Ontological Modeling: Part 3

Terry Halpin
LogicBlox

This is the third in a series of articles on ontology-based approaches to modeling. The main focus is on popular ontology languages proposed for the Semantic Web, such as the Resource Description Framework (RDF), RDF Schema (RDFS), and the Web Ontology Language (OWL). OWL is based on description logic. A later series of articles will explore other logic-based languages such as Datalog. The first article [4] provided a simple introduction to ontologies and the Semantic Web, and covered most of the basic concepts in the Resource Description Framework (RDF), contrasting them with other data modeling approaches. The second article [5] discussed the N3 notation for RDF, and covered the basics of RDF Schema. This third article provides further coverage of RDFS, and introduces different flavors of the Web Ontology language (OWL).

Quick Review of Some RDF and RDFS Basics

Before discussing further aspects of RDFS [7], let's review some basic ideas covered in earlier articles [4, 5]. RDF statements are binary relationship instances represented as (subject, predicate, object) triples. Subjects and predicates are resources, and are identified by Uniform Resource Identifier (URI)[1] references, such as http://www.w3.org/TR/rdf-primer/#basicconcepts. Objects are either resources or literals (constants). Literals may be untyped (e.g. "Australia") or typed (e.g. 63: xsd;nonNegativeInteger).

To save writing and improve readability, a simplified notation for RDF known as Notation 3 (N3)[2] may be used [9]. In N3, you can predeclare namespace prefixes and then use those prefixes later as shorthand for full URIs when identifying resources. Identifiers using such namespace prefixes are called qualified names (qnames), and are formed by prepending the prefix to a colon ":" followed by the local identifier. A blank prefix is understood to mean the namespace is the current document, and we'll assume this is the case for the examples discussed in this article. N3 treats "a" as shorthand for "rdf:type" (i.e. is an instance of). For example, here are three RDF statements in N3 notation:

```
:Terry a  :Person.
:Terry :isFatherOf :Linda.
:Terry :isFatherOf :Selena.
```

N3 allows use of a semicolon to assume the same subject, and allows a comma to assume the same subject and predicate (hence commas may be used as a separator in an object list). So the above three statements about the subject :Terry may be declared more briefly thus:

```
:Terry a  :Person;
      :isFatherOf :Linda, :Selena.
```

RDFS builds on RDF by adding various features, such as formal support for classes and subclassing. Resources may be typed as instances of classes using rdfs:Class. For example, the statement :Person a rdfs:Class declares that Person is a class. In RDFS, use of the rdf:type (or N3 'a') predicate implies that the following object is a class. For example, the statement :USA a :Country asserts not only that the resource :USA is an instance of the resource :Country, but also that :Country is a class.

In RDFS, a statement of the form *A* rdfs:subClassOf *B* declares that *A* is a subclass of *B*, which also implies that both *A* and *B* are classes. The rdfs:subClassOf predicate is transitive, so given the statements :Girl rdfs:subClassOf :FemalePerson and :FemalePerson rdfs:subClassOf :Person we may infer that :Girl rdfs:subClassOf :Person.

---

[1] OWL 2 uses *IRIs* (Internationalized Resource Identifiers) [2] instead of URIs, to cater for non-English characters.

[2] Most of the N3 syntax has been incorporated into *Turtle* (Terse RDF Triple Language) [1], which is one of the surface syntaxes that may be used for OWL 2.

In RDFS, given any resources *a*, *B* and *C*, where *a* is an instance of *B*, and *B* is a subclass of *C*, we may infer that *a* is an instance of *C*. For example, given the statements :Terry a :MalePerson and :MalePerson rdfs:subClassOf :Person we may infer that :Terry a :Person.

RDFS allows the subject of a predicate to be restricted to a specified domain (or set of domains), and the object of a predicate to be restricted to a specified range (or set of ranges). For example, the fact type Person drives Car may be declared as follows:

:drives rdfs:domain :Person.
:drives rdfs:range :Car.

Predicate Subsetting in RDFS

As an additional feature, RDFS supports predicate subsetting via its predefined *rdfs:subPropertyOf* predicate. Unfortunately, RDF uses the term "property" to mean "binary predicate", so here "subPropertyOf" means something like "subPredicateOf". The syntax is:

predicate1 rdfs:subPropertyOf *predicate2*

As specified in the RDFS standard [7], the meaning is that each (subject, object) pair related by *predicate1* is also related by *predicate2*. So given any subject *a*, object *b*, and predicates *R* and *S*, and using "&" for "and", "$\subseteq$" for "rdfs:subPropertyOf", and "$\rightarrow$" for implies, the following implication holds:

$(a\ R\ b\ \ \&\ \ R \subseteq S)\ \rightarrow\ a\ S\ b$

For example, from the following two statements

:Terry :isFatherOf :Selena.
:isFatherOf rdfs:subPropertyOf :isaParentOf.

we may infer

:Terry :isaParentOf :Selena.

We can declare that the subject and object of each parenthood relationship must be a person as follows:

:isaParentOf rdfs:domain :Person.
:isaParentOf rdfs:range :Person.

Given these restrictions and the above subProperty declaration for fatherhood, it follows that the domain and range of the fatherhood predicate is also restricted to Person.

One way of modeling this in Object-Role Modeling (ORM) [3] and the Unified Modeling Language (UML) is to specify a *subset constraint* from the fatherhood association to the parenthood association, as shown in Figure 1. Each notation depicts the constraint using a dashed arrow from the subset association to the superset association, annotated by the subsetOf operator (ORM uses "$\subseteq$", and UML uses "{subset}"). With this approach, fatherhood and parenthood facts are asserted, and the subset constraint ensures that each ordered pair of persons in a fatherhood fact also appears in a parenthood fact.

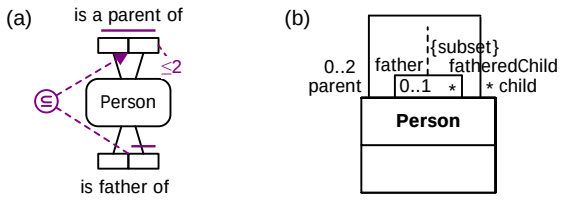

Figure 1 A subset constraint from fatherhood to parenthood in (a) ORM and (b) UML.

If the model includes information about a person's gender, the fatherhood predicate may instead be modeled as a derived fact type, as shown in Figure 2. The ORM schema includes derivation rules for the MalePerson subtype and fatherhood fact type, expressed in FORML 2. ORM appends an asterisk "*" to indicate derivation, whereas UML prepends a slash "/". The derivation rules are omitted from the UML class diagram, but could be added as notes or as expressions in the Object Constraint Language (OCL) [6].

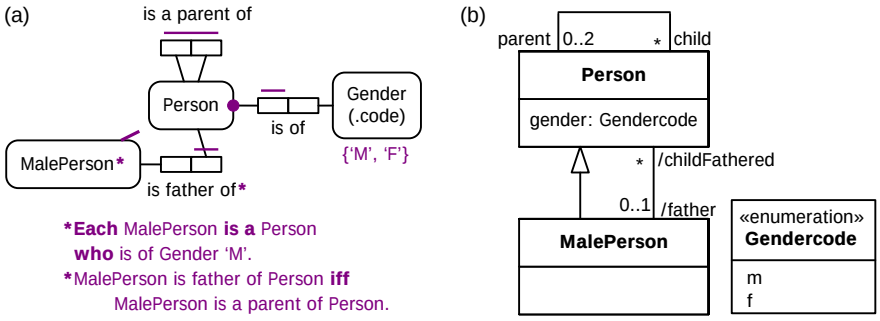

Figure 2 Fatherhood derived from parenthood and gender in (a) ORM and (b) UML.

The fatherOf predicate is a special case of the parentOf predicate, so the subset relationship between the two predicates is a logical necessity. A subset constraint may be also used for cases where the subset relationship is merely contingent on the business domain. For example, consider a world where people can drive only the cars that they own. This may be modeled in ORM and UML as shown in Figure 3. In RDFS, if we restrict the domain and range of each predicate to Person and Car, we can formulate this situation as follows:

:drives rdfs:domain :Person.
:drives rdfs:range :Car.
:drives rdfs:subPropertyOf :owns.

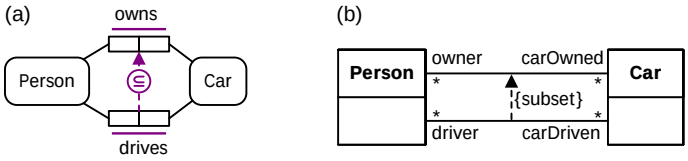

Figure 3. A subset constraint from car driving to car ownership in (a) ORM and (b) UML.

These examples illustrate the use of the subPropertyOf predicate for both predicate specialization (e.g. fatherhood as a special case of parenthood) as well as simple subsetting (driving is not a special case of ownership).

The subPropertyOf predicate is *transitive*. For example, from

:isFatherOf rdfs:subPropertyOf :isaParentOf.
:isaParentOf rdfs:subPropertyOf :isanAncestorOf.

we may infer

:isFatherOf rdfs:subPropertyOf :isanAncestorOf.

Further features of RDFS (e.g. support for set membership) may be found in its specification [7]. For information modeling purposes, RDFS is too inexpressive. For example, it cannot constraint relationships to be other than *m:n* (many-to-many). At the same time, RDFS is too "expressive" in its lack of restrictions. Like RDF, RDFS allows a type to be an instance of itself. Moreover, its underlying logic is undecidable—there are formulae expressible in its logic for which, even in principle, no algorithm can be constructed to determine whether those formulae are logical truths. Clearly, for realistic applications something better than RDFS is needed. For such reasons, the *Web Ontology Language (OWL)* was developed.

Overview of OWL and its Sublanguages

As in RDF and RDFS, predicates in OWL are restricted to binary relationships, so unary and *n*-ary ($n \geq 3$) relationships are not supported. OWL has a simpler formal semantics than RDFS, forbidding some RDFS features, while also supporting other features absent from RDFS, such as the ability to capture further constraints.

I use the term *model* to include both the schema (structure) and a population of instances. OWL refers to an ontology schema for a given business domain as a terminological box (*Tbox*), and a satisfying population of fact instances as an assertional box (*Abox*). While Tboxes (schemas) may be declared in OWL, populations of these schemas must instead be expressed in RDF/XML, or a language that can be transformed to RDF/XML. This situation is summarized in Figure 4.

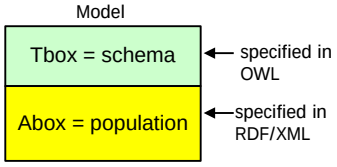

Figure 4. The Tbox is a schema, and an Abox is a population of assertions (statements) conforming to the schema.

Information on the Web is often incomplete. So OWL adopts the *Open World Assumption* (*OWA*). Hence the failure to find or infer a proposition (e.g. Humankind will survive after the year 2012) does not imply that the proposition is false.

OWL was endorsed as a World Wide Web Consortium (W3C) recommendation in February 2004 [8]. In October 2009, a major upgrade to the language, known as *OWL 2*, was recommended by the W3C [11], and the previous version was renamed *OWL 1*. OWL 1 has three flavors: OWL Lite, OWL DL, and OWL Full. As shown in Figure 5(a), OWL Lite is a subset of OWL DL, which itself is a subset of OWL Full.

OWL Lite is used for basic classification hierarchies where all the constraints are simple. It is based on the description logic *SHIF(D)*, has a low formal complexity, and is decidable. OWL Lite can express only simple constraints, and supports only simple inferences (e.g. those based on transitivity). *OWL DL* is much more expressive while still being decidable, and is based on the description logic *SHOIN(D)*. OWL DL includes all the constructs of OWL Full but restricts their use. For example, OWL DL is a fragment of first order logic, so does not allow a class to be declared an instance of another class in the same model. *OWL Full* is more expressive, allowing all RDF expressions, including allowing a class to be an instance of itself. OWL Full is undecidable, going beyond even first-order logic. In spite of its undecidable nature, OWL Full still cannot express some important constraints that are expressible in ORM (e.g. acyclicity).

As shown in Figure 5(b), OWL 2 introduced three new sublanguages as subsets of OWL DL: OWL 2 EL, OWL 2 QL, and OWL 2 RL. *OWL 2 EL* is designed to provide good performance for basic reasoning tasks on large ontologies, and is based on the *EL* family of description logics that provide existential quantification but not universal quantification. *OWL 2 QL* is designed to provide good performance with basic queries on large ontologies, where those queries can be implemented by rewriting them into queries in a standard relational query language such as SQL. *OWL 2 RL* is designed to provide good performance for managing ontologies with rule languages such as rule-extended database technologies operating directly on RDF triples. Extensive details on OWL 2 may be found on the W3C Website [10, 12, 13, 14].

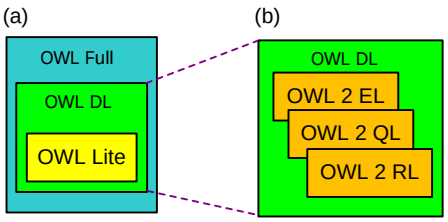

Figure 5. Three OWL 1 languages (a), and three new sublanguages of OWL DL added in OWL 2 (b).

Cardinality Restrictions in OWL 1

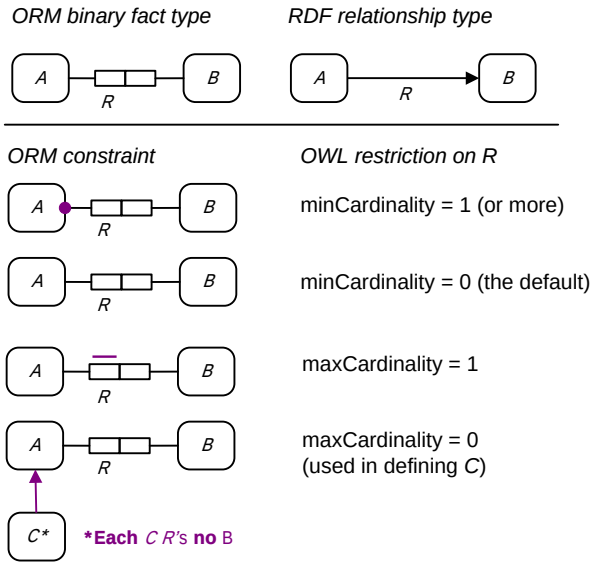

Figure 6. Some correspondences between ORM and cardinality restrictions in OWL.

By default, all relationships in OWL are optional and *m:n*. OWL Lite extends RDFS by enabling predicate roles to be declared mandatory, and predicates to be declared *n*:1, 1:*n*, or 1:1, using *owl:minCardinality and owl:maxCardinality restrictions*. Figure 6, adapted from [3], shows how these restrictions on a predicate *R* with a class domain *A* and a class range *B* translate into ORM.

Although Figure 6 depicts the RDF relationship type as an RDF graph, OWL has no graphic notation, so its statements must be represented textually. Restricting the predicate *R* with a minCardinality of 1 means that each instance of *A* is related via *R* to at least one instance of *B*. In ORM terms, the first role of *R* is mandatory (shown by the solid dot). A minCardinality of 0 is the default, and means the role played by *A* is optional. A maxCardinality of 1 means that each instance of *A* is related via *R* to at most one instance of *B*. In ORM terms, the first role of *R* has a simple uniqueness constraint (shown by a bar on the role). These minCardinality and maxCardinality settings are equivalent to the same UML minimum and maximum multiplicity settings on *B*'s role in the association (see [3], p. 360 for a detailed correspondence between ORM constraints and UML multiplicity constraints on binary associations).

Assigning a maxCardinality of 0 is somewhat different. This means that each instance of the relevant class (let's call it *C*) being discussed is related via *R* to *no* instance of *B*. In Figure 6, *C* is a subclass of *A*. As a concrete example, let *A R B* be the fact type Person drives Car, and let *C* be the subtype NonDriver, defined by the ORM derivation rule Each NonDriver is a Person who drives no Car.

As a shortcut when the minimum and maximum cardinalities are the same, a simple *owl:cardinality* restriction may be assigned. So setting owl:cardinality to 0 means forbidden, and setting owl:cardinality to 1 means exactly 1. For example, the OWL cardinality restriction that each company has exactly one fax number (depicted in ORM and UML in Figure 7) may be set out in RDF/XML as follows:

```
<owl:Restriction>
    <owl:onProperty rdf:resource = "#hasFaxNr"/>
    <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
```

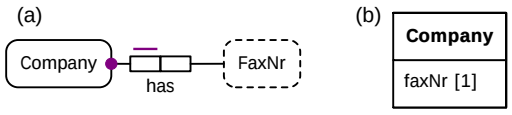

Figure 7. A mandatory *n*:1 relationship in ORM (a), and equivalent attribute in UML (b).

OWL DL and OWL Full allow a minCardinality setting of *n* greater than 1. In ORM this adds to the mandatory constraint on the left-hand role a frequency constraint of ≥*n* (where *n* > 1), as shown in Figure 8. In UML, this corresponds to a minimum multiplicity constraint of *n*..* on the right-hand role.

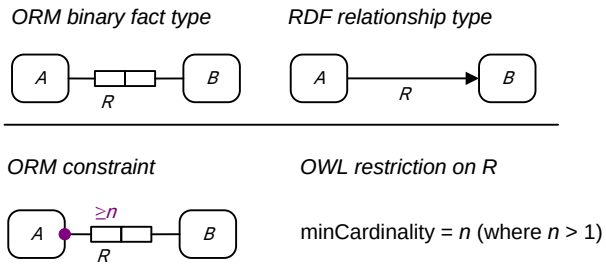

Figure 8. A mandatory and frequency constraint in ORM (a) and equivalent minCardinality restriction in OWL (b).

For example, to constrain each translator to speak at least two languages, in OWL we would restrict the speaks predicate with domain Translator and range Language to have minCardinality = 2. Figure 9 shows the ORM and UML schemas for this example.

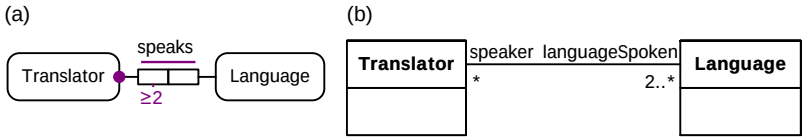

Figure 9. Applying an owl:minCardinality = 2 restriction to the speaks predicate in (a) ORM and (b) UML.

Conclusion

This article reviewed some basics of RDF and RDFS, discussed the use of rdfs:subPropertyOf for subsetting of predicates, provided an overview of OWL and its sublanguages, and then examined cardinality restrictions in OWL 1. The next few articles will consider OWL 1 and OWL 2 in some depth.

References

1. Beckett, D. & Berners-Lee, T. 2008, 'Turtle – Terse RDF Triple Language', W3C. URL: http://www.w3.org/TeamSubmission/turtle/#sec-diff-n3.
2. Duerst, M. & Suignard, M. 2005, 'RFC 3987: Internationalized Resource Identifiers (IRIs)', IETF, January 2005. URL: http://www.ietf.org/rfc/rfc3987.txt.
3. Halpin, T. & Morgan, T. 2008, *Information Modeling and Relational Databases, 2nd edition*, Morgan Kaufmann, San Francisco.
4. Halpin, T. 2009, 'Ontological Modeling: Part 1', *Business Rules Journal*, Vol. 10, No. 9 (Sep. 2009), URL: http://www.BRCommunity.com/a2009/b496.html.
5. Halpin, T. 2009, 'Ontological Modeling: Part 2', *Business Rules Journal*, Vol. 10, No. 12 (Dec. 2009), URL: http://www.BRCommunity.com/a2009/b513.html.
6. Warmer, J. & Kleppe, A. 2003, *The Object Constraint Language, Second Edition*, Addison-Wesley.
7. W3C 2004, 'RDF Vocabulary Description Language 1.0: RDF Schema', URL: http://www.w3.org/TR/rdf-schema/.
8. W3C 2004, 'OWL Web Ontology Language Overview', URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/.
9. W3C 2006, 'Notation 3 (N3): A Readable RDF Syntax', Ed. T. Berners-Lee, URL: http://www.w3.org/DesignIssues/Notation3.
10. W3C 2009, 'OWL 2 Web Ontology Language: New Features and Rationale', URL: http://www.w3.org/TR/owl2-new-features/.

11. W3C 2009, ‘OWL 2 Web Ontology Language: Document Overview’, URL: http://www.w3.org/TR/owl2-overview/.
12. W3C 2009, ‘OWL 2 Web Ontology Language: Primer’, URL: http://www.w3.org/TR/owl2-primer/.
13. W3C 2009, ‘OWL 2 Web Ontology Language: Profiles’, URL: http://www.w3.org/TR/owl2-profiles/.
14. W3C 2009, ‘OWL 2 Web Ontology Language: Quick Reference Guide’, URL: http://www.w3.org/TR/owl2-quick-reference/.