

Ontologies in RDF(S)

Steffen Staab
Michael Erdmann
Alexander Maedche

Institute AIFB
Karlsruhe University
76128 Karlsruhe, Germany

`{staab, erdmann, maedche}@aifb.uni-karlsruhe.de`

<http://www.aifb.uni-karlsruhe.de/WBS>

Linköping University Electronic Press
Linköping, Sweden

<http://www.ep.liu.se/ea/cis/2001/???/>

*Published on 30. December, 2001 by
Linköping University Electronic Press
581 83 Linköping, Sweden*

**Linköping Electronic Articles in
Computer and Information Science**
*ISSN 1401-9841
Series editor: Erik Sandewall*

©2001 S. Staab, M. Erdmann, A. Maedche
*Typeset by the authors using L^AT_EX
Formatted using étendu style*

Recommended citation:

*<Authors>. <Title>. Linköping Electronic Articles in
Computer and Information Science, Vol. 6(2001): nr ?.
<http://www.ep.liu.se/ea/cis/2001/???/>. 30. December, 2001.*

This URL will also contain a link to the authors' home pages.

*The publishers will keep this article on-line on the Internet
(or its possible replacement network in the future)
for a period of 25 years from the date of publication,
barring exceptional circumstances as described separately.*

*The on-line availability of the article implies
a permanent permission for anyone to read the article on-line,
to print out single copies of it, and to use it unchanged
for any non-commercial research and educational purpose,
including making copies for classroom use.
This permission can not be revoked by subsequent
transfers of copyright. All other uses of the article are
conditional on the consent of the copyright owners.*

*The publication of the article on the date stated above
included also the production of a limited number of copies
on paper, which were archived in Swedish university libraries
like all other written works published in Sweden.
The publisher has taken technical and administrative measures
to assure that the on-line version of the article will be
permanently accessible using the URL stated above,
unchanged, and permanently equal to the archived printed copies
at least until the expiration of the publication period.*

*For additional information about the Linköping University
Electronic Press and its procedures for publication and for
assurance of document integrity, please refer to
its WWW home page: <http://www.ep.liu.se/>
or by conventional mail to the address stated above.*

Abstract

RDF(S)¹ constitutes a newly emerging standard for metadata that is about to turn the World Wide Web into a machine-understandable knowledge base. It is an XML application that allows for the denotation of facts and schemata in a web-compatible format, building on an elaborate object-model for describing concepts and relations. Thus, it might turn up as a natural choice for a widely-useable ontology description language. However, its lack of capabilities for describing the semantics of concepts and relations beyond those provided by inheritance mechanisms makes it a rather weak language for even the most austere knowledge-based system. This paper presents an approach for modeling ontologies in RDF(S) that also considers axioms as objects that are describable in RDF(S). Thus, we provide flexible, extensible, and adequate means for accessing and exchanging axioms in RDF(S). Our approach follows the spirit of the World Wide Web, as we do not assume a global axiom specification language that is too intractable for one purpose and too weak for the next, but rather a methodology that allows (communities of) users to specify what axioms are interesting in their domain.

This paper is a revised version of a paper presented at the ECDL-2000 Workshop on *Semantic Web*, Lisbon, Portugal, September 2000.

¹We use “RDF(S)” to refer to the combined technologies of RDF and RDFS.

1 Introduction

The development of the World Wide Web is about to mature from a technical platform that allows for the transportation of information from sources to humans (albeit in many syntactic formats) to the communication of knowledge from Web sources to machines. The knowledge food chain has started with technical protocols and preliminary formats for information presentation (HTML – HyperText Markup Language) over a general methodology for separating information contents from layout (XML – eXtensible Markup Language, XSL – eXtensible Stylesheet Language) to reach the realms of knowledge provisioning by the means of RDF and RDFS.

RDF (Resource Description Framework) is a W3C recommendation (Lassila & Swick, 1999) that provides description facilities for knowledge pieces, *viz.* for triples that denote relations between pairs of objects. To exchange and process RDF models they can be serialized in XML. RDF exploits the means of XML to allow for disjoint namespaces, linking and referring between namespaces and, hence, is a general methodology for sharing machine-processable knowledge in a distributed setting. On top of RDF the simple schema language *RDFS* (Resource Description Framework Schema; (Brickley & Guha, 1999)) has been defined to offer a distinguished vocabulary to model class and property hierarchies and other basic schema primitives that can be referred to from RDF models. To phrase the role of RDFS in knowledge engineering terminology, it allows to define a simple *ontology* that particular RDF documents may be checked against to determine consistency.

Ontologies have shown their usefulness in application areas such as intelligent information integration or information brokering. Therefore their use is highly interesting for web applications, which may also profit from long term experiences made in the knowledge acquisition community. Nevertheless, while support for modeling of ontological concepts and relations has been extensively provided in RDF(S), the same cannot be said about the modeling of ontological axioms — one of the key ingredients in ontology definitions and one of the major benefits of ontology applications.

RDF(S) offers only the most basic modeling primitives for ontology modeling. Even though there are good and bad choices for particular formal languages, one must face the principal trade-off between tractability and expressiveness of a language. RDF(S) has been placed nearer to the low end of expressiveness, because it has been conceived to be applicable to vast web resources! In contrast to common knowledge representation languages, RDF(S) has not been meant to be the definitive answer to all knowledge representation problems, but rather an *extensible core language*. The namespace and reification mechanisms of RDF(S) allow (communities of) users to define their very own standards in RDF(S) format — extending the core definitions and semantics. As RDF(S) leaves the well-trodden paths of knowledge engineering at this point, we must reconsider crucial issues concerning ontology modeling and ontology applications. To name but a few, we mention the problem of merging and mapping between namespaces, scalability issues, or the definition and usage of ontological axioms.

In this paper we concentrate on the latter, namely on how to model axioms in RDF(S) following the stipulations, *(i)*, that the core semantics of RDF(S) is re-used such that “pure” RDF(S) applications may still process the core object-model definitions, *(ii)*, that the semantics is preserved between different inferencing tools (at least to a large extent), and, *(iii)*, that axiom modeling is adaptable to reflect diverging needs of different com-

munities. Current proposals neglect or even conflict with one or several of these requirements. For instance, the first requirement is violated by the ontology exchange language XOL (Karp et al., 1999) making *all* the object-model definitions indigestible for most RDF(S) applications. The interchangeability and adaptability stipulation is extremely difficult to meet by the parse-tree-based representation of MetaLog (Marchiori & Saarela, 1998), since it obliges to first-order logic formulae. We will show how to adapt a general methodology that we have proposed for axiom modeling (Staab & Maedche, 2000) to be applied to the engineering of ontologies with RDF(S). Our approach is based on translations of RDF(S) axiom specifications into various target systems that provide the inferencing services. As our running example, we map axiom specifications into an F-Logic format that has already served as the core system for SiLRi, an inference service for core RDF (Decker et al., 1998). Our methodology is centered around categorization of axioms, because this allows for a more concise description of the *semantic meaning* rather than a particular syntactic representation of axioms. Thus, we get a better grip on extensions and adaptations to particular target inferencing systems.

In the following, we introduce the RDF(S) data model and describe how to define an object model in RDF(S) including practical issues of ontology documentation (Section 2). Then we describe our methodology for using RDF(S) such that axioms may be engineered and exchanged. We describe the core idea of our approach and illustrate with several examples how to realize our approach (Section 3). In a case study (Section 4) we illustrate the application of our approach in our ontology engineering environment, *OntoEdit*, and our semantic community web portal, *KA2Portal* (Staab et al., 2000). Before we conclude, we give a brief survey of related work.

2 Modeling Concepts and Relations in RDF(S)

In this section we will first take a look at the core ontology engineering task, i.e. at the RDF(S) data model proper, and then exploit RDF(S) also for purposes of practical ontology engineering, viz. for documentation of newly defined or reused ontologies. This will lay the groundwork for the modeling of axioms in Section 3.

2.1 The RDF(S) Data Model

RDF(S) is an abstract data model that defines relationships between entities (called resources in RDF) in a similar fashion as semantic nets. Statements in RDF describe resources, that can be web pages or surrogates for real world objects like publications, pieces of art, persons, or institutions. We illustrate how concepts and relations can be modelled in RDF(S) by presenting a sample ontology in the abstract data model and only afterwards show how these concepts and relations are presented in the XML-serialisation of RDF(S).

2.1.1 RDF

As already mentioned RDF(S) consists of two closely related parts: RDF and RDF Schema. The foundation of RDF(S) is laid out by RDF which defines basic entities, like resources, properties, and statements. Anything in RDF(S) is a resource. Resources may be related to each other or to literal (i.e. atomic) values via properties. Such a relationship represents a

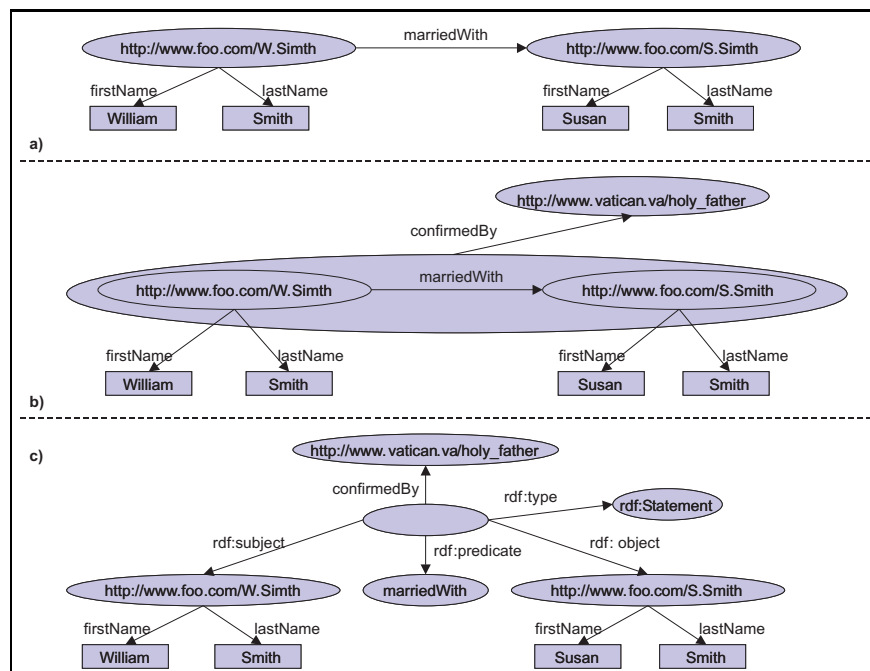


Figure 1: An example RDF data model.

statement that itself may be considered a resource, i.e. reification is directly built into the RDF data model. Thus, it is possible to make statements about statements. These basic notions can be easily depicted in a graphical notation that resembles semantic nets. To illustrate the possibilities of pure RDF the following statements are expressed in RDF and depicted in Figure 1²:

- Firstly, in part (a) of Figure 1 two resources are defined, each carrying a `FIRSTNAME` and a `LASTNAME` property with literal values, identifying the resources as William and Susan Smith, respectively. These two resources come with a URI as their unique global identifier and they are related via the property `MARRIEDWITH`, which expresses that William is married with Susan.
- Part (b) of the illustration shows a convenient shortcut for expressing more complex statements, i.e. reifying a statement and defining a property for the new resource. The example denotes that the marriage between William and Susan has been confirmed by the resource representing the Holy Father in Rome.
- The RDF data model offers the predefined resource `rdf:statement` and the predefined properties `rdf:subject`, `rdf:predicate`, and `rdf:object` to reify a statement as a resource. The actual model for the example (b) is depicted in part (c) of Figure 1. Note that the reified statement makes no claims about the truth value of what is reified, i.e. if one wants to express that William and Susan are married *and* that this marriage has been confirmed by the pope then the actual data model must contain a union of part (a) and part (c) of the example illustration.

²Resources are represented by ovals, literal values by shaded rectangles and properties by directed, labeled arcs.

2.1.2 RDFS

As a companion standard to RDF, the schema language RDFS is more important with respect to ontological modeling of domains. RDFS offers a distinguished vocabulary defined on top of RDF to allow the modelling of object models with cleanly defined semantics. The terms introduced in RDFS build the groundwork for the extensions of RDF(S) that are proposed in this paper. The relevant RDFS terms are presented in the following list.

- The most general class in RDF(S) is `rdfs:Resource`. It has two subclasses, namely `rdfs:Class` and `rdf:Property` (cf. Figure 2³). When specifying a domain specific schema for RDF(S), the classes and properties defined in this schema will become instances of these two resources.
- The resource `rdfs:Class` denotes the set of all classes in an object-oriented sense. That means, that classes like `appl:Person` or `appl:Organisation` are instances of the meta-class `rdfs:Class`.
- The same holds for properties, i.e. each property defined in an application specific RDF schema is an instance of `rdf:Property`, e.g. `appl:marriedWith`
- RDFS defines the special property `rdfs:subClassOf` that defines the subclass relationship between classes. Since `rdfs:subClassOf` is transitive, definitions are inherited by the more specific classes from the more general classes and resources that are instances of a class are automatically instances of all superclasses of this class. In RDF(S) it is prohibited that any class is an `rdfs:subClassOf` itself or of one of its subclasses.
- Similar to `rdfs:subClassOf`, which defines a hierarchy of classes, another special type of relation `rdfs:subPropertyOf` defines a hierarchy of properties, e.g. one may express that `FATHEROF` is an `rdfs:subPropertyOf` `PARENTOF`.
- RDFS allows to define the domain and range restrictions associated with properties. For instance, these restrictions allow the definition that persons and only persons may be `MARRIEDWITH` and only with other persons.

As depicted in the middle layer of Figure 2 the domain specific classes `appl:Person`, `appl:Man`, and `appl:Woman` are defined as instances of `rdfs:Class`. In the same way domain specific property types are defined as instances of `rdf:Property`, i.e. `APPL:MARRIEDWITH`, `APPL:FIRSTNAME`, and `APPL:LASTNAME`.

2.1.3 The use of XML Namespaces in RDF(S)

The XML namespace mechanism plays a crucial role for the development of RDF schemata and applications. It allows to distinguish between different modeling layers (cf. Figure 2 and 3) and to reuse and integrate existing schemata and applications. At the time being, there exist a number of *canonical* namespaces, e.g. for RDF, RDFS, and Dublin Core (cf. Section 2.2). We here introduce two new namespaces that aim at two different

³The reader may note that only a very small part of RDF(S) is depicted in the RDF/RDFS layer of the figure. Furthermore, the relation `APPL:MARRIEDWITH` in the data layer is identical to the resource `APPL:MARRIEDWITH` in the schema layer.

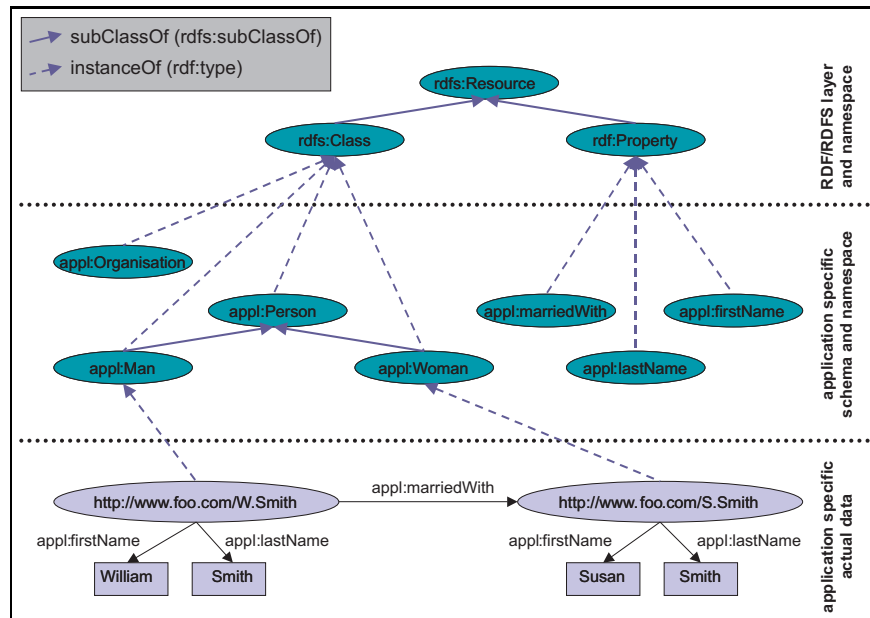


Figure 2: An example RDF schema and its embedding in RDF(S).

objectives, viz. the comprehensive documentation of ontologies and the capturing of our proposal for the modeling of ontological axioms

An actual ontology definition occurs at a concrete URL⁴. It defines shorthand notations which refer to our actual namespaces for ontology documentation and modeling of ontological axioms, abbreviated `oDoc` and `o`, respectively. An actual application that uses our example ontology will define a shorthand identifier like `appl` in order to refer to this particular, application-specific ontology. Figures 2 and 3 presume these shorthand notations for the namespaces we have just mentioned.

2.1.4 XML serialization of RDF(S)

One important aspect for the success of RDF in the WWW is the way RDF models are represented and exchanged, namely via XML. In the following excerpt of the RDF schema document <http://ontoserver.aifb.uni-karlsruhe.de/schema/example.rdf>, the classes and property types defined in Figure 2 are represented in XML and the domains and ranges of the properties are defined using the RDF constraint properties `rdfs:domain` and `rdfs:range`.

```
<rdf:Description ID="Person">
  <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
  <rdfs:subClassOf
    rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
</rdf:Description>

<rdf:Description ID="Man">
  <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
  <rdfs:subClassOf rdf:resource="#Person"/>
</rdf:Description>

<rdf:Description ID="Woman">
  <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
  <rdfs:subClassOf rdf:resource="#Person"/>
</rdf:Description>
```

⁴The reader may actually compare with the documents that appear at these URLs, e.g. <http://ontoserver.aifb.uni-karlsruhe.de/schema/example.rdf>

```

<rdf:Description ID="Organisation">
  <rdf:type resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Class"/>
  <rdfs:subClassOf
    rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
</rdf:Description>

<rdf:Description ID="firstName">
  <rdf:type
    resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Property"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string"/>
</rdf:Description>

<rdf:Description ID="lastName">
  <rdf:type
    resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Property"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string"/>
</rdf:Description>

<rdf:Description rdf:ID="marriedWith">
  <rdf:type
    resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Property"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Description>

```

2.2 Modeling ontology metadata using RDF Dublin Core

Metadata about ontologies, such as the title, authors, version, statistical data, etc. are important for practical tasks of ontology engineering and exchange. In our approach we have adopted the well-established and standardized RDF Dublin Core Metadata element set (Weibel & Miller, 1998). This element set comprises fifteen elements which together capture basic aspects related to the description of resources. Ensuring a maximal level of generality and exchangeability, our ontologies are labeled using this basic element set. Since ontologies represent a very particular class of resource, the general Dublin Core metadata description does not offer sufficient support for ontology engineering and exchange. Hence, we describe further semantic types in the schema located at

<http://ontoserver.aifb.uni-karlsruhe.de/schema/ontodoc>

and instantiate these types when we build a new ontology. The example below illustrates our usage and extension of Dublin Core by an excerpt of an exemplary ontology metadata description.

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc = "http://purl.oclc.org/dc"
  xmlns:odoc = "http://ontoserver.aifb.uni-karlsruhe.de/schema/ontodoc">

  <rdf:Description about = "">
    <dc:title>An Example Ontology</dc:title>
    <dc:creator>
      <rdf:Bag>
        <rdf:li>Steffen Staab</rdf:li>
        <rdf:li>Michael Erdmann</rdf:li>
        <rdf:li>Alexander Maedche</rdf:li>
      </rdf:Bag>
    </dc:creator>
    <dc:date>2000-02-29</dc:date>
    <dc:format>text/xml</dc:format>
    <dc:description>
      An example ontology modeled for this small application
    </dc:description>
    <dc:subject>Ontology, RDF</dc:subject>

    <odoc:url>http://ontoserver.aifb.uni-karlsruhe.de/schema/example.rdf</odoc:url>
    <odoc:version>2.1</odoc:version>
    <odoc:last_modification>2000-03-01</odoc:last_modification>
    <odoc:ka_technique>semi-automatic text knowledge acquisition</odoc:ka_technique>

```

```

<odoc:ontology_type>domain ontology</odoc:ontology_type>
<odoc:no_concepts>24</odoc:no_concepts>
<odoc:no_relations>13</odoc:no_relations>
<odoc:no_axioms>9</odoc:no_axioms>
<odoc:highest_depth_level>4</odoc:highest_depth_level>
</rdf:Description>
</rdf:RDF>

```

3 Modeling of Axioms in RDF(S)

Having prepared the object-model and documentation backbone for ontologies in RDF(S), we may now approach the third pillar of our approach, viz. the specification of axioms in RDF(S). The basic idea that we pursue is the specification and serialization of axioms in RDF(S) such that they remain easily representable and exchangeable between different ontology engineering, representation and inferencing environments. The principal specification needs to be rather independent of particular target systems (to whatever extent this is possible at all) in order to be of value in a distributed web setting with many different basic applications.

3.1 Axioms are Objects, too

Representation of interesting axioms that are deemed to be applied in different inferencing applications turns out to be difficult. The reason is that typically some kind of non-propositional logic is involved that deals with quantifiers and quantifier scope. Axioms are difficult to grasp, since the representation of quantifier scope and its likes is usually what the nitty-gritty details of a particular syntax, on which a particular inferencing application is based, are about. An ontology representation in RDF(S) should, however, abstract from particular target systems.

A closer look at the bread and butter issues of ontology modeling reveals that many axioms that need to be formulated aim at much simpler purposes than arbitrary logic structures. Indeed, we have found that many axioms in our applications belong to one of a list of major axiom categories:

1. Axioms for a relational algebra
 - (a) Reflexivity of relations
 - (b) Irreflexivity of relations
 - (c) Symmetry of relations
 - (d) Asymmetry of relations
 - (e) Antisymmetry of relations
 - (f) Transitivity of relations
 - (g) Inverse relations
2. Composition of relations⁵
3. (Exhaustive) Partitions⁶
4. Axioms for subrelation relationships
5. Axioms for part-whole reasoning

⁵E.g., FATHERINLAWOF is composed by the relations FATHEROF and MARRIEDWITH.

⁶E.g., concepts *Man* and *Woman* share no instances.

6. Axioms that are derivations of the above mentioned

- (a) Locally symmetric relations
- (b) Locally transitive relations
- (c) Locally inverse relations

Our principal idea for representing ontologies with axioms in RDF(S) is based on this categorization. The categories allow to distinguish between the structures that are repeatedly found in axiom specifications from a corresponding description in a particular language. Hence, one may describe axioms as complex objects (one could term them instantiations of axiom schemata) in RDF(S) that refer to concepts and relations, which are also denoted in RDF(S). For sets of axiom types we presume the definition of different RDF schemata. Similar to the case of simple metadata structures, the RDF schema responsible for an axiom categorization obliges to a particular semantics of its axiom types — which may be realized in a number of different inferencing systems like description logics systems (e.g., (Horrocks, 1998)) or frame logic systems (Decker et al., 1998). The schema defined in our namespace `http://ontoserver.aifb.uni-karlsruhe.de/schema/ontordf` stands for the semantics defined in this and our previous papers (Maedche et al., 2000; Staab & Maedche, 2000).⁷ The schema is also listed in the appendix of this paper (cf. Section A). Other communities may, of course, find other reasoning schemes more important, or they may just need an extension compared to what we provide here.

At the *symbol level*, we provide a RDF(S) syntax (i.e. serialization) to denote particular types of axioms. The categorization really constitutes a *knowledge level* that is independent from particular machines. In order to use an ontology denoted with our RDF(S) approach, one determines the appropriate axiom category and its actual instantiation found in a RDF(S) piece of ontology, translates it into a corresponding logical representation and executes it by an inferencing engine that is able to reason with (some of) the relevant axiom types.

Figure 3 summarizes our approach for modeling axiom specifications in RDF(S). It depicts the core of the RDF(S) definitions and our extension for axiom categorizations (i.e. our ontology meta layer). A simple ontology, especially a set of application specific relationships, is defined in terms of our extension to RDF(S).

In the following subsections, we will further elucidate our approach by proceeding through a few simple examples of our categorization of axiom specifications listed above. In particular our scheme is, (*A*) to show the representations of axioms in RDF(S) and (*B*) to show a structurally equivalent F(rame)-Logic representation that may easily be derived from its RDF(S) counterpart (cf. (Kifer et al., 1995; Decker, 1998) on F-Logic). Then, (*C*) we exploit the expressiveness of F-Logic in order to specify translation axioms that work directly on the F-Logic object representation of axioms. Thus, (*B*) in combination with (*C*) describes a formally concise and executable translation. For better illustration, we finally, (*D*), indicate the result of our translation by exemplary target representations of the axioms stated in RDF(S).

The reader should note here that we do neither believe that F-Logic fulfills all the requirements that one might wish from an ontology inferencing language, nor do we believe that the axiom types we mention exhaust all

⁷The reader may note that we have chosen names to coincide with many conventional names, e.g. “symmetry” of relations.

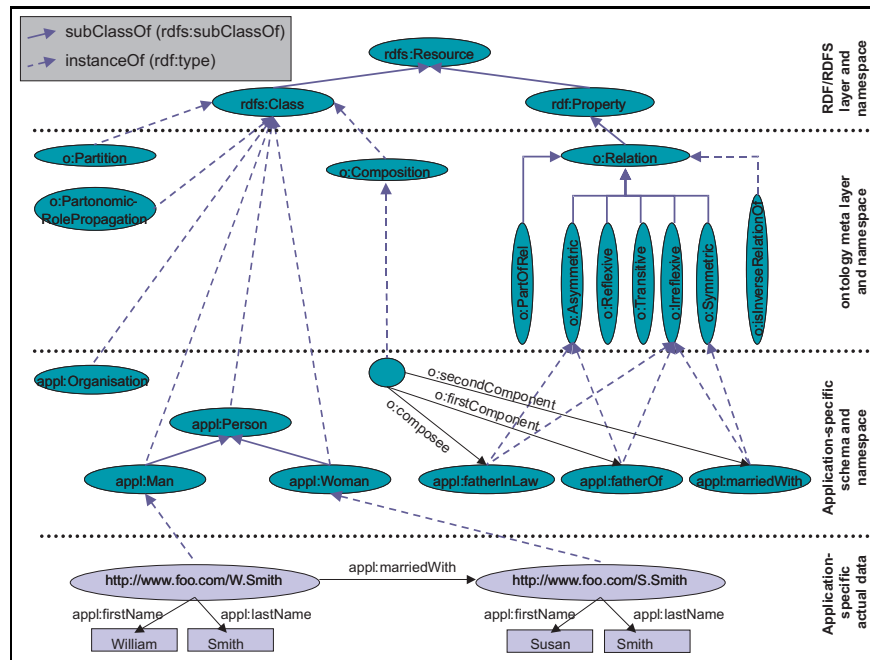


Figure 3: An excerpt of the example object model and an instantiation of classes, properties, and axioms in RDF(S)

relevant types. Rather we believe that our experiences in particular domains will push for further categorizations of axioms, further translation mechanisms, and, hence, further extensions of the core RDF(S) representation. All that will have to be agreed upon by communities that want to engineer and exchange ontologies with interesting axioms across particularities of inference engines. Our main objective is to acquaint the reader with our *principle methodology* that is transportable to other translation approaches, inferencing systems, and other axiom types, when need arises.

3.2 Axioms for a relational algebra

The axiom types that we have shown above are listed such that simpler axioms tend to appear first. Axiom specifications that are referred to as “axioms for a relational algebra” rank among the simplest ones. They describe axioms with rather local effects, because their implications only affect one or two relations. We here show one simple example of these in order to explain the basic approach and some syntax. The principle approach easily transfers to all axiom types from 1.(a)-(g) to 5.

Let us consider an example for symmetry. A common denotation for the symmetry of a relation `MARRIEDWITH` (such as used for “William is married with Susan”) in first-order predicate logic boils down to:

$$(1) \forall X, Y \text{ MARRIEDWITH}(X, Y) \leftarrow \text{MARRIEDWITH}(Y, X).$$

In F-Logic, this would be a valid axiom specification, too. Most often, however, modelers that use F-Logic take advantage of the object-oriented syntax. Concept definitions in F-Logic for *Person* having an attribute `MARRIEDWITH` and *Man* being a subconcept of *Person* is given in (2), while a fact that William is a *Man* who is `MARRIEDWITH` Susan appears like in (3).

(2) $Person[MARRIEDWITH \Rightarrow Person]$.
 $Man:Person$.

(3) $William:Man[MARRIEDWITH \rightarrow Susan]$.

Hence, a rule corresponding to (1) is given by (4).

(4) $\forall X, Y \ Y[MARRIEDWITH \rightarrow X] \leftarrow X[MARRIEDWITH \rightarrow Y]$.

We denote symmetry as a predicate that holds for particular relations:

(5) $SYMMETRIC(MARRIEDWITH)$.

In RDF(S), this specification may easily be realized by a newly agreed upon class `o:Symmetric`:

(6) `<o:Symmetric rdf:ID="marriedWith"/>`

For a particular language like F-Logic, one may then derive the implications of symmetry by a general rule and, thus, ground the meaning of the predicate $SYMMETRIC$ in a particular target system. The corresponding transformation rule (here in F-Logic) states that if for all symmetric relations R and object instances X and Y it holds that X is related to Y via R , then Y is also related to X via R .

(7) $\forall R, X, Y \ Y[R \rightarrow X] \leftarrow SYMMETRIC(R) \text{ and } X[R \rightarrow Y]$.

This small example already shows three advantages:

1. The axiom specification (6) is rather target-system independent.
2. It is easily realizable in RDF(S).
3. Our approach for denoting symmetry is much sparser than its initial counterpart (4), because (7) is implicitly assumed as the agreed semantics for our schema definition.

The latter point deserves some more attention: Obviously, the agreement cannot directly be described in RDF(S). Thus, it is subject to specifications of RDF(S) terminology *outside* of the RDF(S) documents, which need to be agreed upon by a community of users. This might be considered a disadvantage, however, this is eventually the case for RDF, RDFS, and all of their extensions like OIL or DAML+OIL.

Following our strategy sketched in the previous subsection, these steps from RDF representation to axiom meaning are now summarized in Table 1. For easier understanding, we will reuse this table layout also in the following subsection.

| | | |
|---|--|------------------------|
| A | <code><o:Symmetric rdf:ID="marriedWith"/></code> | RDF(S) |
| B | $SYMMETRIC(MARRIEDWITH)$ | F-Logic Representation |
| C | $\forall R, X, Y \ Y[R \rightarrow X] \leftarrow SYMMETRIC(R)$ $\text{and } X[R \rightarrow Y]$. | Translation Axiom |
| D | $\forall X, Y \ X[MARRIEDWITH \rightarrow Y] \leftarrow$ $Y[MARRIEDWITH \rightarrow X]$. | Target Axiom |

Table 1: Symmetry

| | |
|---|---|
| A | <pre> <o:Composition rdf:ID="FatherInLawComp"> <o:composee rdf:Resource="fatherInLawOf"/> <o:firstComponent rdf:Resource="fatherOf"/> <o:secondComponent rdf:Resource="marriedWith"/> </o:Composition> </pre> |
| B | COMPOSITION(FATHERINLAWOF, FATHEROF, MARRIEDWITH) |
| C | $\forall R, Q, S, X, Y, Z \quad X[S \rightarrow Z] \leftarrow$ $\text{COMPOSITION}(S, R, Q) \text{ and } X[R \rightarrow Y] \text{ and } Y[Q \rightarrow Z].$ |
| D | $\forall X, Y, Z \quad X[\text{FATHERINLAWOF} \rightarrow Z] \leftarrow$ $X[\text{FATHEROF} \rightarrow Y] \text{ and } Y[\text{MARRIEDWITH} \rightarrow Z].$ |

Table 2: Composition

3.3 Composition of relations

The next example concerns composition of relations. For instance, if a first person is FATHEROF a second person who is MARRIEDWITH a third person then one may assert that the first person is the FATHERINLAWOF the third person. Again different inferencing systems may require completely different realizations of such an implication. The object description of such an axiom may easily be denoted in F-Logic or in RDF(S) (cf. Table 2). The transformation rule works very similarly as the transformation rule for symmetry.

3.4 Locally inverse relations

In our practice of implementing several knowledge portals⁸ we found that the inferences brought about by categories 1 to 5 were extremely useful. However, we also felt that these categories were often too general to be applied directly or when they were applied they easily yielded overly generic results.

For instance, given a conceptual model with concepts *WeddingParty*, *Person*, *PartyService* and properties HASMEMBER, PARTYAT, and SERVE. The definition of inverses is desirable. However, defining that the inverse of PARTYAT is HASMEMBER and that the inverse of SERVE is also HASMEMBER leads to undesired consequences, viz. it entails that for every person who does PARTYAT a particular wedding party there is a — correct — relation, HASMEMBER, from the party to this person and an — incorrect — relation, SERVE, from that person to that party. Thus, without intricate changes to the conceptual model one would have to live without the definition of inverses.

An easily viable way around the problem is the definition of *local inverseness* that also consider the domain of a relation⁹, before asserting an inverse relationship. Thus, (8) asserts that the inverse of HASMEMBER is PARTYAT, only if the corresponding domain is restricted to *WeddingParty*.

(8) LOCALINVERSE(*WeddingParty*, HASMEMBER, *Person*, PARTYAT)

Table 3 captures the corresponding translation and target axioms.

⁸Cf. (Staab et al., 2000; Staab & Maedche, 2001; Staab et al., 2001) and <http://ontobroker.aifb.uni-karlsruhe.de/demos.html>, <http://ontobroker.semanticweb.org/>.

⁹Actually, we have extended the model to consider either restrictions of the domain of a property *or* of its range *or* of its domain and its range.

cessed using ontology structures for easier discovery of resources as well as for easier development and maintenance of the portal.

Figure 4 depicts the overall framework. Based on the community ontology, actual facts are contributed by the knowledge acquisition community or crawled from their web sites, e.g. annotated web pages or RDF sources. The SiLRi reasoning engine (Decker et al., 1998) provides the ontology service and allows for querying the accumulated knowledge base. Thus it acts as the back end for the actual portal. Users may either query the inference engine by clicking together a query or they may use forms of the web portal to retrieve knowledge.

Semantic inferences are crucial for the service that is provided by the KA2Portal. So far, people could contribute semantic information, e.g. by providing RDF sources. However, it was so far impossible to provide semantics about these RDF sources beyond the means that are inherent in core RDF(S). Conversely, the ontology was described in F-Logic (Kifer et al., 1995). Therefore, people could look at the KA2 ontology, but the ontology was initially not built with the spirit of the Semantic Web, as it was not transparent to software agents on the Web and not reusable by them.

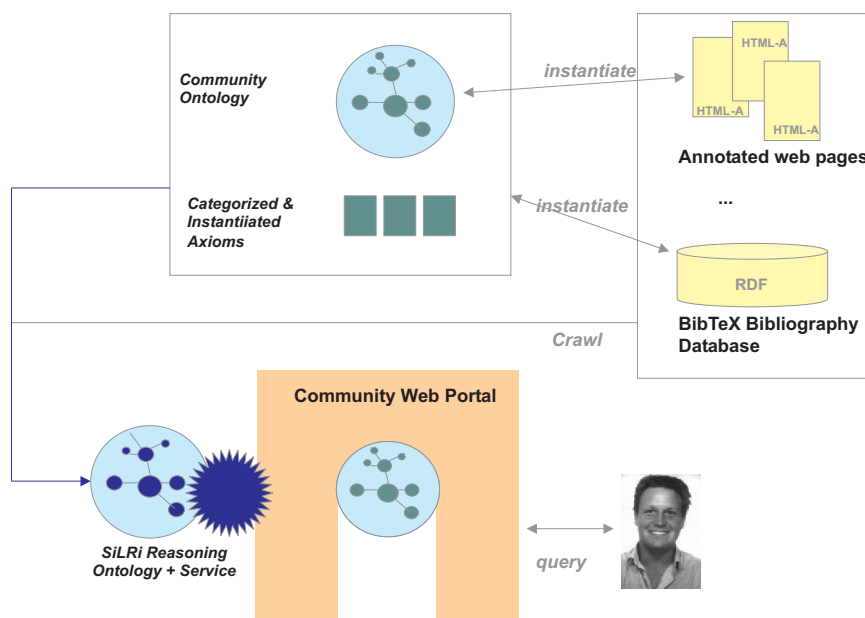


Figure 4: KA2Portal

Hence, there came up the need for representing the whole ontology, including axioms, on the Web. Thereby we felt two needs. First we did not want to come up with our own idiosyncratic syntax, but rather wanted to adhere to RDF(S) mechanisms as elaborated above. Second, we wanted to provide some tool support for engineering axioms. The framework that we have shown above allowed us both. We will show some excerpt of its application in the KA2Portal in the following.

4.2 Modeling the core ontology

We use our Ontology Engineering Environment OntoEdit for engineering class and property definitions in RDF(S) with graphical means. In particular, we may express that (cf. Figure 5 left and upper right window)

- *Book*, *Journal*, and *Article* are all subclasses of *Publication*
- A *Book* CONTAINSARTICLE *Article*
- A *Journal* CONTAINSARTICLE *Article*
- An *Article* is INBOOK *Book*
- An *Article* is INJOURNAL *Journal*

This is specified in RDF(S) as follows:

```

<rdfs:Class rdf:ID="Publication"/>
<rdfs:Class rdf:ID="Article">
  <rdfs:subClassOf rdf:resource="Publication"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Book">
  <rdfs:subClassOf rdf:resource="Publication"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Journal">
  <rdfs:subClassOf rdf:resource="Publication"/>
</rdfs:Class>

<rdf:Property rdf:ID="containsArticle">
  <rdfs:domain rdf:resource="Publication"/>
  <rdfs:range rdf:resource="Article"/>
</rdf:Property>
<rdf:Property rdf:ID="inBook"/>
  <rdfs:domain rdf:resource="Article"/>
  <rdfs:range rdf:resource="Book"/>
</rdf:Property>
<rdf:Property rdf:ID="inJournal"/>
  <rdfs:domain rdf:resource="Article"/>
  <rdfs:range rdf:resource="Journal"/>
</rdf:Property>

```

The ontology defines the conceptual backbone for contributing RDF metadata, e.g. bibliography entries:

```

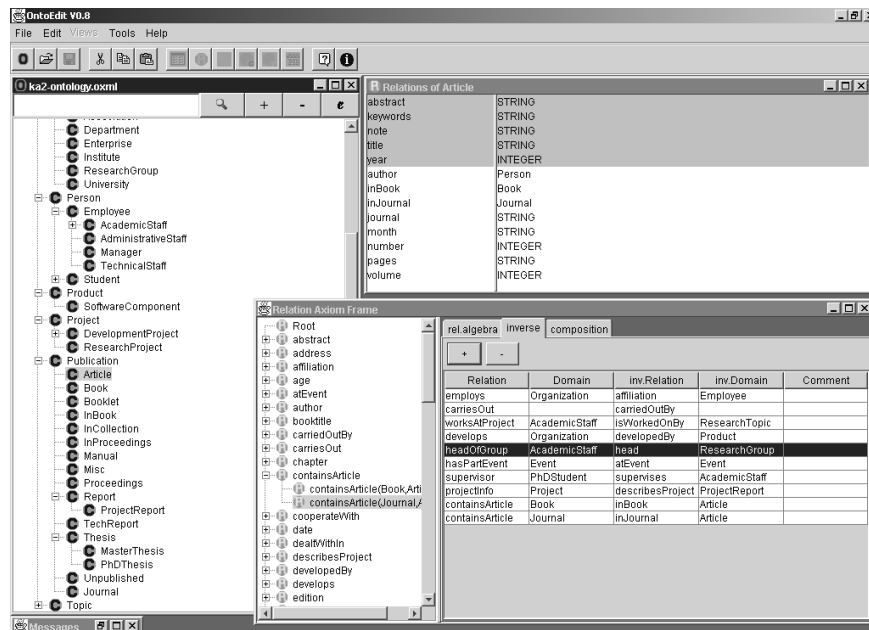
<ka2:Book rdf:ID="book:WeavingTheWeb">
  <ka2:author rdf:resource="http://w3c.org/person/tbl"/>
  <ka2:title>Weaving The Web</ka2:title>
....
</ka2:Book>

```

4.3 Modeling an ontology with axioms in RDF(S)

The axioms complete the core ontology. For KA2Portal we use locally inverse relations in order to express that:

- If a particular book CONTAINSARTICLE a particular article, this article is related via INBOOK to that book.
- If a particular article appears INBOOK in a particular book, this book is related via CONTAINSARTICLE that article.
- If a particular journal CONTAINSARTICLE a particular article, this article is related via INJOURNAL to that book.
- If a particular article appears INJOURNAL in a particular journal, this journal is related via CONTAINSARTICLE that article.



Left: KA2 Taxonomy; Right: Interface for Inverseness and Local Inverseness of Relations **explanation!!**

Figure 5: Snapshot of OntoEdit Web Ontology Workbench

Specified in our RDF(S) style denotation of axioms this boils down to:

```
<o:LocalInverse>
  <o:firstDomain rdf:Resource="Book"/>
  <o:firstRelation rdf:Resource="containsArticle"/>
  <o:secondDomain rdf:Resource="Article"/>
  <o:secondRelation rdf:Resource="inBook"/>
</o:LocalInverse>
<o:LocalInverse>
  <o:firstDomain rdf:Resource="Journal"/>
  <o:firstRelation rdf:Resource="containsArticle"/>
  <o:secondDomain rdf:Resource="Article"/>
  <o:secondRelation rdf:Resource="inJournal"/>
</o:LocalInverse>
```

OntoEdit allows to provide the components, `o:firstDomain`, `o:firstRelation`, `o:secondDomain`, `o:secondRelation`, in a table (cf. last two lines in table of right lower window in Figure 5). If `o:firstDomain` and `o:secondDomain` are provided, OntoEdit outputs the specification for locally inverse relations. If only `o:firstRelation` and `o:secondRelation` are provided, they are assumed to be immediate inverses.

5 Related Work

The proposal described in this paper is based on several related approaches, viz. we have built on considerations made for the RDF inference service SiLRi (Decker et al., 1998), the ontology engineering environments ODE (Blázquez et al., 1998) and Protégé (Grosso et al., 1999), the ontology interchange language OIL (Horrocks et al., 2000), considerations made by

Gruber (Gruber, 1993), and our own earlier work on general ontology engineering (Maedche et al., 2000; Staab & Maedche, 2000).

A purpose similar to our general goal of representing ontologies in RDF(S) is pursued with OIL (Horrocks et al., 2000). Actually, OIL might be considered a very sophisticated instantiation of our approach, as the definition of concepts and relations in description logics is equivalent to the instantiation of a small number of axiom schemata in a particular logical framework (cf. (Brachman, 1979)).

There are a number of other approaches for ontology exchange and representation in XML formats that we do not want to elaborate here, as they did not intend to support the RDF(S) metadata standard, which is one of our primary concerns (e.g. (Marchiori & Saarela, 1998; Karp et al., 1999; Heflin & Hendler, 2000)).

Concerning inferencing rather than representation, SiLRi (Decker et al., 1998) was one of the first approaches to provide inferencing facilities for RDF. It delivers most of the basic inferencing functions one wants to have in RDF and, hence, has provided a good start for many RDF applications. In fact, it even allows to use axioms, but these axioms may not be denoted in RDF, but only directly in F-Logic. It lacks capabilities for axiom representation in RDF(S) that our proposal provides.

Concerning engineering, we have discussed how to push the engineering of ontological axioms from the *symbol level* onto the *knowledge level* in our earlier proposals (Maedche et al., 2000; Staab & Maedche, 2000). There we follow and extend the general arguments made for ODE (Blázquez et al., 1998) and Ontolingua (Fikes et al., 1997). This strategy has helped us here in providing an RDF(S) object representation for a number of different axiom types. Nearest to our actual RDF(S)-based ontology engineering tool is Protégé (Grosso et al., 1999), which provides comprehensive and sophisticated support for editing RDFS and RDF. Nevertheless, Protégé currently lacks any support for axiom modeling and inferencing — though our approach may be very easy to transfer to Protégé, too.

6 Discussion

We have presented a new approach towards engineering ontologies in RDF and RDFS. Our objectives aim at the usage of existing inferencing services such as provided by deductive database mechanisms (Decker et al., 1998) or description logics systems (Horrocks, 1998). We reach these objectives through a *methodology* that classifies axioms into axiom types according to their *semantic meaning*. Each type receives an object representation that abstracts from scoping issues and is easily representable in RDF(S). Axiom descriptions only keep references to concepts and relations necessary to distinguish one particular axiom of one type from another one of the same type.

Our proposed extension of RDF(S) has been made with a clear goal in mind — the complete retention of the expressibility and semantics of RDF(S) for the representation of ontologies. This includes the relationship between ontologies and instances, both represented in RDF(S). Especially, the notion of *consistency* (cf. (Brickley & Guha, 1999)) between an RDF model and a schema also holds for ontologies expressed in RDF(S). The integration of the newly defined resources has been carried out in a such a way that all RDF processors capable of processing RDF schemas can correctly interpret RDF models following the ontology schema, even if they

do not *understand* the semantics of the resources in the σ -namespace.

Special applications like OntoEdit (Maedche et al., 2000) can interpret the σ -namespace correctly and thus fully benefit from the richer modelling primitives, if the RDF model is valid¹⁰ according to the defined ontology schema.

Acknowledgements. The research presented in this paper has been partially funded by BMBF under grant number 01IN802 (project “GETESS”) and by EU in the IST-1999-10132 project On-To-Knowledge. We thank our student Dirk Wenke who implemented large parts of the RDF(S)-based ontology editor and our colleague Stefan Decker who first proposed RDF(S) for specification of ontologies.

References

- Blázquez, M., Fernández, M., García-Pinar, J. M., & Gómez-Pérez, A. (1998). Building ontologies at the knowledge level using the ontology design environment. In *Proc. of the 11th Int. Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, Banff, Canada, October 1998.
- Brachman, R. (1979). On the epistemological status of semantic networks. *Associative Networks*, pages 3–50.
- Brickley, D. & Guha, R. (1999). Resource description framework (RDF) schema specification. Technical report, W3C. W3C Proposed Recommendation. <http://www.w3.org/TR/PR-rdf-schema/>.
- Decker, S. (1998). On domain-specific declarative knowledge representation and database languages. In *Proc. of the 5th Knowledge Representation meets Databases Workshop (KRDB98)*, pages 9.1–9.7.
- Decker, S., Brickley, D., Saarela, J., & Angele, J. (1998). A query and inference service for RDF. In *QL'98 - The Query Languages Workshop*. W3C. <http://www.w3.org/TandS/QL/QL98/>.
- Fikes, R., Farquhar, A., & Rice, J. (1997). Tools for assembling modular ontologies in Ontolingua. In *Proc. of AAAI 97*, pages 436–441.
- Grosso, E., Eriksson, H., Ferguson, R. W., Tu, S. W., & Musen, M. M. (1999). Knowledge modeling at the millennium — the design and evolution of Protégé-2000. In *Proc. of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Canada, October 1999.
- Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(1):199–220.
- Heflin, J. & Hendler, J. (2000). Dynamic Ontologies on the Web. In *Proceedings of American Association for Artificial Intelligence Conference (AAAI-2000)*. Menlo Park, California, AAAI Press, pages 443–449.
- Horrocks, I. (1998). Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647.
- Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., Harmelen, F. V., Klein, M., Staab, S., & Studer, R. (2000). The ontology interchange language oil: The grease between ontologies. Technical report, Dep. of Computer Science, Univ. of Manchester, UK/ Vrije Universiteit Amsterdam, NL/ Administrator, Nederland B.V./ AIFB, Univ. of Karlsruhe, DE. <http://www.cs.vu.nl/~dieter/oil/>.

¹⁰cf. the “Validator” section in <http://www.ics.forth.gr/proj/isst/RDF/> for a set of operations to check for validity

- Karp, P. D., Chaudhri, V. K., & Thomere, J. (1999). XOL: An XML-based ontology exchange language. Technical report. Version 0.3.
- Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843.
- Lassila, O. & Swick, R. R. (1999). Resource description framework (RDF) model and syntax specification. Technical report, W3C. W3C Recommendation. <http://www.w3.org/TR/REC-rdf-syntax>.
- Maedche, A., Schnurr, H.-P., Staab, S., & Studer, R. (2000). Representation language-neutral modeling of ontologies. In Frank, U. (Ed.), *Proceedings of the German Workshop "Modellierung-2000"*. Koblenz, Germany, April, 5-7, 2000. Fölbach-Verlag.
- Marchiori, M. & Saarela, J. (1998). Query + metadata + logic = metalog. In *QL'98 - The Query Languages Workshop*. W3C. <http://www.w3.org/TandS/QL/QL98/>.
- Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R., & Sure, Y. (2000). Semantic community web portals. In *WWW9 — Proceedings of the 9th International World Wide Web Conference, Amsterdam, The Netherlands, May, 15-19, 2000*. Elsevier.
- Staab, S. & Maedche, A. (2001). Knowledge portals — ontologies at work. *AI Magazine*, 21(2).
- Staab, S., Schnurr, H.-P., Studer, R., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1).
- Staab, S. & Maedche, A. (2000). Axioms are objects, too — ontology engineering beyond the modeling of concepts and relations. In Benjamins, V., Gomez-Perez, A., & Guarino, N. (Eds.), *Proceedings of the ECAI 2000 Workshop on Ontologies and Problem-Solving Methods. Berlin, August 21-22, 2000*.
- Weibel, S. & Miller, E. (1998). Dublin core metadata. Technical report. <http://purl.oclc.org/dc>.

A The RDF schema for categories of relationships

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
  <rdfs:Class ID="Relation">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  </rdfs:Class>

  <rdfs:Class ID="Asymmetric">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>

  <rdfs:Class ID="Reflexive">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>

  <rdfs:Class ID="Transitive">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>

  <rdfs:Class ID="Irreflexive">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>

  <rdfs:Class ID="Symmetric">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>

  <rdfs:Class ID="PartOfRel">
    <rdfs:subClassOf rdf:resource="#Relation"/>
  </rdfs:Class>
</rdf:RDF>
```

```

</rdfs:Class>

<rdf:Description ID="isInverseRelationOf">
  <rdf:type rdf:resource="#Relation"/>
</rdf:Description>

<!-- Definitions for LOCALLY-INVERSE-RELATIONS -->

<rdfs:Class ID="LocalInverse"/>

<rdf:Property ID="firstDomain">
  <rdfs:domain rdf:resource="#LocalInverse"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Class"/>
</rdf:Property>

<rdf:Property ID="firstRelation">
  <rdfs:domain rdf:resource="#LocalInverse"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<rdf:Property ID="secondDomain">
  <rdfs:domain rdf:resource="#LocalInverse"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Class"/>
</rdf:Property>

<rdf:Property ID="secondRelation">
  <rdfs:domain rdf:resource="#LocalInverse"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<!-- Definitions for COMPOSITION -->

<rdfs:Class ID="Composition"/>

<rdf:Property ID="composee">
  <rdfs:domain rdf:resource="#Composition"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<rdf:Property ID="firstComponent">
  <rdfs:domain rdf:resource="#Composition"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<rdf:Property ID="secondComponent">
  <rdfs:domain rdf:resource="#Composition"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<!-- Definitions for PARTITION -->

<rdfs:Class ID="Partition"/>

<rdf:Property ID="partitionee">
  <rdfs:domain rdf:resource="#Partition"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<rdf:Property ID="parts">
  <rdfs:domain rdf:resource="#Partition"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"/>
</rdf:Property>

<!-- Definitions for General Axioms-->

<rdfs:Class ID="GeneralAxiom">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
</rdfs:Class>

<rdf:Property ID="lang">
  <rdfs:domain rdf:resource="GeneralAxiom"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string"/>
</rdf:Property>

<rdf:Property ID="text">
  <rdfs:domain rdf:resource="GeneralAxiom"/>
  <rdfs:range rdf:resource="http://www.w3.org/TR/xmlschema-2/#string"/>
</rdf:Property>

</rdf:RDF>

```