Extending RDF(S) with Contextual and Definitional Knowledge

Alexandre Delteil, Catherine Faron-Zucker
ACACIA project, INRIA, 2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France
{Alexandre.Delteil, Catherine.Faron}@sophia.inria.fr

Abstract

RDF(S) is the emerging standard for knowledge representation on the Web. In the European IST project CoMMA dedicated to ontology guided information retrieval in a corporate memory, the semantic annotations describing the Intranet documents are represented in RDF(S). In this context, the RDF(S) expressivity appears to be too much limited. Compared to object-oriented representation languages, description logics, or conceptual graphs, RDF(S) does not enable to define classes or properties nor represent axioms inside an ontology. In this paper, we propose an extension of RDF(S) to express this kind of definitional knowledge, and more generally contextual knowledge on the Semantic Web. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

1 Introduction

The need of a Semantic Web is now well recognized and always more emphasized [Berners Lee, 1999]. The huge amount of information available on the web has become overwhelming, and knowledge based reasoning now is the key to lead the Web to its full potential. In the last few years, a new generation of knowledge based search engines has arisen, among which the most famous are *SHOE* [Luke *et al.*, 1997] and *Ontobroker* [Fensel *et al.*, 1998]. They rely on extensions of HTML to annotate Web documents with semantic metadata, thus enabling semantic content guided search. For interoperability on the Web, the importance of widely accepted standards is emphasized. *Resource Description Framework* (RDF) is the emerging standard proposed by the W3C for the representation and exchange of metadata on the Semantic Web [RDF, 1999]; it has an XML syntax. *RDF Schema* (RDFS) is the standard dedicated to the representation of ontological knowledge used in RDF statements [RDFS, 2000].

In the context of the 'CoMMA' European IST project, RDFS is the knowledge representation language used to annotate the Intranet documents of an organization. These annotations are exploited for knowledge based information retrieval on the Intranet by using the inference engine *CORESE* implemented in our team [Corby *et al.*, 2000]. However the expressivity of RDF(S) appears too much limited to represent the ontological knowledge of the corporate memory. Inference rules representing domain axioms, class and property definitions are crucial for intelligent information retrieval on the Web. The need for inference rules is well-known since the first information retrieval systems on the Semantic Web. Axiomatic knowledge, algebraic properties of relations, or domain axioms are the key to discover implicit knowledge in Web page annotations so that information retrieval be independent of the point of view adopted when annotating [Heflin *et al.*, 1998]. [Martin *et al.*, 2000] claim the need for additional features and conventions in RDF.

When compared to object-oriented knowledge representation languages, description logics, or conceptual graphs, RDF(S) does not enable to define classes or properties nor represent axioms [DAML, 2001; OIL, 2000]. In this paper, we propose an extension of RDF(S) with class, property and axiom definitions. We call it DRDF(S) for *Defined Resource Description Framework*. DRDFS more generally enables to express contextual knowledge on the Web. The RDF philosophy consists in letting anybody free to declare anything about any resource. Therefore the knowledge of who and in which context a special annotation has been stated is crucial. DRDF(S) enables to assign a context to any cluster of annotations, in particular for definitional contexts. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

In the next section, we present the RDF(S) model. Section 3 is dedicated to the comparison of RDF(S) and the Conceptual Graphs model. Section 4 presents an extension of RDF(S) with contexts, and section 5 an extension of existential quantification handling. The RDF extensions for defining classes, properties and axioms are presented in sections 6, 7 and 8. The metamodel of DRDF(S) is described in section 9. Section 10 is dedicated to a comparison between DRDF(S) and other Web languages.

2 The RDF(S) Model

2.1 RDF and RDFS

RDF is the emerging Web standard for annotating resources, such as images or documents, with semantic metadata [RDF, 1999]. These Web resources are identified by their URIs. In addition, anonymous resources provide a limited way of existential quantification. An RDF description consists in a set of statements; each one specifying a value of a property of a resource. A statement is thus a triple (resource, property, value), a value being either a resource or a literal. The RDF data model is close to semantic nets. A set of statements is viewed as a directed labeled graph: a vertex is either a resource or a literal; an arc between two vertices is labeled by a property. RDF is provided with an XML syntax.

Figure 1 presents an example of RDF graph and its XML serialization. It is the annotation of the Web page of T-Nova which is a subdivision of Deutsche Telekom. The examples highlighting our paper are all based on the CoMMA ontology.

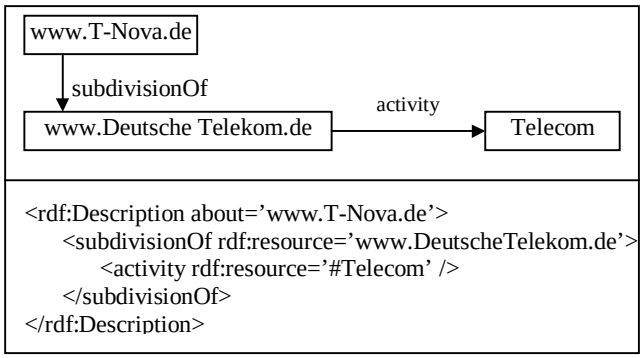


Figure 1. An example of RDF annotation.

RDF Schema (RDFS) is dedicated to the specification of schemas representing the ontological knowledge used in RDF statements [RDFS, 2000]. A schema consists in a set of declarations of classes and properties. Multi-inheritance is allowed for both classes and properties. A property is declared with a signature allowing several domains and one single range: the domains of a property constraint the classes this property can be applied to, and its range the class the value of this property belongs to.

The RDFS metamodel is presented in Figure 2. This definition is recursive: the terms of RDFS are themselves defined in the RDFS model. More precisely, the RDFS metamodel itself is defined as a set of statements by using the two core RDFS properties: subclassOf and type which denote respectively the subsumption relation between classes and the instantiation relation between an instance and a class.

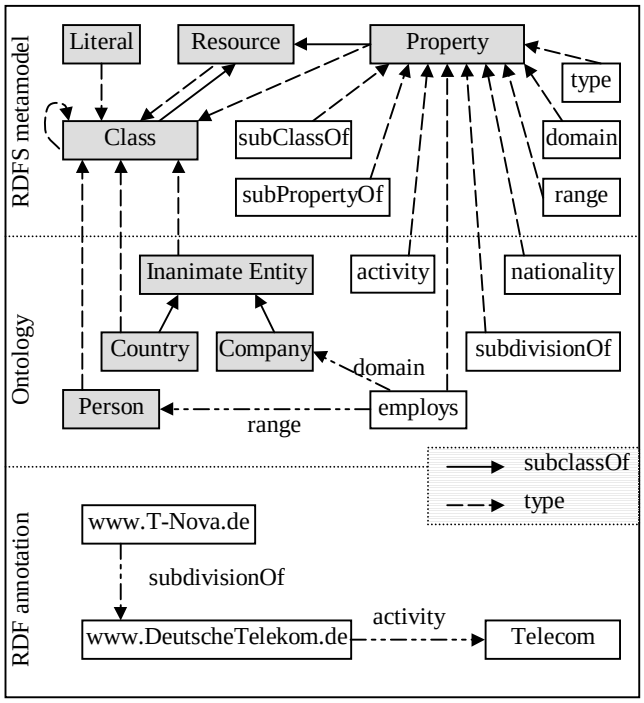


Figure 2. The RDFS metamodel and an RDFS schema.

To represent domain specific knowledge, a schema is defined by refining the core RDFS. As shown in Figure 2, domain specific classes are declared as instances of the "Class" resource and domain specific properties are declared as instances of the "Property" resource. The "subclassOf" and "subPropertyOf" properties enable to define class hierarchies and property hierarchies.

2.2 RDF Limitations

A Triple Model. The RDF data model is a triple model: an RDF statement is a triple (resource, property, value). When asserted, RDF triples are clustered inside annotations. An annotation can thus be viewed as a graph, subgraph of the great RDF graph representing the whole set of annotations on the Web. However, "there is no distinction between the statements made in a single sentence and the statements made in separate sentences" [RDF, 1999]. Let us consider two different annotations relative to two different research projects which the employee 46 of T-Nova participates to:

- {(employee-46, worksIn, T-Nova), (employee-46, project, CoMMA), (employee-46, activity, endUser)}

- {(employee-46, worksIn, T-Nova), (employee-46, project, projectX), (employee-46, activity, developer)}.

The whole RDF graph does not distinguish between these two clusters of statements. Employee 46 is both endUser and developer: the knowledge of which activity inside of a project he is implicated in is lost.

RDF Reification. The RDF model is provided with a reification mechanism dedicated to higher order statements about statements. A statement (r, p, v) is reified into a resource s described by the four following properties: the *subject* property identifies the resource r, the *predicate* property identifies the original property p, the *object* property identifies the property value v, the *type* property describes the type of s; all reified statements are instances of *Statement*. Figure 3 presents the following reification: 'Observer-3002 says that the rating of Newsletter-425 is seminal'.

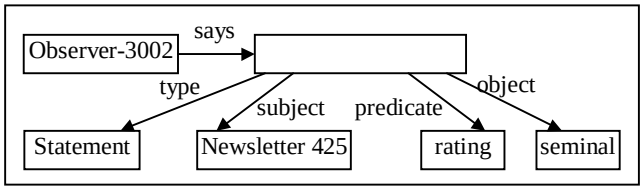
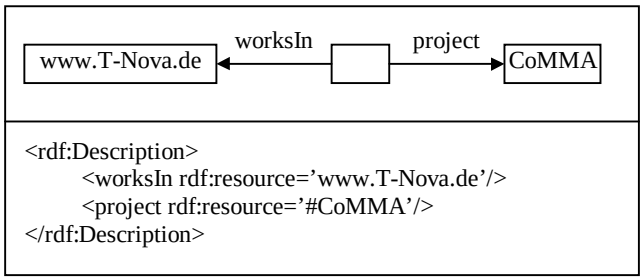


Figure 3. An example of reification.

Let us consider now the reification of a set of statements. It requires the use of a container to refer to the collection of the resources reifying these statements. This leads to quite complicate graphs (see Figure 10 in [RDF, 1999]). Moreover a statement containing an anonymous resource can not always be reified: the values of the properties *subject* and *object* must have an identifier.

Existential quantification. The RDF model focuses on the description of identified resources but allows a limited form of existential quantification through the anonymous resource feature. Let us consider the following RDF statements describing an anonymous resource:




```
<rdf:Description>
      <worksIn rdf:resource='www.T-Nova.de'/>
      <project rdf:resource='#CoMMA'/>
</rdf:Description>
```

Figure 4. An example of anonymous resource.

This existential quantification is handled by automatically generating an ID for the anonymous resource. However, such a handling of existential knowledge through constants is a limited solution and a graph containing a cycle with more than one anonymous resource can not be represented in RDF (Figure 5).

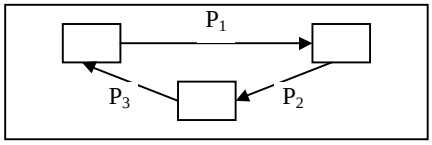


Figure 5. An RDF graph without XML serialization

Classes and properties. An RDF Schema is made of atomic classes and properties. The RDFS model does not enable the definition of classes or properties. More generally, inferences cannot be represented in the model.

3 The RDF(S) and Conceptual Graphs Models

3.1 The Conceptual Graphs Model

Conceptual Graphs [Sowa, 1984; Sowa, 1999] is a knowledge representation model descending from Existential Graphs [Pierce, 1932] and Semantic Networks. A conceptual graph is a bipartite (not necessarily connected) graph composed of concept nodes, and relation nodes describing relations between these concepts.

Each concept node c of a graph G is labeled by a couple $<type(c)\ ,\ referent(c)>$, where *referent(c)* is either the generic marker * corresponding to the existential quantification or an individual marker corresponding to an identifier; M is the set of all the individual markers. Each relation node r of a graph G is labeled by a relation type *type(r)*; each relation type is associated with a signature expressing constraints on the types of the concepts that may be linked to its arcs in a graph.

Concept types (respectively relation types of same arity) build up a set T_c (resp. T_r) partially ordered by a generalization/specialization relation $\leq_c$ (resp. $\geq_r$). (T_c, T_r, M) defines the *support* upon which conceptual graphs are constructed. A support thus represents a domain ontology.
The semantics of the Conceptual Graphs model relies on the translation of a graph G into a first order logic formula thanks to a Φ operator as defined in [Sowa, 1984]: $\Phi(G)$ is the conjunction of unary predicates translating the concept nodes of G and n-ary predicates translating the n-ary relation nodes of G; an existential quantification is introduced for each generic concept.

Conceptual graphs are provided with a generalization/specialization relation $\leq_G$ corresponding to the logical implication: $G_1 \leq_G G_2$ iff $\Phi(G_1) => \Phi(G_2)$. The fundamental operation called *projection* enables to determine the generalization relation between two graphs: $G_1 \leq_G G_2$ iff there exists a projection π from G_2 to G_1. π is a graph morphism such that the label of a node n_1 of G_1 is a specialization of the label of a node n_2 of G_2 with $n_1 = \pi(n_2)$. Reasoning with conceptual graphs is based on the projection, which is sound and complete with respect to logical deduction.

3.2 Mapping of the RDF(S) and CG models

The RDFS and CG models share many common features and a mapping can easily be established between RDFS and a large subset of the CG model. An in-depth comparison of both models is studied in [Corby *et al.*, 2000].

- Both models distinguish between ontological knowledge and assertional knowledge. First the class (resp. property) hierarchy in a RDF Schema corresponds to the concept (resp. relation) type hierarchy in a CG support; this distinction is common to most knowledge representation languages. Second, and more important, RDFS properties are declared as first class entities like RDFS classes, in just the same way that relation types are declared independently of concept types. This is this common handling of properties that makes relevant the mapping of RDFS and CG models. In particular, it can be opposed to object-oriented approaches, where properties are defined inside of classes.

- In both models, the assertional knowledge is positive, conjunctive and existential.

- Both models allow a way of reification.

- In both models, the assertional knowledge is represented by directed labeled graphs. An RDF graph G may be translated into a conceptual graph CG as follows:

  - Each arc labeled with a property p in G is translated into a relation node of type p in CG.
  - Each node labeled with an identified resource in G is translated into an individual concept in CG whose marker is the resource identifier. Its type corresponds to the class the identified resource is linked to by a *rdf:type* property in G.
  - Each node labeled with an anonymous resource in G is translated into a generic concept in CG. Its type corresponds to the class the anonymous resource is linked to by a *rdf:type* property in G.

Regarding the handling of classes and properties, the RDF(S) and CG models differ on several points. However these differences can be quite easily handled when mapping RDF and CG models.

- RDF binary properties versus CG n-ary relation types: the RDF data model intrinsically only supports binary relations, whereas the CG model authorizes n-ary relations. However it is possible to express n-ary relations with binary properties by using an intermediate resource with additional properties of this resource giving the remaining relations [RDF, 1999].

- RDF multi-instantiation versus CG mono-instantiation: the RDF data model supports multi-instantiation whereas the CG model does not. However the declaration of a resource as instance of several classes in RDF can be translated in the CG model by generating the concept type corresponding to the most general specialization of the concept types translating these classes.

- Property and relation type signatures: in the RDF data model, a property may have several domains whereas in the CG model, a relation type is constrained by a single domain. However the multiple domains of an RDF property may be translated into a single domain of a CG relation type by generating the concept type corresponding to the most general specialization of the concept types translating the domains of the property.

3.3Additional expressivity of the CG model

In addition to the features the CG model shares with RDF(S), it is provided with additional features insuring a greater expressivity. Regarding the existing mapping between both models, these features will be the key to an extension of RDF(S) based on the CG model [Delteil *et al.*, 2001].

A graph model

A conceptual graph represents a piece of knowledge separate from the other conceptual graphs of the base it belongs to. Let us consider again the two projects of T-Nova which Employee-46 participates in. The statements relative to one project are clustered in one conceptual graph and then separated from the statements relative to the other projects.
The two conceptual graphs are the following:

- [Project : CoMMA] <--(project)<-- [T : Employee-46] -->(activity)--> [EndUser: *].

- [Project : projectX] <--(project)<-- [T : Employee-46] -->(activity)--> [Developer : *].

A CG base is a set of conceptual graphs that cannot be decomposed in smaller pieces of knowledge without loss of information.

Reification

A conceptual graph g is reified into a marker whose value is g.
Let us consider again the following reification: 'Observer-3002 says that the rating of Newsletter-425 is seminal'. It is represented by the following conceptual graph:

[T : Observer-3002] ---> (says) ---> [Proposition :
[T : Newsletter-425] ---> (rating) ---> [T : seminal]].

In the RDF model, the reification of a set of statements requires the use of a container to refer to the collection of the resources reifying these statements. In the CG model, since the notion of graph is intrinsic to the model, the

equivalent reification remains based on the initial basic mechanism.

Existential Quantification
The CG model allows to represent every existential, positive and conjunctive proposition without any restriction.

Type definitions and axioms
In the CG model, concept type and relation type are either atomic or defined [Leclere, 1997]. Graph rules allow the representation of inference rules [Salvat and Mugnier, 1996].

Starting from the correspondence between RDF(S) and the conceptual graph model, we propose an extension of RDF(S) based on the CG model to provide the former with an expressivity equivalent to the one of the latter. We call this extension DRDFS.

4 Extending RDFS with contexts

The RDF model provides no way of expressing independent pieces of knowledge. We propose to extend RDF with a notion of context to express the clustering of statements much more easily than RDF containers. A context identifies a sub-graph of the whole RDF graph, so that a triple can be stated inside of a special context. This extension is based on the similarities between the RDF and CG models: a context is just the translation of a conceptual graph. The CG model provides a direct way of expressing independent pieces of knowledge through graphs: a conceptual graph implicitly defines a context. The representation of contexts for various applications (quotations, viewpoint, …) is direct in the CG model. Conceptual graphs are particularly useful as definitional contexts enabling the definition of concepts or axioms. By introducing contexts in RDF, we propose a very general mechanism that will be the keystone of further extensions, like class or rule definitions.

To extend RDFS with contexts, we introduce the following new RDF primitives:
'Context': A context is a resource of type *Context*. *Context* is a subclass of *rdfs:Class*.
'isContextOf': A resource is linked by a *isContextOf* property to the context it belongs to.
'referent': An anonymous resource is linked by a *referent* property to the identified resource it refers to.

The rules for constructing RDF contexts are based on the translation of conceptual graphs into RDF:

- An individual concept [C: r] of a conceptual graph G is represented by three RDF triples (c∅, type, C), (c∅, referent, r), (G, isContextOf, c∅), where c∅ is an anonymous resource (whose ID is automatically generated by RDF parsers).
- A generic concept [C: *] of a conceptual graph G is represented by two RDF triples (c∅, type, C), (G, isContextOf, c∅).
- A generic concept [C: *x] of a graph G is represented by three RDF triples (c∅, type, C), (c∅, referent, x), (G, isContextOf, c∅), where x is an instance of the class *Variable* (this class will be further described in next section).
- A relation R between two concepts [C_1: r_1] and [C_2: r_2] of a conceptual graph G is represented by an RDF property P between the two anonymous resources $c\emptyset_1$ and $c\emptyset_2$.
- The resource G is an instance of the class Context; this is represented by the triple (G, type, Context).

Note that to represent a context, it could be sufficient to link a single anonymous resource of it to the resource G representing it by the *isContextOf* property.

Let us consider again the two projects of T-Nova which Employee-46 participates in. As shown in Figure 6, the statements relative to one project can now be clustered in a context and then separated from the statements relative to the other projects.

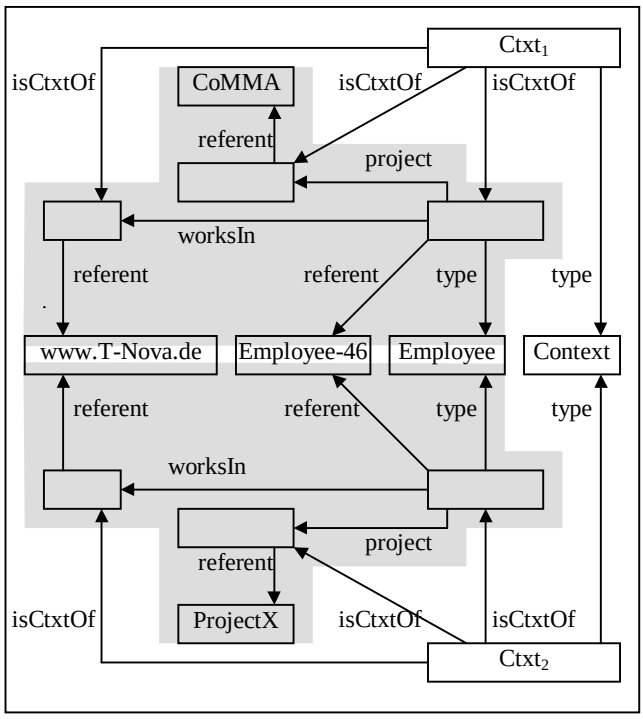


Figure 6. Two contexts about the resource Employee-46.

The rules for extracting the set S of the triples belonging to a context from the whole RDF graph are the following:

- Select a resource G of type Context; S <-- {(G, type, Context)}.
- Select all the anonymous resources $c\varnothing_i$ for which the value of the *isContextOf* property is G; for each i, S <-- S ∪ {(G, isContextOf, $c\varnothing_i$)}.
- Select all the identified resources r_j values of a *referent* property of a resource $c\varnothing_i$; S <-- S ∪ {($c\varnothing_i$, referent, r_j)}.
- Select all the properties p_{ik} between two resources $c\varnothing_i$ and $c\varnothing_k$; S <-- S ∪ {($c\varnothing_i$, p_{ik}, $c\varnothing_k$)}.

Regarding the whole RDF graph, a context defines, just like a conceptual graph, a piece of knowledge, i.e. an independent clustering of statements. A context is defined from a resource G of type Context as the largest subgraph of the whole RDF graph whose all internal nodes excepted G are anonymous resources $c\varnothing_i$. A context is thus an abstraction that enables to talk about representations of resources (through anonymous resources) rather than directly about resources. For instance, in Figure 6, the resource Empoyee-46 is referred to by two distinct anonymous resources in two different contexts. Anonymous resources are "externally identified" by the *referent* property.
This general notion of context will appear of particular interest for expressing definitional contexts.

5 Extension of RDF(S) with existential quantification

The RDF model allows a limited form of existential quantification through the anonymous resource feature. The introduction of the *referent* property provides the RDF model with a general mechanism for existential quantification handling.

To extend RDFS with existential quantification, we introduce the following new RDF primitives:
'Variable': A variable is a resource of type *Variable*. *Variable* is a subclass of *rdfs:Class*.
'parameter': A variable is linked by a *parameter* property to the context it belongs to.

An existential quantification is represented by an anonymous resource described by a *referent* property whose value is an instance of *Variable*. The scope of a variable is the context it belongs to, just like in first-order logic, where the scope of a variable is the formula it belongs to.

In an RDF graph, an anonymous resource can be duplicated into several anonymous resources coreferencing a same variable; the new graph remains semantically equivalent to the initial one. This enables the XML serialization of RDF graphs embedding a cycle with anonymous resources. Figure 7 presents one DRDF graph semantically equivalent to the RDF graph of Figure 5 that could not be serialized in the XML syntax. The cycle is resolved by introducing a second anonymous resource and two *referent* properties sharing the same value:

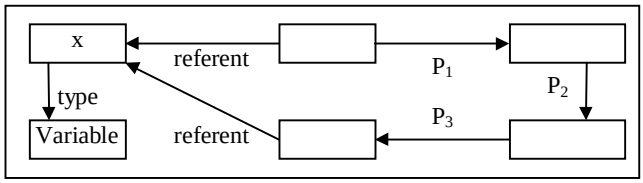


Figure 7. An example of existential quantification.

6 Extending RDFS with class definition

DRDF(S) class definition is descended from type definition in the CG model. A class definition is a monadic abstraction, i.e. a context whose one resource of type Variable is considered as formal parameter.

To extend RDFS with class definitions, we introduce the following new RDF primitives:
'DefinedClass': A defined class is of type *DefinedClass*. *DefinedClass* is a subclass of *rdfs:Class*.
'hasDefinition': A defined class is linked by a *hasDefinition* property to its definitional context.
'formalParameter': The variable linked to the definitional context by a *FormalParameter* property corresponds to the formal parameter of a monadic lambda abstraction.

Figure 8 describes the definition of the 'WebPage' class, as a document having HTML for representation system. The XML serialization of this graph is provided in appendix 2.

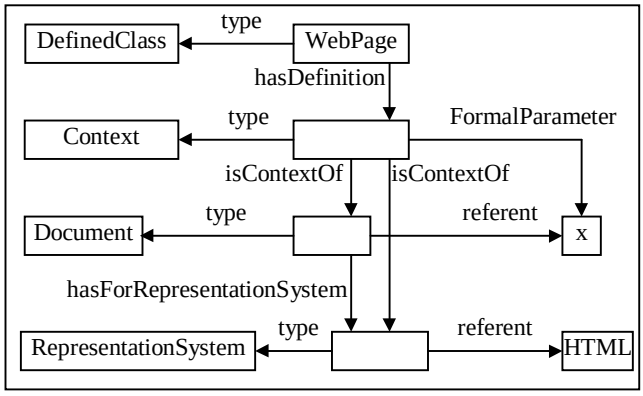


Figure 8. Definition of the 'WebPage' class.

7 Extending RDFS with property definition

DRDF(S) property definition is descended from type definition in the CG model. A property definition is a diadic abstraction, i.e. a context whose two resources of type Variable are considered as formal parameters.

To extend RDFS with property definitions, we introduce the following new RDF primitives:
'DefinedProperty': A defined property is of type *DefinedProperty*. *DefinedProperty* is a subclass of *rdf:Property*.
'firstFormalParameter' and 'secondFormalParameter': The variables linked to the definitional context by these properties correspond to the formal parameters of a diadic lambda abstraction.

Figure 9 describes the definition of the 'colleague' property, as a relation between two persons working in the same institute.

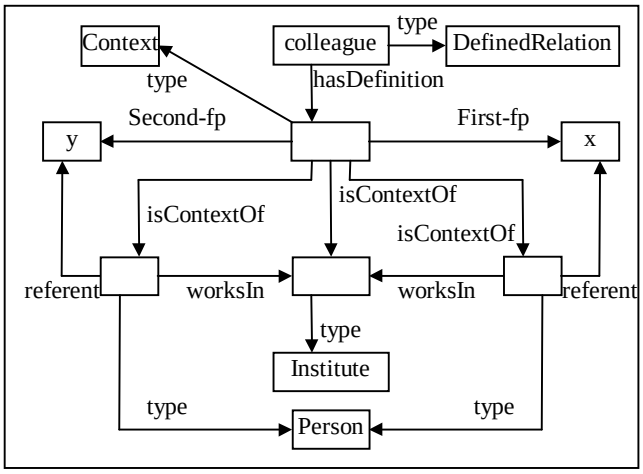


Figure 9. Definition of the 'colleague' property.

8 Extending RDFS with axioms

DRDF(S) axiom definition is descended from graph rules in the CG model. An axiom is a couple of lambda abstractions, i.e. two contexts representing the hypothesis and the conclusion.

To extend RDFS with axiom definitions, we introduce the following new RDF primitives:
'Axiom': An axiom is a resource of type *Axiom*. *Axiom* is a subclass of *Context*.
'if': An axiom is linked by an *if* property to the context defining its hypothesis.
'then': An axiom is linked by a *then* property to the context defining its conclusion.

The variables linked by a *formalParameter* property to the resource of type *Axiom* correspond to the formal parameters common to the two lambda abstractions.
Figure 10 describes the definition of the axiom "If x is colleague of y, then y is colleague of x".

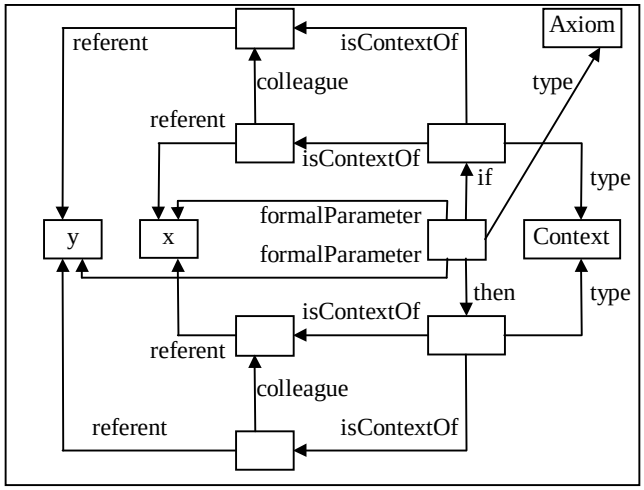


Figure 10. Definition of an axiom.

9 The Defined Resource Description Framework Schema (DRDFS)

9.1 The RDFS metamodel

The extensions introduced in previous sections are a refinement of the core RDFS and remain totally compliant with the RDF triple model. We call Defined Resource Description Framework Schema (DRDFS) the set of RDFS primitives augmented with the ones we introduce. The namespace prefix 'drdfs' is used to differentiate these new elements from the standard RDFS ones. For readability, we respect the RDFS convention that the first letter of class names is capital while the first letter of property names is small.
The metamodel of DRDFS is presented in Figure 11; its XML serialization is provided in Appendix 1.

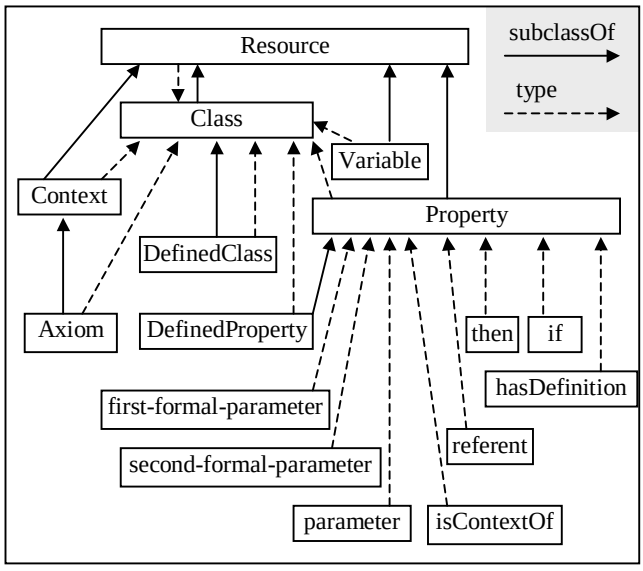


Figure 11. The DRDFS metamodel.

9.2Semantics of DRDFS

The semantics of DRDFS relies on its translation into the CG formalism. Conceptual graphs are themselves translated into first order logic formulae thanks to the Φ operator defined in [Sowa, 1984].

9.3Reasoning with DRDFS

The RDF model is dedicated to knowledge representation and interoperability on the Web. It does not address the problem of reasoning with the formalized knowledge; the only inference mechanisms it provides are the subsumption relations between classes and properties. Regarding the mapping established between the RDF and CG models in [Corby *et al.*, 2000], CG engines are good candidates for reasoning on the Semantic Web: the CORESE system developed in our team is a first step in this direction. Algorithms will be implemented in CORESE for reasoning with type definitions. They will be based on [Leclere, 1997] and will enable reasoning with the full DRDFS.

10 Related Work

Several languages for ontology representation and exchange are existing [Corcho and Gomez-Perez, 2000], among which RDF(S), OIL [Fensel *et al.*, 2000] and DAML [DAML, 2001] are dedicated to the Semantic Web. Like DRDF(S), OIL and DAML are tentatives of improvement of RDF(S); they are defined as an RDF Schema.

OIL enables to define classes and restrict property ranges and domains through boolean combinations of classes. In particular, it enables negation in class definitions, which is not provided in DRDFS. OIL is based on a DL. When compared to it, what DRDFS provides with its CG's expressivity is the possibility to express any positive, conjunctive and existential graph in a definition. The absence of variables in DLs does not enable to express RDF graphs embedding cycles; the class definitions in OIL are then limited to 'serializable' graphs. Contrary to OIL, DRDFS stays in the spirit of RDF(S), namely the representation of positive, conjunctive and existential knowledge. In our opinion, this better meets the needs of the Semantic Web.

DAML provides primitives to express relations between classes (disjonction, intersection, union, complementarity, ...) and enrich properties (minimal and maximal cardinality, transitivity, inverse, ...). DAML is provided with OOL features. It provides no mechanism for class or property definitions. It is therefore orthogonal to both OIL and DRDFS. As the merge of DAML and OIL led to DAML+OIL, it should be interesting to integrate the DAML features into DRDF(S).

In addition, DRDFS addresses the problem of the representation of contextual knowledge on the semantic web. This is of special interest to identify the origin of an annotation on the Web.

11 Conclusion

DRDF(S) is an extension of RDF(S) dedicated to ontology representation on the Semantic Web. It enables the representation of axioms, class and property definitions in ontologies. More generally, it provides a way to represent contextual knowledge on the Web.

In the framework of the CoMMA project, DRDF(S) should enable the representation of rich domain ontologies for intelligent IR in a company's Intranet. Since DRDF(S) is an RDF Schema, it is compliant with existing RDF parsers. However the semantics of the primitives specific to DRDF(S) can not be understood by them. We are currently working on a DRDF(S) interpreter for the existing platform CORESE.

The grounds of DRDF(S) rely on the existing mapping between RDF(S) and CGs; it is an extension of RDF(S) guided by the CG features. Regarding the similarities the RDF(S) and CG models share, the latter could contribute to the elaboration of a standard language for knowledge representation, interoperability and reasoning on the Semantic Web. We hope that DRDF(S) will contribute to the ongoing work of the W3C committee for improving RDFS and meet the needs of the e-business community.

References

[Berners Lee, 1999]. T. Berners Lee. Weaving the Web, Harper San Francisco.

[Chein and Mugnier, 1997] M. Chein, M.L. Mugnier. Positive Nested Conceptual Graphs. In proc. of the 5th International Conference on Conceptual Structures (ICCS'97), Seattle, WA, USA, Lukose D., Delugach, Keeler M. Searle L. and Sowa J. eds, Lecture Notes in Artificial Intelligence, LNIA 1257, Springer Verlag, p. 95-109, 1997.

[Corby *et al.*, 2000] O. Corby, R. Dieng, C. Hebert. A conceptual graph model for W3C Resource Description Framework. In proc. of the 8th International Conference on Conceptual Structures (ICCS'00), Darmstadt, Germany, Lecture Notes in Artificial Intelligence, LNCS 1867, Springer Verlag, 2000.

[Corcho and Gomez-Perez, 2000] O. Corcho, A. Gomez-Perez. A Roadmap to Ontology Specification Languages. In proc. of the 12th EKAW 2000, Juan-Les-Pins, France, LNAI 1937, Springer Verlag, p. 80-96, 2000.

[DAML, 2001] Darpa Agent Markup Language. http://www.daml.org.

[Delteil *et al.*, 2001] A. Delteil, C. Faron-Zucker and R. Dieng. Extensions of RDFS Based on the Conceptual Graph Model. To appear in proc. of ICCS'2001.

[Fensel *et al.*, 2000] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann and M. Klein. OIL in a nutshell. In proc of the 12th EKAW, Juan-Les-Pins, France, LNAI 1937, Springer Verlag, p. 1-16, 2000.

[Fensel *et al.*, 1998] D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: Or How to Enable Intelligent Access to the WWW. In proc. of the 11th International Workshop

on Knowledge Acquisition, Modeling and Management (KAW 1998), Banff, Canada, 1998.
[Heflin *et al.*, 1998] J. Heflin, J. Hendler, S. Luke. Reading between the Lines: Using SHOE to Discover Implicit Knowledge from the Web. In proc. of the AAAI Workshop on Artificial Intelligence and Information Integration. WS-98-14. AAAI Press, p. 51-57, 1998.
[Leclere, 1997] M. Leclere. Reasoning with type definitions. In proc. of the 5th International Conference on Conceptual Structures (ICCS'97), Seattle, WA, USA, Lukose D., Delugach, Keeler M. Searle L. and Sowa J. eds, Lecture Notes in Artificial Intelligence, LNIA 1257, Springer Verlag, p. 401-415, 1997.
[Luke *et al.*, 1997] S. Luke, L. Spector, D. Rager, J. Hendler. Ontology based Web agents, In proc. of the 1st International Conference on Autonomous Agent, 1997.
[Martin and Eklund, 2000] P. Martin, P. Eklund. Conventions for Knowledge Representation via RDF, In proc. of WebNet 2000, San Antonio, Texas, ACCE press, p. 378-383, 2000.
[OIL, 2000] Ontology Inference Layer. http://www.ontoknowledge.org/oil.
[Pierce, 1932] C.S. Pierce. Collected Papers of Charles Sanders Pierce. In HARTSHORNE C. & WEISS P. eds. Harvard University Press, Cambridge, MA, 1932.
[RDF, 1999] Resource Description Framework Model and Syntax Specification, 1999. W3C Recommendation.
[RDFS, 2000] Resource Description Framework Schema Specification, 2000. W3C Candidate Recommendation.
[Salvat and Mugnier, 1996] E. Salvat, M.L. Mugnier. Sound and complete forward and backward chainings of graph rules, In proc. of the 4th International Conference on Conceptual Structures (ICCS'96), Sydney, Australia, Lecture Notes in Artificial Intelligence, LNCS 1115, Springer Verlag, p. 248-262, 1996.
[Sowa, 1984] J.F. Sowa. Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading, Massachusetts, 1984.
[Sowa, 1999] J.F. Sowa. Conceptual Graphs: DpANS. http://concept.cs.uah.edu/CG/Standard.html 1999.

12 Appendix

Appendix 1: RDF Schema of DRDFS

```
<rdf:RDF xml:lang='en'
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
xmlns:drdfs='http://www.inria.fr/acacia/drdfs-schema#' >

<rdfs:Class rdf:ID='DefinedClass'>
    <rdfs:subclassOf rdf: resource=
'http://www.w3.org/2000/01/rdf-schema#Class' />
</drdfs:DefinedClass>

<rdfs:Class rdf:ID='DefinedProperty'>
    <rdfs:subclassOf
resource='http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property' />
</drdfs:DefinedClass>

<rdfs:Class rdf:ID='Context'>
    <rdfs:subclassOf
resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdfs:Class>

<rdfs:Class rdf:ID='Axiom'>
    <rdfs:subclassOf resource='#Context' />
</rdfs:Class>

<rdfs:Class rdf:ID='Variable'>
    <rdfs:subclassOf
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdfs:Class>

<rdf:Property ID='hasDefinition'>
<rdfs:domain rdf:resource='#DefinedRelation' />
      <rdfs:domain rdf:resource='#DefinedConcept' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>

<rdf:Property ID='referent'>
    <rdfs:domain
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
    <rdfs:range
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdf:Property>

<rdf:Property ID='isContextOf'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range
rdf:resource='http://www.w3.org/2000/01/rdf-
schema#Resource' />
</rdf:Property>

<rdf:Property ID='parameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='formalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='firstFormalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='secondFormalParameter'>
    <rdfs:domain rdf:resource='#Context' />
    <rdfs:range rdf:resource='#Variable' />
</rdf:Property>

<rdf:Property ID='if'>
    <rdfs:domain rdf:resource='#Axiom' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>

<rdf:Property ID='if'>
```

```
    <rdfs:domain rdf:resource='#Axiom' />
    <rdfs:range rdf:resource='#Context' />
</rdf:Property>
</rdf:RDF>
```

Appendix 2: Definition of « WebPage » in DRDFS

```
<drdfs:DefinedClass rdf:ID='WebPage'>
    <rdfs:subclassOf rdf:resource='http://…#Document' />
    <drdfs:hasDefinition>
        <drdfs:Context>
            <drdfs:formalParameter rdf:resource='#x' />
            <drdfs:parameter rdf:resource='#y' />
            <drdfs:isContextOf>
                <Document>
                    <drdfs:referent rdf:resource='#x' />
                    <hasForRepresentationSystem>
                        <Format>
                        <drdfs:referent rdf:resource='#y' />
                        </Format>
                    </hasForRepresentationSystem>
                </Document>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#y'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:hasDefinition>
</drdfs:DefinedClass>
```

Appendix 3: Definition of « colleague » in DRDFS

```
<drdfs:DefinedRelation rdf:ID='colleague' >
    <drdfs:hasDefinition>
        <drdfs:Context>
            <drdfs:formalParameter rdf:resource='#x' />
            <drdfs:formalParameter rdf:resource='#y' />
            <drdfs:parameter rdf:resource='#z' />
            <drdfs:isContextOf>
                <Person>
                    <drdfs:referent rdf:resource='#x'/>
                    <worksIn>
                        <Institute>
                            <drdfs:referent rdf:resource='#z'/>
                        </Institute>
                    </worksIn>
                </Person>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <Person>
                    <drdfs:referent rdf:resource='#y'/>
                    <worksIn>
                        <Institute
                            <drdfs:referent rdf:resource='#z'/>
                        </Institute>
                    </worksIn>
                </Person>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#z'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:hasDefinition>
</drdfs:DefinedRelation>
```

Appendix 4: Representation of « If x is a colleague of y, then y is a colleague of x » in DRDFS

```
<drdfs:Axiom>
    <drdfs:formalParameter rdf:resource='#x' />
:   <drdfs:if>
        <drdfs:Context>
            <drdfs:isContextOf>
                <rdf:Description>
                <drdfs:referent rdf:resource='#x'/>
                <colleague>
                        <rdf:Description>
                            <drdfs:referent rdf:resource='#y'/>
                        </rdf:Description>
                    </colleague>
                </rdf:Description>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#y'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
:   </drdfs:if>
:   <drdfs:then>
        <drdfs:Context>
            <drdfs:isContextOf>
                <rdf:Description>
                <drdfs:referent rdf:resource='#y'/>
                <colleague>
                        <rdf:Description>
                            <drdfs:referent rdf:resource='#x'/>
                        </rdf:Description>
                    </colleague>
                </rdf:Description>
            </drdfs:isContextOf>
            <drdfs:isContextOf>
                <rdf:Description>
                    <drdfs:referent rdf:resource='#x'/>
                </rdf:Description>
            </drdfs:isContextOf>
        </drdfs:Context>
    </drdfs:then>
</drdfs:Axiom>
```