Ontological Modeling: Part 12

Terry Halpin
INTI International University and LogicBlox

This is the twelfth 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 [2] introduced ontologies and the Semantic Web, and covered basic concepts in the Resource Description Framework (RDF), contrasting them with other data modeling approaches. The second article [3] discussed the N3 notation for RDF, and covered the basics of RDF Schema. The third article [4] provided further coverage of RDFS, and introduced different flavors of the Web Ontology language (OWL). The fourth article [5] discussed basic features of OWL, mainly using Manchester syntax. The fifth article [6] discussed OWL taxonomy, comparison operators for classes, data types and predicates, and examined inverses, functional roles and keys in more depth. The sixth article [7] covered cardinality restrictions in OWL 2. The seventh article [8] discussed the union, intersection, and complement operators in OWL 2. The eighth article [9] explored support for ring constraints within OWL 2. The ninth article [10] discussed enumerated types as well as value restrictions on properties in OWL 2. The tenth article [11] examined OWL 2's support for property chains, and compared this with related concepts in data modeling approaches. The eleventh article [12] reviewed the logical status of structural statements in OWL 2, contrasting this with other data modeling approaches that support both integrity constraints and derivation rules. The current article discusses negated facts in OWL, and avoiding circularity when declaring subproperty chains.

Negated Facts in OWL

OWL adopts the *Open World Assumption*, allowing that our knowledge of the relevant business domain might be *incomplete*. Hence, if a proposition is neither asserted nor derived, it is treated as "possibly true" (we may not infer that it is false). For example, consider the data model shown in Figure 1, which is displayed in Object-Role Modeling (ORM) notation [6]. Here some facts are recorded about three famous scientists. The database is incomplete in the sense that we have not recorded where Einstein was born, and we have omitted some facts about further languages in which some of the scientists were fluent.

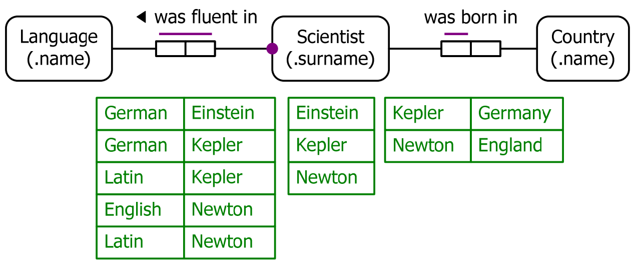


Figure 1 An ORM model storing incomplete knowledge about some scientists.

The model may be simply coded in OWL using techniques discussed in earlier articles. Table 1 shows one way to do this in both Manchester and Turtle syntax. For simplicity, we assume the name entries in Figure 1 may be used as Internationalized Resource Identifiers (IRIs).

Table 1 Coding the model in Figure 1 in OWL

Manchester Syntax	*Turtle Syntax*
DisjointClasses: Language, Scientist, Country	[] a owl:AllDisjointClasses ;
ObjectProperty: wasBornIn	owl:members (:Language :Scientist :Country).
Domain: Scientist	:wasBornIn a owl:ObjectProperty, owl:FunctionalProperty.
Range: Country	:wasBornIn rdfs:domain :Scientist ;
Characteristics: Functional	rdfs:range :Country.
ObjectProperty: wasFluentIn	:wasFluentIn rdfs:domain :Scientist ;
Domain: Scientist	rdfs:range :Language.
Range: Language	:Scientist rdfs:subClassOf
Class: Scientist	[] owl:Restriction; owl:onProperty :wasFluentIn;
SubClassOf: wasFluentIn min 1	owl:minCardinality 1.
Individual: Einstein	:Einstein :wasFluentIn :German.
Facts: wasFluentIn German	:Kepler :wasFluentIn :German, :Latin.
Individual: Kepler	:Kepler :wasBornIn :Germany.
Facts: wasFluentIn German,	:Newton :wasFluentIn :English, :Latin.
wasFluentIn Latin,	:Newton :wasBornIn :England.
wasBornIn Germany	
Individual: Newton	
Facts: wasFluentIn English,	
wasFluentIn Latin,	
wasBornIn England	

The incompleteness of the database could arise because the missing facts are unknown to the modeler, or simply because he/she has not yet completed entry of all the relevant known facts of interest. Regardless, it is impossible to determine from this OWL database the truth value of any of the following propositions.

(1) Einstein was born in Germany.
(2) Einstein was born in England.
(3) Einstein was fluent in Latin.
(4) Einstein was fluent in English.

In actual fact, (1), (3) and (4) are true and (2) is false, In fact, Einstein was also fluent in French and Italian. Now suppose that while we are still unsure where Einstein was born, we do know for a fact that he was not born in England, as indicated by (5) below.

(5) Einstein was not born in England.

Here proposition (5) is said to be the *negation* of proposition (2). In English, the logical operator "not" is often used to perform negation. Although knowing that Einstein was not born in England does not tell us where he was born, it is still informative to declare such a negated fact.

OWL enables you to explicitly assert the negation of an atomic proposition by declaring a *NegativePropertyAssertion*. Manchester Syntax uses the logical not operator before the relevant predicate object pair in the Facts slot for the individual. Turtle uses a lengthy owl:NegativePropertyAssertion syntax. For example, we may assert that Einstein was not born in England as shown in Table 2.

Table 2 Asserting that Einstein was not born in England

Manchester Syntax	*Turtle Syntax*
Individual: Einstein	[ ] a owl:NegativePropertyAssertion ;
Facts: not wasBornIn England	owl:sourceIndividual :Einstein ;
	owl:assertionProperty :wasBornIn ;
	owl:targetIndividual :England .

The example just considered negates an object property, since the target object of the triple is an entity. Data properties (where the target object is a literal) may be negated in a similar way, but using "targetValue" instead of "targetIndividual". For example, treating IQs as numbers we may assert that Einstein's IQ is not equal to 100 as shown in Table 3.

Table 3 Asserting that Einstein's IQ is not 100

Manchester Syntax	*Turtle Syntax*
DataProperty: hasIQ Domain: Scientist Range: xsd:nonNegativeInteger Characteristics: Functional Individual: Einstein Facts: not hasIQ 100	:hasIQ a owl:DatatypeProperty, owl:FunctionalProperty. :hasIQ rdfs:domain :Scientist ; rdfs:range : xsd:nonNegativeInteger. [ ] a owl:NegativePropertyAssertion ; owl:sourceIndividual :Einstein ; owl:assertionProperty :hasIQ ; owl:targetValue "100" ^^xsd:nonNegativeInteger .

Contrast this approach with typical database approaches, which often employ the closed world assumption, in which all relevant facts are assumed to be known, so that a fact that is neither asserted nor derived is assumed to be false. For example, in a relational database, if a language fluency table is populated with the fluency data in Figure 1 one would typically assume that Einstein is not fluent in any language other than German. In contrast, the default stance in OWL is that "anything is possible until you say otherwise" [6, p. 17].

Avoiding Circularity with SubProperty Chains

As discussed in a previous article [11], a *property chain* in OWL is an expression that composes two or more properties in a chain of subject-predicate-object facts in which the grammatical object of each fact (other than the last) is also the grammatical subject of the following fact. A named relationship between the first subject and the last object in the chain may be inferred by declaring a *subproperty* condition on that chain. For example, grandparenthood may be inferred from a chain of two parenthood relationships as shown in Table 4. Manchester syntax uses "o" as the composition operator.

Table 4 Inferring grandparenthood from a parenthood chain in OWL

Manchester Syntax	*Turtle Syntax*
ObjectProperty: isaGrandparentOf SubPropertyChain: isaParentOf o isaParentOf	:isaGrandparentOf owl:propertyChainAxiom (:isaParentOf :isaParentOf).

An ORM schema for this situation is shown in Figure 2. Here grandparenthood is specified as a semiderived fact type using a derivation rule whose main operator is "if". The semiderived nature (indicated by "$^+$") allows some instances of grandparenthood to be derived if the relevant parenthood facts are known, while other grandparenthood instances may be simply asserted (e.g. when the relevant parenthood facts are unknown).

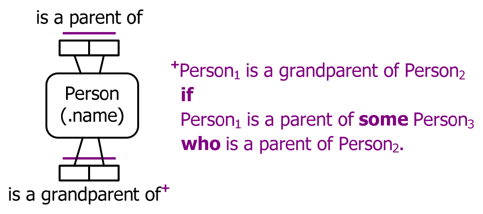

is a parent of
Person
(.name)
is a grandparent of+
+Person1 is a grandparent of Person2
if
Person1 is a parent of some Person3
who is a parent of Person2.

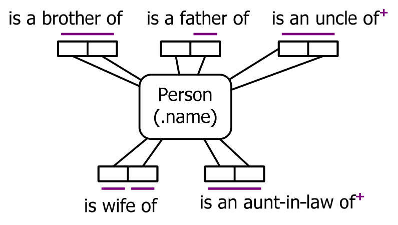
Figure 2 An ORM schema with grandparenthood modeled as a semiderived fact type.

In classical logic, this conditional rule may be specified using either of the following equivalent formulations. Here "$\forall$", "$\exists$", "$\leftarrow$" and "&" may respectively be read "For each", "there exists", "if" and "and".

$$\forall x,y\ [x \text{ is a grandparent of } y \ \leftarrow\ \exists z(x \text{ is a parent of } z\ \&\ z \text{ is a parent of } y)].$$
$$\forall x,y,z\ [x \text{ is a grandparent of } y \ \leftarrow\ (x \text{ is a parent of } z\ \&\ z \text{ is a parent of } y)].$$

Notice that the conditional goes one way only. We use $\leftarrow$ (if), not $\equiv$ (if and only if). This allows us to assert some grandparenthood facts without knowing relevant parenthood facts that could be used to derive them. In this way, classical logic caters for incomplete information in a similar way to OWL.

In dealing with property chain axioms, OWL includes formal restrictions to avoid circular reasoning that could lead to problems of decidability. We now illustrate this issue by adapting some examples provided in the OWL 2 specification [17, pp. 116-118]. Figure 3 shows an ORM model to semiderive unclehood and aunt-in-law relationships. Note that the aunt-in-law derivation rule makes use of the unclehood predicate which itself is semiderived. This is not a problem. Indeed, in many deductive systems, including datalog, it is quite common for one derivation rule to call predicates that are derived in other rules. However, we do need to be careful that the calls are not circular. In this example, the unclehood rule makes use of the asserted brotherhood and fatherhood predicates, and the aunt-in-law rule makes use of the asserted wife and semiderived unclehood predicates, so no cycles are involved.



+Person$_1$ is an uncle of Person$_2$
if
Person$_1$ is a brother of **some** Person$_3$
who is a father of Person$_2$.

+Person$_1$ is an aunt-in-law of Person$_2$
if
Person$_1$ is wife of **some** Person$_3$
who is an uncle of Person$_2$.

Figure 3 An ORM model to semiderive unclehood and aunt-in-law relationships.

Table 5 shows the OWL property chain axioms for the same situation in Manchester and Turtle syntax. Including both rules in the same model is quite legal since no circular reasoning is involved. The corresponding formulae in classical logic are set out below.

$$\forall x,y,z\ [x \text{ is an uncle of } y \ \leftarrow\ (x \text{ is a brother of } z\ \&\ z \text{ is a father of } y)].$$
$$\forall x,y,z\ [x \text{ is an aunt-in-law of } y \ \leftarrow\ (x \text{ is wife of } z\ \&\ z \text{ is an uncle of } y)].$$

Table 5 Inferring unclehood and aunt-in-law properties in OWL

Manchester Syntax	*Turtle Syntax*
ObjectProperty: isanUncleOf SubPropertyChain: isaBrotherOf o isaFatherOf	:isanUncleOf owl:propertyChainAxiom (:isaBrotherOf :isaFatherOf).
ObjectProperty: isanAuntInLawOf SubPropertyChain: isWifeOf o isanUncleOf	:isanAuntInLawOf owl:propertyChainAxiom (:isWifeOf :isanUncleOf).

As an example that does cause problems, consider the ORM model in Figure 4. Here unclehood is semiderived from brotherhood and fatherhood as before, except that brotherhood itself is now semiderived. Moreover, the derivation rule for brotherhood calls the unclehood predicate, which itself depends on brotherhood. So the derivation rules in combination involve a vicious circle.

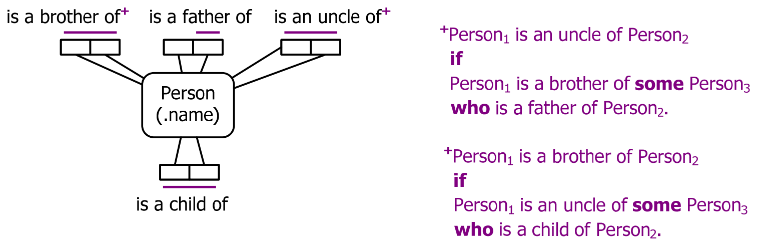

is a brother of+
is a father of
is an uncle of+
Person
(.name)
is a child of
+Person1 is an uncle of Person2
if
Person1 is a brother of some Person3
who is a father of Person2.
+Person1 is a brother of Person2
if
Person1 is an uncle of some Person3
who is a child of Person2.

Figure 4 An inconsistent ORM model with circular derivation rules.

Table 6 shows the illegal property chain combination in OWL. The corresponding rules in classical logic are set out below.

$\forall x,y,z$ [x is an uncle of y $\leftarrow$ (x is a brother of z & z is a father of y)].
$\forall x,y,z$ [x is a brother of y $\leftarrow$ (x is an uncle of z & z is a child of y)].

Table 6 An illegal combination of OWL rules involving circularity

Manchester Syntax	*Turtle Syntax*
ObjectProperty: isanUncleOf SubPropertyChain: isaBrotherOf o isaFatherOf	:isanUncleOf owl:propertyChainAxiom (:isaBrotherOf :isaFatherOf).
ObjectProperty: isaBrotherOf SubPropertyChain: isanUncleOf o isaChildOf	:isaBrotherOf owl:propertyChainAxiom (:isanUncleOf :isaChildOf).

Note that derivation rules may involve recursion without invoking circular reasoning. For example, the derivation rule in Figure 5 is recursive, since the childhood predicate appears in both the head and the body of the rule. However, this does not lead to circularity. Another classical example of a non-circular, recursive rule is the derivation of ancestorhood from parenthood: $\forall x,y,z$ [x is an ancestor of y $\leftarrow$ (x is a parent of y $\vee$ (x is a parent of z & z is an ancestor of y))].

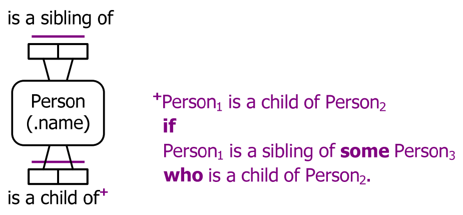

is a sibling of
Person
(.name)
is a child of+
+Person1 is a child of Person2
if
Person1 is a sibling of some Person3
who is a child of Person2.

Figure 5 An ORM model with a recursive, but non-circular derivation rule.

Conclusion

The current article discussed how to specify negated facts in OWL 2, and illustrated the need to avoid circular reasoning when using property chain axioms. As this aspect of subproperty chains is arguably most easily understood in terms of semiderivation, our data model examples in this article focused on ORM rather than UML [13]. A basic discussion of property chains that includes UML examples may be found in [11]. The next article examines further advanced aspects of OWL.

References

1. Halpin, T. & Morgan, T. 2008, *Information Modeling and Relational Databases*, *2nd edition*, Morgan Kaufmann, San Francisco.
2. Halpin, T. 2009, ‘Ontological Modeling: Part 1’, *Business Rules Journal*, Vol. 10, No. 9 (Sep. 2009), URL: http://www.BRCommunity.com/a2009/b496.html.
3. Halpin, T. 2009, ‘Ontological Modeling: Part 2’, *Business Rules Journal*, Vol. 10, No. 12 (Dec. 2009), URL: http://www.BRCommunity.com/a2009/b513.html.
4. Halpin, T. 2010, ‘Ontological Modeling: Part 3’, *Business Rules Journal*, Vol. 11, No. 3 (March 2010), URL: http://www.BRCommunity.com/a2010/b527.html.
5. Halpin, T. 2010, ‘Ontological Modeling: Part 4’, *Business Rules Journal*, Vol. 11, No. 6 (June 2010), URL: http://www.BRCommunity.com/a2010/b539.html.
6. Halpin, T. 2010, ‘Ontological Modeling: Part 5’, *Business Rules Journal*, Vol. 11, No. 12 (Dec. 2010), URL: http://www.BRCommunity.com/a2010/b570.html.
7. Halpin, T. 2011, ‘Ontological Modeling: Part 6’, *Business Rules Journal*, Vol. 12, No. 2 (Feb., 2011), URL: http://www.BRCommunity.com/a2011/b579.html.
8. Halpin, T. 2011, ‘Ontological Modeling: Part 7’, *Business Rules Journal*, Vol. 12, No. 6 (Jun., 2011), URL: http://www.BRCommunity.com/a2011/b602.html.
9. Halpin, T. 2011, ‘Ontological Modeling: Part 8’, *Business Rules Journal*, Vol. 12, No. 9 (Sep., 2011), URL: http://www.BRCommunity.com/a2011/b614.html.
10. Halpin, T. 2011, ‘Ontological Modeling: Part 9’, *Business Rules Journal*, Vol. 12, No. 12 (Dec., 2011), URL: http://www.BRCommunity.com/a2011/b629.html.
11. Halpin, T. 2012, ‘Ontological Modeling: Part 10’, *Business Rules Journal*, Vol. 13, No. 3 (Mar., 2012), URL: http://www.BRCommunity.com/a2012/b644.html.
12. Halpin, T. 2012, ‘Ontological Modeling: Part 11’, *Business Rules Journal*, Vol. 13, No. 6 (Jun., 2012), URL: http://www.BRCommunity.com/a2012/b657.html.
13. Object Management Group 2011, *OMG Unified Modeling Language Specification*, version 2.4.1. Available online at: http://www.omg.org/spec/UML/2.4.1/.
14. W3C 2009, ‘OWL 2 Web Ontology Language: Primer’, URL: http://www.w3.org/TR/owl2-primer/.
15. W3C 2009, ‘OWL 2 Web Ontology Language: Direct Semantics’, URL: http://www.w3.org/TR/owl2-direct-semantics/.
16. W3C 2009, ‘OWL 2 Web Ontology Language Manchester Syntax’, URL: http://www.w3.org/TR/owl2-manchester-syntax/.
17. W3C 2009, ‘OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax’, URL: http://www.w3.org/TR/owl2-syntax/.