3C, we stand in a long tradition in Computer Science and AI, which argues that the declarative approach to semantics leads to more shareable and extensible information and knowledge sources. The arguments about XML vs. RDF do not change substantially when XML schema is used instead of XML DTDs for specifying XML document structure. Readers might be tempted to compare XML schema's "type-extension" mechanism with the "subclassOf" mechanism in RDF schema, but the similarity between them is only superficial. In fact, the type-extension mechanism cannot be used to model ontological subtypes at all: in XML schema, if type T´ is derived from type T, then elements of the derived type T´ are not necessarily members of the original type T. In the subClassOf relationship in RDF schema, on the other hand, a member of a subclass is also a member of the original super-class. As a result, subClassOf can be used to model ontological subtyping, whereas XML schema's type extension cannot.

man's distinction of the three layers in a knowledge representation system[14]:

- the *implementation level* consists of data structures for a particular implementation;
- the *logic level* defines, in an abstract way, the inferences that are performed by the system; and
- the *epistemological level* defines adequate representation primitives for expressing knowledge in a convenient way—usually those used by a knowledge engineer.

The epistemological level is usually defined by a grammar that defines the language of interest, but the representation primitives can also be regarded as an ontology—as objects of a particular domain. Using this view, domain-modeling techniques are again suitable for defining a knowledge representation language. Thus, the epistemological level is itself an ontology that defines the terms of the representation language. Defining an ontology in RDF means defining an RDF schema, which specifies all the concepts and relationships of the particular language. (One example of a language definition as an ontology is the RDF schema, which is defined in terms of itself.[9])

Defining a More Expressive Ontology Language

Figure 7 (next page) shows how the RDF schema mechanism can be used to define elements of OIL. The shaded ellipses are elements that must be added to the existing schema definition to obtain a schema for OIL.

We illustrate this principle with an example from the ontology in Figure 1. The definition of *herbivore* uses the subclass-of modeling primitive, which is a relation that identifies the two arguments as objects: the class (*herbivore*) and a sophisticated class expression (*animal AND NOT carnivore*). The class expression can justifiably be viewed as an object because the expression *animal AND NOT carnivore* indeed defines a new (unnamed) class.

Defining the language primitives as an ontology results in the RDF graph depicted in Figure 7, which defines several properties and classes. The class oil:ClassExpression is a placeholder class that groups various types of class expressions for definitional purposes. oil:AND and oil:NOT are two particular types of class expressions. The property oil:hasOperand is an auxiliary property needed to connect an operator-type class expression with another class expression.

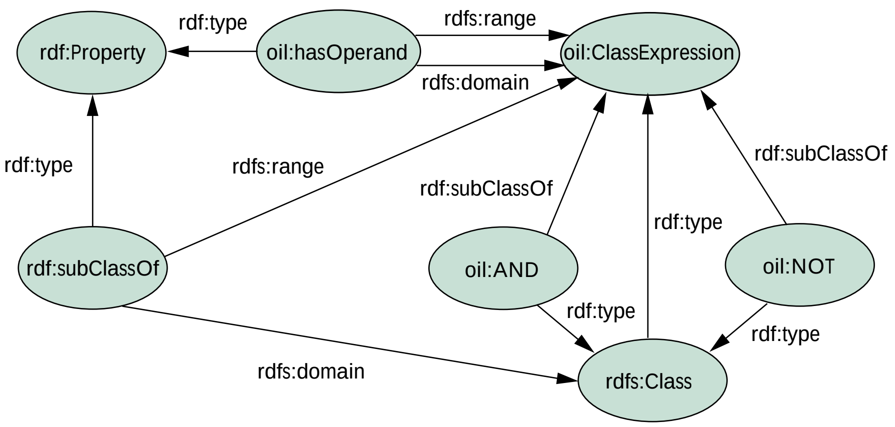


Figure 7. RDF graph for several OILproperties and classes.
/ / AU: Please 1-2 sentences to explain what the elements show at a high (graphical) level. / /

The existing description of the primitive rdfs:subClassOf is extended: the range now includes oil:ClassExpression (instead of just rdfs:Class). Also, the existing description of rdfs:Class is extended: It is also a subclass of oil:ClassExpression (because it is considered a particular type of class expression).

Using the graph in Figure 7, the expression:

```
class-def defined herbivore subclass-of animal,
   NOT carnivore
```

can be described in RDF as depicted in Figure 8.

Because every ontology (RDF schema) uses its own namespace (we chose the prefix *oil*), terms from different ontologies can be mixed in one RDF document without confusion.[3] RDF defines a clear object structure, so it is possible to make assertions with one language about an object defined in terms of another language. For example, we could mix the ontology with behavior statements about different animal classes using a finite-state automaton language.[15] This is not possible in XML because a tag's meaning (object, attribute, value, and so on) is not defined, and nothing can be assumed about the object structure.

Using OIL

The domain ontology defines a vocabulary—properties and classes—that can be used to write instance information in RDF. We propose a mechanism for extending the RDF data model with modeling primitives from any ontology language. Our approach for using the primitives from ontology language *L* to describe a particular domain has three steps:

- Step 1. Describe language *L*'s modeling primitives using RDF schema (effectively writing the meta-ontology of *L* in RDF Schema).
- Step 2. Describe a specific ontology in *L* using the resulting RDF schema document.
- Step 3. Use the RDF schema documents to describe instances of the specific *L* ontology modeled in step 2.

Table 1 lists the expression types for each step, along with ... //AU: What are the other two columns exactly?// This three-step approach suggests an additional requirement for an ontology language *L*: Because RDF schema is already an ontology definition language, *L* must be compatible with it. Thus, existing RDF schema processors can make maximum use of ontologies defined in *L*. The same tools are applicable for all possible *L*s, which leads to flexibility in designing customized languages.

MERGING AN ONTOLOGY LANGUAGE WITH RDF SCHEMA

Defining an ontology language as an extension of RDF schema means every RDF-schema ontology is valid in the new language (for example, an OIL processor will also understand RDF schema). However, by defining the new language as closely as possible to RDF schema, we also maximize reuse of existing RDF schema-based applications and tools. Because the ontology language usually contains vocabulary the RDF schema processor does not know, however, 100-percent compatibility is not possible.

In OIL, a class can be a subclass of a Boolean class expression, but the original RDF subclass statement only allows primitive classes as values. We thus had to extend the OIL subclass statement definition. We also introduced oil:SlotConstraint to allow restrictions on slots in class definitions—another aspect of OIL that is unavailable in RDFS. To maintain maximum compatibility with existing applications, however, it is recommended to use the RDF schema vocabulary wherever possible (for a case study on how to do this with OIL, see Broekstra et al.[16]).

RDF follows an object-attribute-value model, so we made a basic design decision that every OIL expression would be an object. To allow subexpressions, we introduced auxiliary attributes that do not correspond to any original OIL vocabulary (oil:hasOperand, oil:hasClass, and oil:hasProperty). Another difference is how slot expressions are handled: Slots are separate entities in RDF, so we had to treat slot expressions as subclass expressions, as in the underlying description logic framework.

The major integration points between RDF/RDFS and OIL are defined by the abstract OIL class ClassExpression. Furthermore, OIL-slots are realized as instances of rdf:Property or as subproperties of the original rdf:Property. The subslot relationship is also expressed by original RDF-means—namely, the rdfs:subPropertyOf relationship. rdf:Property is enriched by several properties that specify inverse and transitive roles and cardinality constraints, which were not originally possible in

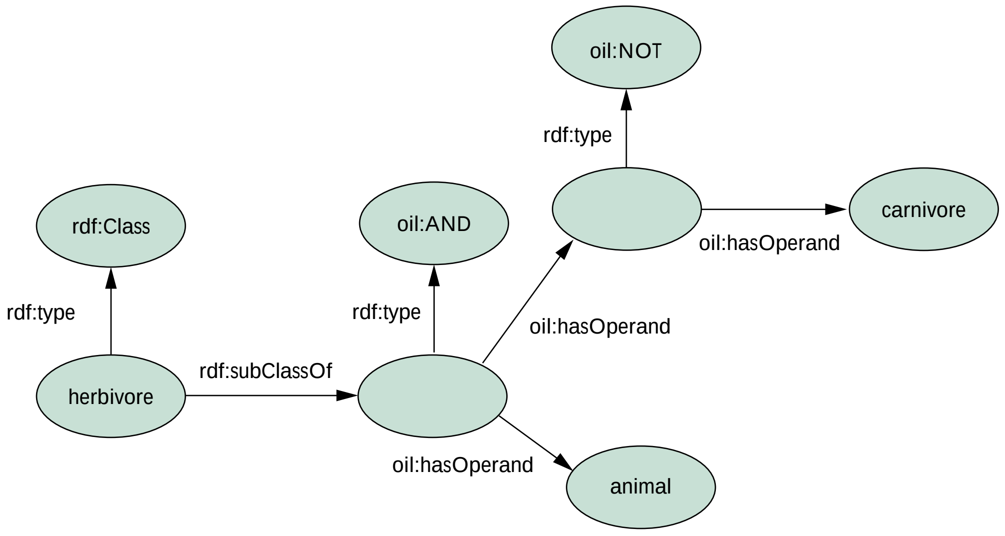


Figure 8. RDF graph for the RDFS-OIL definitions. / / AU: Please add 1-2 sentences explaining what this figure shows. Also, why are there blank ovals? / /

Table 1. Steps in the OIL approach to using an ontology language to extend RDF.

Step	Expression Type	Example	Encoding
1	Modeling primitives of ontology language *L*	oil:AND, oil:NOT, …	RDF: Metaontology in RDF schema
2	Specific ontology expressed in *L*	Class-def giraffe Subclass of herbivore Slot-constraint eats Value-type leaf	RDF: Ontology (using metaontology and RDF schema)
3	Instances of the specific ontology	animal12-eats-leaf34…	RDF (RDF schema, metaontology, and ontology)

Table 2. OIL vocabulary mapping to RDF schema.

OIL original vocabulary	RDF vocabulary
Class-def	rdfs:Class
Subclass-of	rdfs:subClassOf
Slot constraint	oil:hasSlotConstraint
	oil:SlotConstraint
AND	oil:AND
NOT	oil:NOT
Has-value	Oil:HasValue

RDF/RDFS (for details, see Horrocks et al.[2]).

Class definitions are inherited from the original rdfs:Resource from the RDF schema specification. Classes can be related with arbitrary class expressions via subclass or equivalence relationships. By doing this, existing classes from RDFS vocabularies can be accessed and refined in OIL descriptions. Table 2 contains some of the OIL mappings.

We have implemented an inference engine for OIL based on the description logic inference engine, FaCT (http://www.cs.man.ac.uk/~horrocks/FaCT/). We have also performed several case studies, including modeling an ontology for the CIA world fact book (see http://www.ontoknowledge.org/oil), to test the overall framework's flexibility. Furthermore, the DARPA Agent Markup Language (http://www.daml.org) uses the principles of OIL as well.

CHALLENGES

The Web community currently regards XML as the most important step towards semantic integration, but we argue that this is not true in the long run. Semantic interoperability will be a sine qua non for the semantic Web, but it must be achieved by exploiting the current RDF proposals, rather than XML labeling. The RDF data model is sound, and approaches from artificial intelligence and knowledge engineering for establishing semantic interoperability are directly applicable to extending it.

Our experience with OIL shows this proposal is feasible, and a similar strategy should apply to any knowledge-modeling language. The challenge is now for the Web and AI communities to expand this generic method for Web-enabling arbitrary knowledge-representation languages. ■

REFERENCES

1. T. Berners-Lee, *Weaving the Web*, Harper, San Francisco, 1999.
2. I. Horrocks, et al., "The Ontology Interchange Language OIL," tech. report, Free Univ. of Amsterdam, 2000; available online at http://www.ontoknowledge.org/oil/
3. T. Bray, J. Paoli, and C.M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0," W3C Recommendation, Feb. 1998; available online at http://www.w3.org/TR/REC-xml.
4. H. S. Thompson, et al., "XML Schema Part 1: Structures," W3C, working-in-progress, current as of Apr. 2000; available online at http://www.w3.org/TR/2000/WD-xmlschema-1-20000407/.
5. P. V. Biron and A. Malhotra, "XML Schema Part 2: Datatypes," working-in-progress, current as of Apr. 2000; available online at http://www.w3.org/TR/2000/WD-xmlschema-2-20000407/.
6. P. Hoschka, "Synchronized Multimedia Integration Language (SMIL) 1.0 Spec.," W3C Recommendation, June 1998; available online at http://www.w3.org/TR/REC-smil/.
7. J. Clark, "XSL Transformations (XSLT)," W3C Recommendation, Nov. 1999; available online at http://www.w3.org/TR/xslt/.
8. O. Lassila and Ralph Swick, "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, Feb. 1999; available online at http://www.w3.org/TR/REC-rdf-syntax/.
9. D. Brickley and R. Guha, "Resource Description Framework (RDF) Schema Specification," W3C Candidate Recommendation, March 2000; available online at http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
10. M. Page-Jones and L.L. Constantine, *Fundamentals of Object-Oriented Design in UML*, Addison-Wesley, Boston, Mass., 1999.
11. R. Barker, *Entity Relationship Modeling*, Addison-Wesley, Boston, Mass., 1990.
12. J. Jannink, et al., "An Algebra for Semantic Interoperation of Semistructured Data," *Proc. IEEE Knowledge and Data Engineering Exchange Workshop*, IEEE Comp. Soc. Press, Los Alamitos, Calif., 1999.
13. D.L. McGuinness, et al., "The Chimaera Ontology Environment," *Proc. Seventeenth Nat'l Conf. Artificial Intelligence (AAAI 2000)*, AAAI Press, Menlo Park, Calif., 2000.
14. R.J. Brachman, "On the Epistemological Status of Semantic Networks," in *Associative Networks: Representations and Use of Knowledge by Computers*, N.V. Findler, ed., Academic Press, 1979, pp. 3-50.
15. S. Melnik, H. Garcia-Molina, and A. Paepcke, "A Mediation Infrastructure for Digital Library Services," *Proc. ACM Digital Libraries Conf.*, ACM Press, June 2000.
16. Broekstra et al., "OIL: A Case Study in Extending RDF-Schema," tech. report, Vrije Universiteit, Amsterdam, 2000; available online at http://www.ontoknowledge.org/oil/.

Stefan Decker is a postdoctoral fellow at the computer science department at Stanford University and the cofounder of

Ontoprise, a startup company focusing on ontology-based knowledge management. He consults frequently on RDF, XML, and interoperability issues.

Sergey Melnik is a visiting researcher in the computer science department at Stanford University, where he works on database and interoperability topics with special attention to RDF and XML. His research interests include knowledge representation and database systems for the Web, information integration, and digital libraries.

Frank Van Harmelen is a senior lecturer with the artificial intelligence research group at the Vrije Universiteit, Amsterdam. His current interests include specification languages for KBS, using these languages for validation and verification of KBS, developing gradual notions of correctness for KBS, and verifying weakly structured data.

Dieter Fensel is an associate professor at Vrije Universiteit, Amsterdam. His research interests include problem-solving methods of knowledge-based systems, and the use of ontologies to mediate access to heterogeneous knowledge sources and apply them in knowledge management and electronic commerce.

Michel Klein is a PhD student at the Knowledge Engineering and Reasoning Group of the Vrije Universiteit in Amsterdam. His research interests include RDF and RDF-Schema, ontology modeling, maintenance and integration, as well as representation, visualization and querying of semistructured data.

Jeen Broekstra is a PhD student in the knowledge engineering and reasoning group of the Vrije Universiteit in Amsterdam, and a knowledge engineer working for Aidministrator Nederland B.V. His research interests include ontology modeling, maintenance, and integration, as well as representation, visualization, and querying of semistructured data.

Michael Erdmann is a PhD student at the Institute for Applied Computer Science and Formal Description Methods (AIFB) at the University of Karlsruhe. His research interests include semantic knowledge modeling with ontologies, and intelligent web applications based on KR techniques and open web standards.

Ian Horrocks is a lecturer in computer science at the University of Manchester, UK. His research interests include knowledge representation, automated reasoning (particularly decision procedures for description and modal logics), ontological engineering, and optimizing, testing, and evaluating reasoning systems.

Readers can contact the authors at stefan@db.stanford.edu.