

O-Telos-RDF: An Extension of RDF with Enhanced Meta-Modeling and Reification Functionalities

Wolfgang Nejdl, Hadhami Dhraief, Martin Wolpers
Computer Engineering – Knowledge Based Systems
University of Hannover
Appelstr. 4, 30167 Hannover, Germany
email: nejdl,dhraief,wolpers@kbs.uni-hannover.de

November 1, 2001

Abstract

In this paper we describe the formalization of an RDF(S) extension we call O-Telos-RDF, which provides enhanced functionalities for meta-modeling and reified statements. The axiomatic formalization is based very closely on the formalization of the modeling language O-Telos, which is based on a semantic network model quite similar to RDF(S), and has been used as a (meta) modeling language in various application areas during the last 10 years.

Keywords: RDF, semantic web, conceptual modeling, modeling language for the web, metamodeling and reification

1 Introduction

RDF(S) [4, 2] is based on the simple, yet quite powerful model of semantic networks, where every statement is expressed as a binary predicate on ground arguments, with nodes in the graph denoting arbitrary resources and literals, used as subjects and objects of statements, and arcs denoting specific relationships between two of these nodes. RDF(S) falls short however in exploiting both the simplicity and the expressivity of the underlying semantic network formalism. Meta-modeling and reification is cumbersome and restricted in RDF(S), even though it has been envisioned as one of the important application areas of RDF(S).

In an earlier paper [11], we have pointed out the dual use of properties like subclass and type both as primitive (metalevel) concepts to define the RDF(S) concept hierarchy as well as concepts defined in RDF(S) itself, and have proposed a meta-modeling approach separating these uses, which is more in line with a layered metamodeling approach such as described in [5]. More recently, [12] proposed a layered version of RDF(S) called RDFS(FA) motivated by the same observations. In this paper we explore another alternative by using the (meta-) modeling language O-Telos as a basis for an alternative resource description format we call O-Telos-RDF, which retains the principles of RDF(S), but extends its meta-modeling and reification functionalities. Primitive concepts like subclass and type are strictly axiomatized in this language, and can be used together with the other O-Telos-RDF concepts to formalize descriptions on all abstraction levels. The modeling language O-Telos has been defined and axiomatized in [7], based on its predecessor Telos [10, 9], and implemented in the deductive object oriented database system ConceptBase [6]. It has been used in a variety of modeling contexts (see [8] for an overview) during the last 10 years.

The extended functionalities of O-Telos-RDF are based on O-Telos' use of a semantic network (i.e. binary predicates) for modeling in a uniform way all kinds of information (information about objects and specific relationships between these objects, information about classes and relations, information about metaclasses, etc.)

We will use the paper by Conen and Klapsing [3] as a starting point for the description of our RDF(S) variant, O-Telos-RDF, and retain basically all original O-Telos axioms from [7], only modifying them when necessary to make them suitable for a resource description format comparable to RDF(S), and using RDF(S) terminology. Our order of presenting the different O-Telos-RDF concepts reflects the order of RDF(S) concept presentation in [3] which will make it easier for knowledgeable RDF(S) users to compare RDF(S) and the O-Telos-RDF variant.

2 O-Telos-RDF Concepts

O-Telos-RDF graphs are semantic networks, which consist of nodes and (directed) arcs. Nodes represent all kinds of concepts (such as classes, metaclasses etc.) and individuals (such as specific WWW pages and literals), arcs represent specific (property) relationships. Both nodes and arcs are labeled with atomic values or URIs as names. They are both first class entities, and are referenced by IDs (unique within the current namespace), usually in the form of URIs, which either contain the node or arc label (if it is an atomic value), or use the label directly as ID (when the labels are URLs or instances of primitive types like literals). By referencing these IDs, arcs can connect any two other entities (i.e. two nodes, one node and one arc or two arcs), in contrast to RDF(S), where arcs only connect resources (i.e. nodes). As in RDF(S) [3], all nodes have different labels, and represent different entities. Arc labels are not unique, two arcs between two specific entities have different labels, however.

2.1 Basic O-Telos-RDF Concepts and Predicates

2.1.1 Statement

All nodes and arcs in an O-Telos-RDF graph are represented by statements of the general form $s(sid, x, l, y)$ where sid represents the statement ID (a unique identifier of the statement), x and y represent identifiers of (possibly other) statements and l is called the label of the statement. All statement identifiers sid are URIs or as URI-like as possible and are unique globally (except when exactly the same statements are made in two different places). All statements in O-Telos-RDF are implicitly reified, and can be identified by their statement ID.

Axiom 2.1 *Statement identifiers uniquely identify statements:*

$$\begin{aligned} & \forall sid, x1, x2, l1, l2, y1, y2 \ s(sid, x1, l1, y1) \wedge s(sid, x2, l2, y2) \\ & \Rightarrow (x1 = x2) \wedge (y1 = y2) \wedge (l1 = l2) \end{aligned}$$

Example 1: This example gives a first impression of how the resulting statements look like. The exact definitions of the used constructs will follow in later sections. We want to express that a resource “LectureUnit1” has a property “title” with the value “Lecture Unit 1”. The XML-serialization, together with the statements look as follows:

```
<otelos:Description ID="LectureUnit1">
  <type s="otelos:individual"/>
  <title>Lecture Unit 1</title>
</otelos:Description>
```

```
s(sid1,sid1,LectureUnit1,sid1),s(sid2,sid1,title,"Lecture Unit 1")
s(sid3,sid1,type,otelos:individual),s(sid4,sid2,type,otelos:property)
with
sid1=ns:LectureUnit1, sid2=ns:LectureUnit1_title,
sid3=ns:LectureUnit_type_otelos:individual,
sid4=ns:LectureUnit1_title_type_otelos:property
```

and “ns:” stands for the current namespace specifying where these metadata can be found (e.g. “http://www.kbs.uni-hannover.de /ai1/metadata.html#”). The statement identifiers are generated automatically from the other arguments (we’ll discuss how later), so they do not have to be included in the XML serialization.

Tupels sid3 and sid4 specify membership of sid1 and sid2 in the (predefined) O-Telos-RDF classes otelos:individual and otelos:property respectively (similar to rdfs:class and rdf:property), and are themselves members of the O-Telos-RDF class otelos:type. As we will see in the following sections, membership in these predefined classes is specified by the syntactic form of the statements, and we will not include these type-statements in our later examples, as they can always be reconstructed from the axiomatic membership definition for these predefined classes.

Example 2: This example states that a web page with the URI “http://.../Defs.html” has a title called “Definitions”.

```
<otelos:Description about="http://.../Defs.html">
  <type s="otelos:individual"/>
  <title>Definitions</title>
</otelos:Description>
s(sid1,sid1,.../Defs.html,sid1), s(sid2,sid1,title,"Definitions")
with
sid1=http://.../Defs.html, sid2=http://.../Defs.html_title
```

Additionally, sid1 is instance of otelos:individual while sid2 is instance of otelos:property, and these two type-statements are instances of otelos:type.

Comparison to RDF(S) and O-Telos Axiom 2.1 corresponds to the O-Telos axiom 1 (uniqueness of object identifiers). O-Telos-RDF statement identifiers are constructed as URIs or URI-like identifiers in order to be able to reference them easily in other statements. In contrast to O-Telos object identifiers, they are not invisible. Their exact form will be discussed in the following sections. In contrast to RDF(S), there is no difference between “named” and “unnamed” resources, all statements have unique IDs. For Web-Pages, these are the usual URLs. Axiom 1 of [3] states that an RDF(S) statement connects two nodes with a label. The label has to be RFC 2396 [1] conform, statements do not have a unique ID. In O-Telos-RDF, labels can also be atomic values, while statement IDs are unique and usually conform to RFC 2396.

All statements in O-Telos-RDF are implicitly reified, and can be identified by their statement ID. Thus we don't have to reify a statement explicitly which contrasts with the axioms 7, 8 and 9 of [3]. Instead, when one statement talks about another statement, we have to enforce the existence of the statement talked about. This is basically the same as a foreign key constraint in relational database theory.

2.1.2 Individual

Nodes in an O-Telos-RDF graph are represented by statements of the form $s(o,o,o,o)$ or $s(ns:l,ns:l,l,ns:l)$, where o and $ns:l$ are URIs, ns is the current namespace and l is an atomic label. We call these statements individuals, they can be used as subjects and objects in statements.

Axiom 2.2 *The set of all statements, abbreviated as $otelos:statement$ is represented by the individual $s(otelos:statement,otelos:statement,statement,otelos:statement)$, where "otelos:" is the namespace for the O-Telos-RDF definitions.*

Axiom 2.3 *All statements are elements of the set $otelos:statement$.*

$$\forall sid, x, p, y \ s(sid, x, p, y) \Leftrightarrow type(sid, otelos : statement)$$

Axiom 2.4 *The set of all individuals is represented by $otelos:individual$ or (in longer form) $s(otelos:individual,otelos:individual,individual,otelos:individual)$.*

O-Telos-RDF individuals can represent both classes and instantiated objects (as well as metaclasses, metametaclasses, etc.), there is no syntactic distinction between these different abstraction levels. This makes unrestricted metamodelling hierarchies possible in O-Telos-RDF.

Axiom 2.5 *If the label of an individual is an atom, it is unique within its namespace. Together with its namespace, or if the label is already an URI, it is unique globally. Therefore the statement ID of an individual is unique globally in all cases.*

In the following, we will use the statement ID of an individual as an abbreviated name for that individual, i.e. we will be talking about $otelos:statement$, $otelos:individual$, etc. As the statement ID is unique and human readable, we can use this ID also to reference all other statements, which are not individuals.

Comparison to RDF(S) and O-Telos In O-Telos-RDF all individuals are resources in the usual RDF(S) sense, and can be used as subjects and objects in statements. O-Telos states the existence of statements in its axioms 18 and 24 which correspond to axiom 2.2 of O-Telos-RDF. Furthermore O-Telos-RDF states in axiom 2.3 that each statement is at least instance of $otelos:statement$ as does O-Telos in its axiom 23.

Axioms 1 and 2 of [3] state the existence of Resource which is equal to the O-Telos-RDF concept `otelos:individual` (see axiom 2.4), which corresponds to the O-Telos axioms 19 and 25.¹ Axiom 4 of [3] states that a named resource is identified by an URI. O-Telos-RDF broadens the scope of the identifiers with axiom 2.5 so that all individuals are identified by an URI, and further statements can be made about each individual using this URI. Labels of individuals are unique in O-Telos-RDF (corresponding to O-Telos axiom 2).

In contrast to RDF(S), O-Telos-RDF requires the declaration of each individual/resource, so each individual is represented by its corresponding tuple. Writing down these explicit individual declarations could be done by a smart parser which generates the corresponding tuples for each URL automatically. In the following, we will therefore use an XML serialization which does not require explicit individual declarations for URLs.

2.1.3 Class

In O-Telos-RDF, individuals represent both objects and classes. As all individuals can be instantiated, there is no need to introduce a separate class concept. All RDF(S) classes are represented as O-Telos-RDF individuals, as are their members. Class membership is defined using type-statements (see 2.2.1). To enhance readability, we define an individual `otelos:class`, which denotes the same set as `otelos:individual`, and use both `otelos:class` and `otelos:individual` in our XML serialization.

Comparison to RDF(S) and O-Telos Even though RDF(S) defines the concept `rdfs:Class`, no special axioms are defined in [3]. Rather, axioms are defined for the special properties related to `rdfs:Class`, namely `rdf:type`, `rdfs:subClassOf`, `rdfs:range` and `rdfs:domain`. This is similar to the formalization in O-Telos-RDF, which does not define the class concept at all, but has similar restrictions on the properties related to this concept.

2.1.4 Property

Arcs are represented by statements of the form $s(\text{sid}, x, p, y)$ or by the special case $s(\text{ns:p}, \text{otelos:statement}, p, y)$, with $\text{sid} \neq x$, $\text{sid} \neq y$, $\text{ns:p} \neq \text{otelos:statement}$ and $\text{ns:p} \neq y$, and p different from the three reserved labels `type`, `subClassOf` and `subPropertyOf`. We call these statements properties:

¹If we view something as a resource which can be referenced by an identifier, “resource” would correspond to “statement” in O-Telos-RDF, though.

Axiom 2.6 *Definition of properties:*

$$\forall sid, x, p, y \ s(sid, x, p, y) \wedge (sid \neq x) \wedge (sid \neq y) \wedge (p \neq subClassOf) \\ \wedge (p \neq subPropertyOf) \wedge (p \neq type) \Leftrightarrow type(sid, otelos : property)$$

Axiom 2.7 *The set of all properties is denoted by the statement (otelos:property,otelos:statement,property,otelos:statement), or otelos:property.*

Object Scoped Properties In the first case $s(sid,x,p,y)$, p is an atomic value, and is unique within the current namespace in conjunction with the source object x , which is the unique identifier of another statement. The sid is of the form x_p , which is unique in the current namespace, and potentially unique globally (except if another namespace specifies a property with the same label for the same x). This naming of the statement identifiers does not guarantee globally unique statement IDs, but is in line with the object scoped way of looking at properties. If the property $s(sid,x,p,y)$ is meant to represent the property definition for p , then (using RDF(S) terminology), x is the domain of p and y is the range of p . We then call p an *object scoped* property. This definition corresponds to the class-centric way of defining properties used in frame-based languages like O-Telos.

Axiom 2.8 *Names of “object scoped” properties are unique in conjunction with the source object:*

$$\forall sid1, sid2, x, p, y1, y2 \ s(sid1, x, p, y1) \wedge s(sid2, x, p, y2) \\ \Rightarrow (sid1 = sid2) \vee (p = type) \vee (p = subClassOf) \vee (p = subPropertyOf)$$

Example 3: The following description defines LectureUnit, which has a property named title of type literal.

```
<otelos:class ID="LectureUnit">
  <title s="otelos:literal">
</otelos:class>
s(sid1,sid1,LectureUnit,sid1), with sid1=ns:LectureUnit
s(sid2,sid1,title,otelos:literal), with sid2=ns:LectureUnit_title
sid1 is a member of the set otelos:individual, sid2 is a member of the set otelos:property. Both membership-statements are member of the set otelos:type.
```

Globally Scoped Properties In the second (special) case $s(ns:p,otelos:statement,p,y)$, p is an atomic value, and is a unique label within the current namespace ns . We then call p a *globally scoped* property, because it can be used as property for all kinds of statements. This definition corresponds to the property-centric way of defining properties in RDF(S).

Axiom 2.9 *For “globally scoped” properties, axiom 2.8 is extended, so that the names of attributes are unique even without conjunction with the source object:*

$$\forall sid1, x1, x2, p, y1, y2 \ s(ns : p, x1, p, y1) \wedge s(sid1, x2, p, y2) \Rightarrow ((sid1 = ns : p) \\ \wedge (x1 = x2) \wedge (y1 = y2)) \vee (p = type) \vee (p = subClassOf) \vee (p = subPropertyOf)$$

Comparison to RDF(S) and O-Telos O-Telos-RDF axiom 2.8 corresponds to O-Telos axiom 3. Axioms 2.6 and 2.7 correspond to O-Telos axioms 22 and 26. RDF(S) regards properties as a projection of the second argument of the RDF-statement. Therefore [3] state with their axioms 2 and 3 that each label of a statement is a resource and that it is identified by an URI. This holds for O-Telos (axiom 3) and O-Telos-RDF (axiom 2.8) as well, with the exception of literals and some property statements (type etc.) which yield IDs that are not RFC 2396 conform.

In contrast to O-Telos and to RDF, O-Telos-RDF allows both globally or object scoped properties, though the globally scoped properties are just a special case of the object scoped properties. So, as can be seen in example 3, the definition of properties is object centered and is included in the class definition rather than written outside of the class definition in separate statements. If different domains (as in RDF(S)) have to be expressed for a property, this property has to be an object scoped property. When using object scoped properties, different ranges are possible (in contrast to the current RDF(S) specification).

2.1.5 Literals and Other Primitive Types

Literals l are represented as individuals $s(l,l,l,l)$. The set of all literals is denoted by the individual `otelos:literal`. Other primitive types (like the ones defined in the recent XML schema specification, or Integer, Boolean, etc. from O-Telos) can be introduced in the same way. While the statement IDs for these primitive individuals are no URIs, this representation allows us to use these individuals consistently with current RDF(S)/XML statements, by simply including the value in a statement.

Comparison to RDF(S) and O-Telos Axioms 5 and 6 of [3] declare a literal as an object that is not a resource. These literals are instances of `rdfs:Literal`. In contrast to RDF(S), O-Telos (and O-Telos-RDF) declare predefined classes, such as `Literal (String)`, `Boolean`, `Integer`, etc.

2.2 Schema Definition Concepts and Predicates

2.2.1 type

Statements of the form $s(sid,x,type,y)$ denote class membership of x in y . We talk about x being an instance of y , where x and y represent either two individuals or two properties.

Axiom 2.10 Type statements can be written using the auxiliary predicate $type(x,y)$:

$$\forall sid, x, y s(sid, x, type, y) \Rightarrow type(x, y)$$

Axiom 2.11 All statements, where subject and object are different from the statement ID and which use the label “type”, are instances of the set $otelos:type$, represented by the statement $s(otelos:type, otelos:statement, type, otelos:statement)$:

$$\forall sid, x, c s(sid, x, type, c) \wedge (sid \neq x) \wedge (sid \neq c) \Leftrightarrow type(sid, otelos : type)$$

Axiom 2.12 The label “type” of these statements is unique in conjunction with the source object x and the destination object y . The statement identifier sid has therefore the form x_type_y :

$$\forall sid1, sid2, x, p, y s(sid1, x, p, y) \wedge s(sid2, x, p, y) \wedge (p = type) \Rightarrow (sid1 = sid2)$$

Axiom 2.13 Property statements can be written with the name of the property (instead of a new label) by using the auxiliary predicate $P(x,m,y)$:

$$\forall sid1, sid2, x, l, y, c, m, d s(sid1, x, l, y) \wedge s(sid2, c, m, d) \wedge type(sid1, sid2) \Rightarrow P(x, m, y)$$

Axiom 2.14 Properties P of a subject x are always expressed as a property statement, which is an instantiation of a property definition for the class c of which x is a member of:

$$\begin{aligned} & \forall x, l, y, c, m, d, sid1, sid2 type(x, c) \wedge P(x, m, y) \wedge s(sid1, c, m, d) \\ & \Rightarrow \exists s(sid2, x, l, y) \wedge type(sid2, sid1) \end{aligned}$$

Axiom 2.15 In case x is an instance of two classes c and d , which both define a property m , x has also to be an instance of a class g , which is a subclass of both c and d , and which also defines property m :

$$\begin{aligned} & \forall x, m, y, c, d, sid1, sid2, e, f (type(x, c) \wedge type(x, d) \wedge s(sid1, c, m, e) \wedge s(sid2, d, m, f)) \\ & \Rightarrow \exists g, sid3, h type(x, g) \wedge s(sid3, g, m, h) \wedge subClassOf(g, c) \wedge subClassOf(g, d) \end{aligned}$$

Axiom 2.15 handles multiple inheritance/instantiation where two classes of an object both define a common property, by demanding a third class the object is an instance of, which defines this property and makes instantiating the property for x unique.

Example 4: The following example shows the use of type-statements. Let us assume, that there is an individual called LectureUnit that has a property of type String labeled title. An individual called LectureUnit1 is an instance of LectureUnit and instantiates the property title with the value “Lecture Unit 1”. The examples use the namespaces `rdf:`, `rdfs:`, `s:`, `t:` and `otelos:` instead of the URIs with the statement-declarations.

```
<otelos:OTELOS
  <otelos:class ID="LectureUnit"><title s="otelos:literal">
  </otelos:class>
  <otelos:individual ID="LectureUnit1">
    <type s="t:LectureUnit"/><title>Lecture Unit 1</title>
  </otelos:individual>
  <otelos:property ID="LectureUnit1_title"><type s="t:LectureUnit_title"/>
  </otelos:property>
</otelos:OTELOS>
```

We have four main statements and two type statements

```
s(sid1,sid1,LectureUnit,sid1),s(sid2,sid1,title,otelos:literal),  
s(sid3,sid3,LectureUnit1,sid3),s(sid4,sid3,title,"Lecture Unit 1"),  
s(sid5,sid3,type,sid1), s(sid6,sid4,type,sid2)
```

In the following, we will not explicitly specify the type statement for properties like “LectureUnit1_title” in the XML serialization, whenever the label of the instantiated property is the same as the label used in the definition of that property and an explicit type statement is included for the object the instantiated property is associated to. These type definition tuples can be inferred by the O-Telos-RDF parser by using the label to look up the appropriate property definition within the class specified by the type statement for the object containing the instantiated property. If the label of the instantiated property statement is different from the label used in the property definition (used for multivalued properties and in metamodeling applications), we can use an XML serialization for instantiated properties which includes the type of the instantiated property as an additional XML-attribute “type=” to get a compact serialization.

Comparison to RDF(S) and O-Telos Instantiations of classes using “type” are done in the same way in RDF(S) as in O-Telos-RDF axioms 2.12 and 2.10 (corresponding to the O-Telos axioms 4 and 5). Furthermore, objects can instantiate the attributes defined in their classes in the same way in RDF(S) and O-Telos-RDF (axioms 2.13 and 2.14, corresponding to O-Telos axioms 7, 8 and 9). Axiom 2.11 corresponds to the O-Telos axioms 27 and 20. Differently to RDF(S), `otelos:type` stands on its own and is different from `otelos:property` (axiom 2.11, corresponding to O-Telos axiom 20).

Axiom 2.15 (corresponding to O-Telos axiom 17) handles multiple inheritance/instantiation. Such an axiom is not necessary in RDF(S) because properties are defined separately from classes anyway, and a given property can have just one range restriction. Axiom 2.13 (corresponding to O-Telos axioms 7 and 8) defines how to state properties in an RDF(S)-like way. We can therefore translate RDF(S)-like property statements like $P(x,m,y)$ into the corresponding O-Telos-RDF statements by generating a unique object identifier and a label for the statement representing the instantiated property statement, plus an additional statement for the instantiation relationship.

In contrast to RDF(S) [3], which only has three abstraction hierarchies (`rdfs:class`, classes, which are instances of `rdfs:class` and instances of these classes) and represents property instantiation implicitly (i.e. not by explicit statements), O-Telos and hence O-Telos-RDF allow arbitrarily many abstraction hierarchies, because their representation does not distinguish between metaclasses, classes and objects, and

instantiation is represented explicitly for properties as well as for individuals, as defined in axioms 2.13 and 2.14 (corresponding to O-Telos axioms 7, 8 and 9). This leads to instantiated property statements, which have a (possibly) different label than the property definitions they instantiate, and which can be instantiated again if needed. This is different from RDF(S), where instantiation is only explicit for individuals.²

2.2.2 Domain and Range

Domain and range properties like those in RDF(S) are not needed, as domain and range restrictions are already included in all property definitions.

Axiom 2.16 *Subjects and objects in a statement representing an instantiated property are typed by the corresponding property definition:*

$$\begin{aligned} & \forall sid1, sid2, x, l, y \ s(sid1, x, l, y) \wedge type(sid1, sid2) \\ & \Rightarrow \exists c, p, r \ s(sid2, c, p, r) \wedge type(x, c) \wedge type(y, r) \end{aligned}$$

If no domain or range restrictions are desired for a property p, it can be defined as a globally scoped property by `s(ns:p,otelos:statement,p,otelos:statement)`.

Comparison to RDF(S) and O-Telos In contrast to O-Telos-RDF, RDF(S) has to state domain and range using separate properties. [3] define the domain property in rules 19 to 21. It has a cardinality of zero or more which complies with O-Telos-RDF. Furthermore, [3] define the range property in rules 22 to 26. RDF(S) restricts the range property to one range constraint per property. In contrast, each O-Telos-RDF property definition has exactly one range constraint, but for different domains different property definitions with range constraints are possible (axiom 2.16). This corresponds to O-Telos and its axiom 14.

2.2.3 subClassOf

Statements of the form `s(sid,x,subClassOf,y)` denote a subclass relationship between x and y. We talk about x being a subclass of y, where x represents a class and y its superclass. x and y are individuals.

Axiom 2.17 *The label “subClassOf” in subClassOf-statements is unique in conjunction with the source object x and the destination object y, thus sids are of the form x_subClassOf_y:*

$$\frac{\forall sid1, sid2, x, l, y \ s(sid1, x, l, y) \wedge s(sid2, x, l, y) \wedge (l = subClassOf)}{\Rightarrow (sid1 = sid2)}$$

²The missing explicit instantiation of properties in RDF(S) is actually the main reason for its restriction to three abstraction levels, as instantiated property statements neither have a label nor statement ID, which makes it impossible to uniquely reference these statements as first class objects.

Axiom 2.18 We can also define an auxiliary predicate $subClassOf(x,y)$:

$$\forall sid, c, d \ s(sid, c, subClassOf, d) \Rightarrow subClassOf(c, d)$$

Axiom 2.19 The set of all $subClassOf$ -statements is represented by the statement $s(otelos:subClassOf,otelos:individual,subClassOf,otelos:individual)$.

Axiom 2.20 All statements with a label “ $subClassOf$ ” whose sid is not equal to subject and object are members of the set $otelos:subClassOf$:

$$\begin{aligned} & \forall sid, c, d \ s(sid, c, subClassOf, d) \wedge (sid \neq c) \wedge (sid \neq d) \\ & \Leftrightarrow type(sid, otelos : subClassOf) \end{aligned}$$

The $subClassOf$ relationship is a partial order on the statement identifiers, which is reflexive as well as transitive, and does not contain cycles, but uses reflexivity to state equality. We omit the corresponding axioms because of space constraints.

Axiom 2.21 Class membership is inherited upwardly to the superclasses:

$$\forall x, c, d \ type(x, d) \wedge subClassOf(d, c) \Rightarrow type(x, c)$$

Comparison to RDF(S) and O-Telos Our O-Telos axioms correspond closely to the O-Telos and RDF(S) axioms. Contrary to RDF(S), the $subClassOf$ -relationship is defined as a partial order, which is reflexive as well as transitive. $subClassOf$ does not contain cycles, but uses reflexivity just to state equality. Class membership inherits upwardly to the superclasses in all formalisms.

2.2.4 $subPropertyOf$

Axiom 2.22 Subclasses, which define properties with the same name as properties of their classes, refine these properties:

$$\begin{aligned} & \forall sid1, sid2, c, d, e, f, m \ subClassOf(d, c) \wedge s(sid1, c, m, e) \wedge s(sid2, d, m, f) \\ & \Rightarrow subClassOf(f, e) \wedge subPropertyOf(sid2, sid1) \end{aligned}$$

Axiom 2.23 Furthermore, O-Telos-RDF requires subproperties of properties to refine subject and object of the properties as well.

$$\begin{aligned} & \forall sid1, sid2, c, d, e, f, m, l \ subPropertyOf(sid2, sid1) \wedge s(sid1, c, m, e) \\ & \wedge s(sid2, d, l, f) \Rightarrow subClassOf(d, c) \wedge subClassOf(f, e) \end{aligned}$$

For compatibility reasons to RDF(S), we define $otelos:subPropertyOf$ separately from $subClassOf$, even though it has all axioms from $subClassOf$, plus additional ones which are relevant for $subPropertyOf$ only. Statements of the form $s(sid,x,subPropertyOf,y)$ denote a subproperty relationship between x and y .

As for “ $subClassOf$ ”, the label “ $subPropertyOf$ ” in statements is unique in conjunction with the source object x and the destination object y , the $subPropertyOf$ relationship is again a partial order on the statement identifiers, and property membership is inherited upwardly to the superproperties. All $subPropertyOf$ -statements are members of the set $otelos:subPropertyOf$, which is represented by the statement $s(otelos:subPropertyOf,otelos:property,subPropertyOf,otelos:property)$.

Comparison to RDF(S) and O-Telos Close to `otelos:subClassOf` treats O-Telos-RDF `otelos:subPropertyOf` (like O-Telos, which does not distinguish these two concepts at all). This being consistent with RDF(S), [3] defines `rdfs:subPropertyOf` very similar to `rdfs:subClassOf`. O-Telos and O-Telos-RDF declare a special set for the `subPropertyOf`-statements. Property membership in O-Telos-RDF is inherited upwardly to the superclasses. [3] has to define this by introducing new statements for superproperties (rule 17), as no instantiation statements for properties exist in RDF(S). Axioms 2.22 and 2.23 (O-Telos axioms 15 and 16) do not exist in RDF(S).

Due to space constraints, we are not able to include the specification of containers like `rdf:Seq` in O-Telos-RDF, but refer the reader to the extended report. The O-Telos-RDF specification exploits the possibility to state properties about properties, and can avoid the introduction of untyped containers like `rdf:Seq` which have disadvantages for precise semantical modelling.

3 Discussion

In this paper we described the formalization of an RDF(S) extension called O-Telos-RDF, which provides enhanced functionalities for meta-modeling and reified statements. The formalization is based closely on the formalization of the modeling language O-Telos, which is based on a semantic network model similar to RDF(S). Axioms and constraints are clearly defined. In the longer report, we include all these axioms, together with their counterparts in O-Telos and RDF(S), as well as some longer examples.

Compared with RDF, O-Telos-RDF shows its advantages in allowing easier reification of statements, and in metamodelling applications, where more than three abstraction hierarchies are needed. Furthermore, the class centric approach to property definition allows the definition of properties with the same name for different domains, which have different ranges (not possible in RDF(S)).

These properties seem to make O-Telos-RDF more similar to DAML+OIL than RDF(S) is (in the sense that DAML+OIL is more easily/naturally represented in O-Telos-RDF than in RDF(S)). A detailed discussion of this relationship will be included in a forthcoming report, together with a discussion of the reasoning functionalities for O-Telos-RDF based on Datalog^- .

As O-Telos-RDF is a strict extension of RDF(S), all RDF(S) definitions can be translated into O-Telos-RDF statements. Translation from O-Telos-RDF to RDF(S) is only possible, if the O-Telos-RDF definitions don't use O-Telos-RDF specific extensions, e.g. more than three abstraction levels, or properties of properties.

References

- [1] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. In *Internet Official Protocol Standards (STD 1)*, RFC. The Internet Society, rfc 2396 edition, 1998.
<http://www.ietf.org/rfc2396.txt>.
- [2] D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Technical report, World Wide Web Consortium (W3C), 2000.
<http://www.w3.org/TR/rdf-schema>.
- [3] Wolfram Conen and Reinhold Klapsing. A logical interpretation of RDF. In *Linköping Electronic Articles in Computer and Information Science, Semantic Web Area*, volume 5. Linköping University Electronic Press, December 2000.
<http://www.ep.liu.se/ea/cis/2000/013/>.
- [4] W3C Working Group. W3C Resource Description Framework (RDF) Model and Syntax Specification.
<http://www.w3.org/TR/REC-rdf-syntax/>, February 1999.
- [5] ISO/IEC. *Information technology - Information Resource Dictionary System (IRDS) framework, ISO/IEC 10027*, 1990 (E).
- [6] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 – 192, 1995.
- [7] M. Jeusfeld. *Änderungskontrolle in deduktiven Objektbanken*. Infix-Verlag, St. Augustin, Deutschland, 1992.
- [8] M. Jeusfeld and ConceptBase Team. Application papers. Technical report, Chair of Computer Science V - Information Systems, RWTH Aachen, 2000.
<http://www-i5.informatik.rwth-aachen.de/CBdoc/cblit.html#cb-app>.
- [9] B.M. Kramer, V.K. Chandhri, M. Koubarakis, T. Topaloglon, H. Wang, and J. Mylopoulos. Implementing Telos. *SIGART Bulletin*, 2(3), June 1991.
- [10] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: A language for representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4), 1990.

- [11] Wolfgang Nejdl, Martin Wolpes, and Christian Capelle. The RDF schema specification revisited. In *Modellierung 2000*, St. Goar, Germany, April 2000.
<http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2000/modeling2000/wolpers.pdf>.
- [12] Jeff Z. Pan and Ian Horrocks. Metamodeling architecture of web ontology languages. In *Proceedings of the Semantic Web Working Symposium*, pages 131–149, Stanford, July 2001.