Published in U. Priss and D. Corbett and G. Angelova, editors, Proceedings of the 10th International Conference on Conceptual Structures (ICCS'2002), Lecture Notes in Artificial Intelligence, volume 2393, pages 205-219, Borovets, Bulgaria, July 2002. Springer-Verlag.

The Conceptual Graph Formalism as an Ontolingua for Web-Oriented Representation Languages: The RDF Schema Case Study

Olivier Gerbé[1] and Guy W. Mineau[2]

[1] HEC Montreal.
3000, chemin de la Côte-Sainte-Catherine, Montréal, Québec, Canada H3T 2A7
Olivier.Gerbe@hec.ca
[2] Université Laval
Québec, Québec, Canada G1K 7P4
mineau@ift.ulaval.ca

Abstract. The semantic Web entails the standardization of representation mechanisms so that the knowledge contained in a Web document can be retrieved and processed on a semantic level. RDF seems to be the emerging encoding scheme for that purpose. However, there are many different sorts of documents on the Web that do not use RDF as their primary coding scheme. It is expected that many one-to-one mappings between pairs of document representation formalisms will eventually arise. This would create a situation where a young standard such as RDF would generate update problems for all these mappings as it evolves, which is inevitable. Rather, we advocate the use of a common Ontolingua for all these encoding formalisms. Though there may be many knowledge representation formalisms suited for that task, we advocate the use of the conceptual graph formalism.

1 Introduction

The advent of the semantic Web [3] necessarily entails the standardization of representation mechanisms so that the knowledge contained in a Web document can be retrieved and processed on a semantic level. RDF (Resource Description Framework) [25] seems to be the emerging encoding scheme for that purpose. RDF and RDF-S (RDF Schema) [8] define a way to describe Web resources through properties and values that are machine-understandable. RDF-S specifies how to describe RDF vocabularies. The popularity of RDF is bound to grow. However, there are many different sorts of documents that are or could be made available on the Web today that do not use RDF as their primary coding scheme. Let us cite XML (eXtensible Markup Language) for encoding the layout and content of Web pages, UML (Unified modeling Language) [19] diagrams used for conceptual modeling purposes, E-R (Entity-Relationship) [7] diagrams used in the development of database schemas, and soon, trading documents for independent software brokers who will automatically process requests

for different software modules, requests sent by distributed software applications such as agent-based systems. Some of these documents will need to remain in their original encoding format and would not benefit from being translated into a RDF format because: a) their main purpose is better served by the former encoding scheme, b) the expressiveness of RDF may not be sufficient, c) the cost of translating all of these documents, either in processing time and/or storage cost, could be prohibitive, d) the applications that use them would need to be updated, and e) the human intervention required to update them (both applications and documents) would need extensive retraining, which can be costly and in the end, poorly effective. It is expected that many one-to-one mappings between pairs of document representation formalisms will eventually arise like what was done in [10]. This would create a situation where a young standard such as RDF would generate update problems for all these mappings as it evolves, which is inevitable. Rather, we advocate the use of a common Ontolingua for all these encoding formalisms. A mapping then only needs to be from and to this Ontolingua and the target languages: RDF, XML, UML, etc. For instance, translating a UML (class diagram) document into a RDF format would go through that Ontolingua. We believe that, in the long term, system interoperability and flexibility would be best served by such an Ontolingua.

Though there may be many knowledge representation formalisms suited for that task, we advocate the use of the conceptual graph formalism and we demonstrate below that it is a particularly good candidate for that purpose. Tim Berners-Lee compared RDF and conceptual graphs and concluded that Conceptual Graphs are easily integrated with the Semantic Web [2]. Martin and Eklund used CGs to describe and to index Web documents [12]. In effect, the CG formalism:

1. offers a unified and simple representation formalism that covers a wide range of other data and knowledge modeling formalisms,
2. allows matching, transformation, unification and inference operators to process the knowledge that it describes,
3. as a graphical interface to a logic-based system, allows easier interpretation of the knowledge that it encodes,
4. provides for higher representation capabilities such as contexts, modalities, etc., in the same graphical notation as first-order knowledge,
5. is well suited for natural language processing, and is therefore an asset for related applications where the input (textual or annotated documents) is textual, or when the output must be in textual format (e.g., to generate explanations for instance), as is supported by a wealth of literature on the subject [14, 11].

This paper[1] is organized as follows. Section 2 introduces the CG meta-metamodel. Section 3 and 4 present respectively RDF and RDF-S metamodels using CG meta-metamodel and Section 5 illustrates the use of these metamodels. Section 6 presents model transformation metarules and examples. Section 7 reviews related work and Section 8 concludes and discusses further work.

[1] This work is part of a research project supported by HEC Montreal.

2 Conceptual Graphs and Metamodeling

Over the past few years a lot of work has been done on metamodeling [1, 9, 13, 17, 18, 26] but some issues are still debated today: the notion of abstraction layers, their precise role, their relationships, and therefore, their number. In [6] we discussed and argued that we envision three modeling layers, as illustrated in Figure 2: the Meta-Metamodel layer, the Metamodel layer and the Model layer and we shown in [16] that all these layers can be represented under the CG formalism. Figure 1 presents a part of the ontology used to the CG representation.

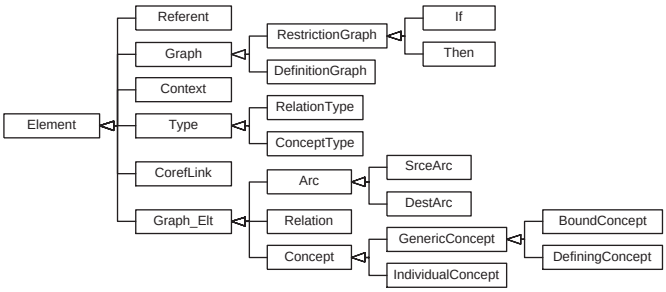

Fig. 1. The CG language type hierarchy.

At the top of the hierarchy are the six basic types of the conceptual graph language: Referent, Graph, Context, Type, CorefLink, and Graph-Elt. Referents (Referent) are internal proxies of the objects of the universe of discourse; graphs (Graph) are the sentences of the language; contexts (Context) allow us to group conceptual graphs; types (Type) are used to categorize referents; co-reference links (CorefLink)associate concepts that represent same elements, and graphs elements (Graph-Elt) that are arcs (Arc), relationships (Relation) and concepts (Concept).

Among concepts we distinguished between individual concepts (IndividualConcept) that represent identified objects and generic concepts (GenericConcept) that represent unidentified objects.

Among graphs we distinguished between definition and restriction graphs. Definition Graphs (DefinitionGraph) are used to define concept types and relation types. Restriction graphs (RestrictionGraph) are graphs that must be always false and that constraint concept types definitions.

Using this language, we can represent different formalisms as shown in Figure 2. At the very top is the meta-metamodel layer, often referenced as M3, where the vocabulary to specify metamodels is defined. In this paper, we will use conceptual graphs to represent metamodels.

The metamodel layer known as M2 defines the set of terms used in M1 level to model the real world. In M2 we can find different metamodels and metarules to transform models from one metamodel to another metamodel. We give as an example in Figure 2 three different metamodels: a metamodel for UML that would define UML Class, UML Object and other UML elements, a metamodel

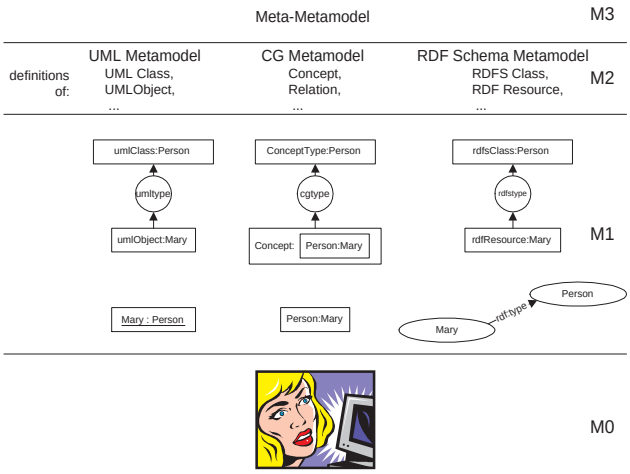

Fig. 2. The three modeling layers and the real world.

for CGs which would include Concept Type, Concept and other CG elements, and a RDF Schema with RDF-S Class, RDF resource and other RDF-S and RDF elements.

Models are defined at the M1 layer. A model is a simplified representation of the real world. In our example we have represented with conceptual graphs the fact that "Mary is a person" using terms defined in M2 under three different metamodels: UML Metamodel, CG Metamodel and RDF Schema Metamodel.

On the left side under the UML Metamodel, the fact that is represented in UML by Mary : Person is represented by Mary is a umlObject that has a relationship umlType with the umlClass Person. The central part of the Figure shows the CG representation. On the right side there is the RDF Schema representation where we expressed that Mary is a rdfResource whose rdfType is the rdfClass Person.

As said in the introduction, in this paper we metamodel RDF Schema elements using conceptual graphs. As a first example, Figures 3 and 4, show the specification of a RDF Schema Class using respectively RDF-S and CGs. And we think that the latter is less confusing. In a nutshell, there has been a lot of work on knowledge representation languages based on semantic networks over the past 30 years and we feel that any graphical specification language (such as RDF-S) should rely on this wealth of expertise in order to come up with a specification formalism that would avoid the pitfalls identified in the literature[24, 20]: the mixing of different abstraction layers in a single expression, the imprecise association of syntactical constructs to semantic roles, and so on.

In specification using CGs, relationships csubt and def are meta-relationships. They belong to the metamodel. The csubt relationship between the concept [ConceptType:RDFSClass] and [ConceptType:RDFSResource] expresses that a RDF-S Class is a kind of RDF-S Resource.

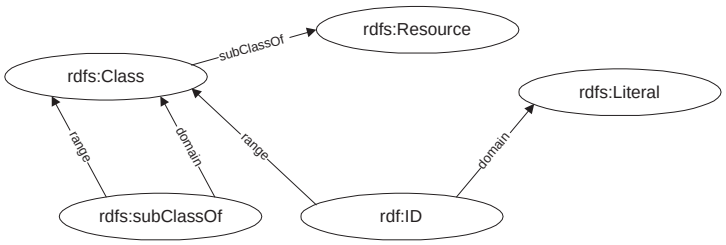

Fig. 3. The definition of the RDF-S Class using RDF-S.

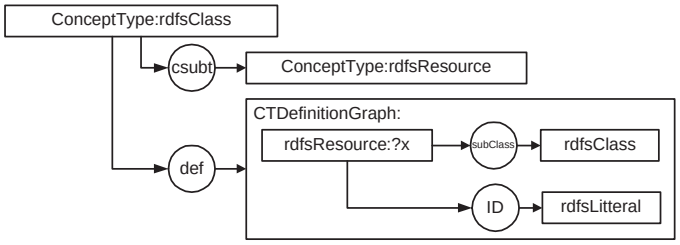

Fig. 4. The definition of the RDF-S Class using CGs.

3 RDF Metamodel

We have seen that we can represent the different levels using CGs. We propose a metamodel for RDF in this section and a metamodel for RDF-S in the next section.

In this section we present the three main elements of RDF: Resource, Property, and Statement, since they form the basis of RDF and are quite sufficient to provide some idea on how other elements of RDF would be represented under the metamodel that we describe in this paper using the CG notation.

3.1 RDF Schema Resource

The main element of RDF is the notion of rdfsResource. rdfsResource is at the top of the class hierarchy and is subclass of itself. Figure 5 presents its CG specification. A rdfsResource has a rdfstype relationship with a rdfsClass, and a rdfslabel relationship with a rdfsLitteral and may also have some literals as comments.

3.2 RDF Schema Property

In a RDF Schema the notions of attribute and relationship are implemented through the unique notion of property. A Property links two classes. One is the class on which the property may be applied (attribute of the class or source of the relationship). The other one is the class in which values may be taken (values of the attribute or target of the relationship). Associated to Property, a

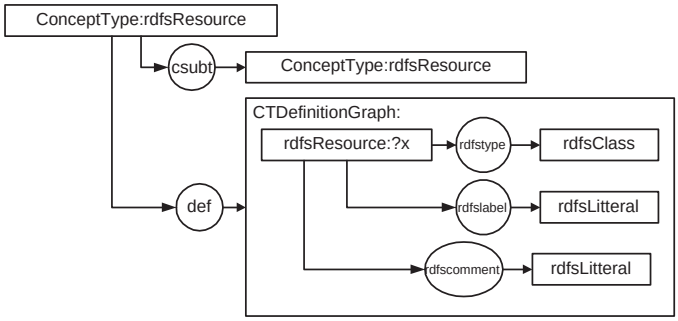

Fig. 5. The rdfsResource specification.

RDF Schema defines two relationships (properties): domain and range. Figure 6 presents the CG representation of the specification of Property.

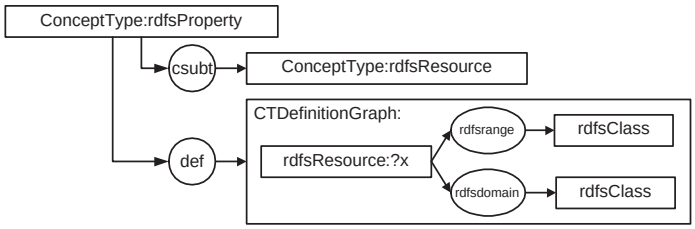

Fig. 6. The Property specification.

3.3 RDF Statement

In RDF, knowledge is represented through statements. A statement is an association between a resource, a property and a value or another resource. Figure 7 illustrates knowledge representation in RDF and its representation using conceptual graphs: a property and a value are linked to the resource they describe. The conceptual relations subject, object and predicate identify roles played by each concept (see 3.4).

3.4 RDF High Order Statement

Sometimes we need to express knowledge about statements. In RDF, statements may be reified. RDF allows the representation of statements about statements, called High Order Statements. Figure 7 shows a high order statement expressed in RDF Syntax and its representation into conceptual graphs. A statement is represented by a resource with four properties. The subject property identifies the described resource. The predicate property identifies the property of the

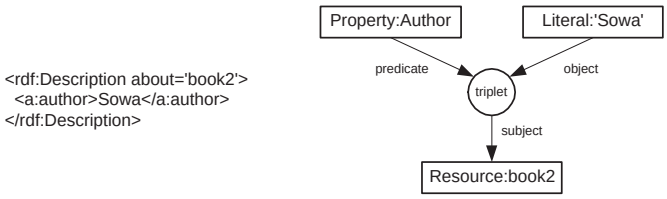

Fig. 7. A RDF Statement.

statement and the object property identifies the value of the property or the resource linked by the property. Contrarily to the previous statement, here using this high order statement, we can state about statements as in "Paul says that the book is authored by Sowa". "The book is authored by Sowa" is expressed as a high order statement that is attributed to Paul.

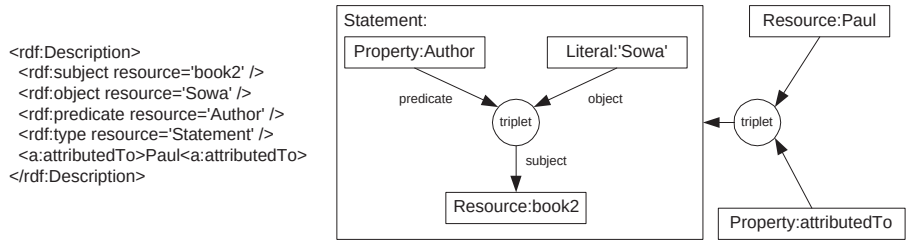

Fig. 8. The RDF High Order Statement.

4 RDF Schema Metamodel

This section presents the main elements of the RDF Schema Metamodel. We will not present all the RDF Schema metamodel elements, but will rather focus on the core classes and properties. Figure 9 presents the type hierarchy of the presented elements.

At the top is the type rdfsResource. All elements of RDF and RDF schemas are considered to be resources. Below we find the two main elements: rdfsClass and rdfsProperty that correspond respectively to CG Concept Type and Relation Type. Constraint Resource is an ad-hoc element used to specify constraints. In particular Constraint Property will be used to implement the constraint properties domain (4.4) and range (4.5).

4.1 RDF Schema Class

A RDF Schema Class corresponds to the abstract notion of Type. As noted in RDF Schema specification [8], this notion is similar to Class in object-oriented

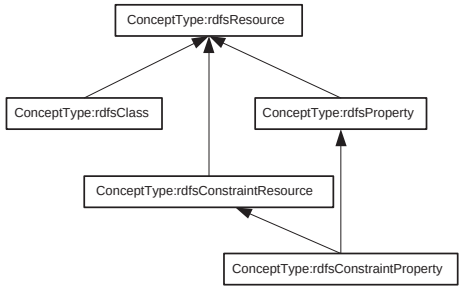

Fig. 9. The RDF Schema Type Hierarchy.

programming languages. This means that class members specification is made at the type level as in UML [19] and not at the data level as in the CG formalism [15]. Figure 10 presents the CG specification representation of rdfsClass. At the metamodel level, a rdfsClass is a kind of rdfsResource so there is a csubt relationship between [ConceptType:rdfsClass] and [ConceptType:rdfsResource]. At the model level, a rdfsClass is a rdfssubClass of another rdfsClass and is identified by an ID.

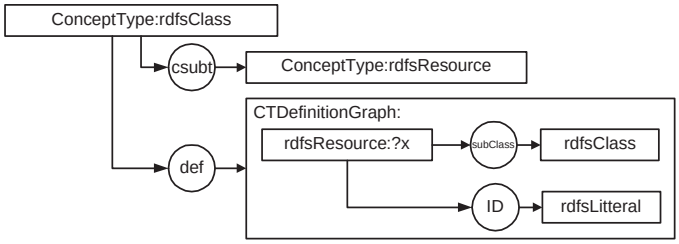

Fig. 10. The Class Specification.

4.2 RDF Schema type Relationship

The RDF-S type relationship is used to indicate that a resource is a member of a class. The relationship links a resource to its class. In a RDF Schema a resource may be linked to more than one class. Figure 11 presents the CG specification of the relationship.

4.3 RDF Schema subClassOf Relationship

The RDF-S subClassOf relationship links a class to its super class. Figure 12 presents the CG specification of the relationship. In a RDF Schema one class may be linked by a subClassOf relationship to more than one class.

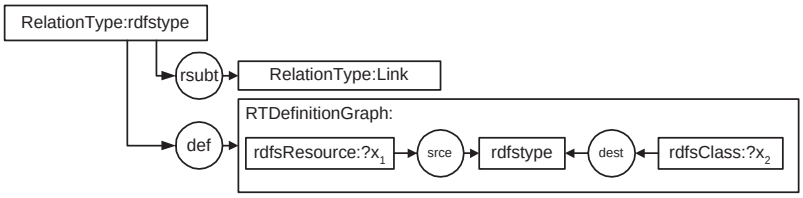

Fig. 11. The RDF-S type relationship specification.

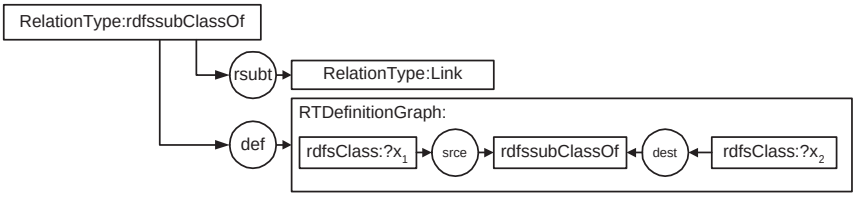

Fig. 12. The RDF-S subClassOf relationship specification.

4.4 RDF Schema domain Relationship

The RDF-S domain relationship links a property to classes whose members can have this property. Figure 13 presents the CG specification of the relationship.

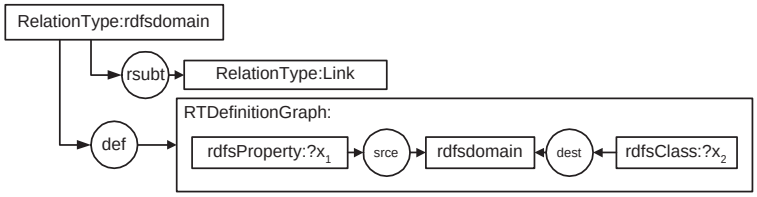

Fig. 13. The RDF-S domain relationship specification.

4.5 RDF Schema range Relationship

The RDF-S range relationship links a property to the class in which the property takes its values. Figure 14 presents the CG specification of this relationship.

We have shown in this section how conceptual graphs can be used to represent the RDF Schema Metamodel. Now in the following section, we will present how to express a RDF-S statement using conceptual graphs.

5 Example of RDF Schema and RDF Representation

Now we have defined RDF and RDF-S metamodel elements, we can represent RDF facts and RDF-S facts in the same formalism. We can have a complete

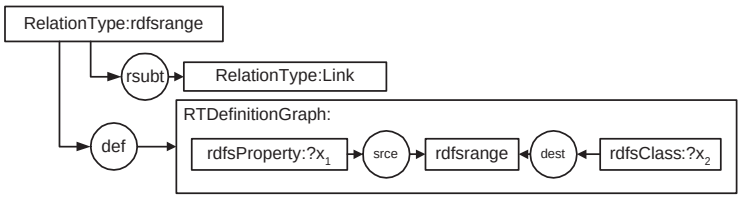

Fig. 14. The RDF-S range relationship specification.

picture of the two levels as illustrated in Figure 15 which adds to Figure 7 the model part. Figure 15 shows a part of the RDF Schema metamodel and the model of the statement. On the upper part of the figure are the main elements of the RDF Schema metamodel. We find the actual model on the lower part of the figure. This figure expresses (through its left part) that the property Author takes its values in the class Litteral and that it may be applied to the class Person. On its right part it expresses the statement itself: the resource Book2 is the subject of the statement, the resource Sowa whose type is Litteral is the object of the statement and the property Author is the predicate of the statement. We added dotted lines between concepts and their concept types to explicitly show relationships between models and metamodels although these relationships are implicitly represented by the name of type in concepts.

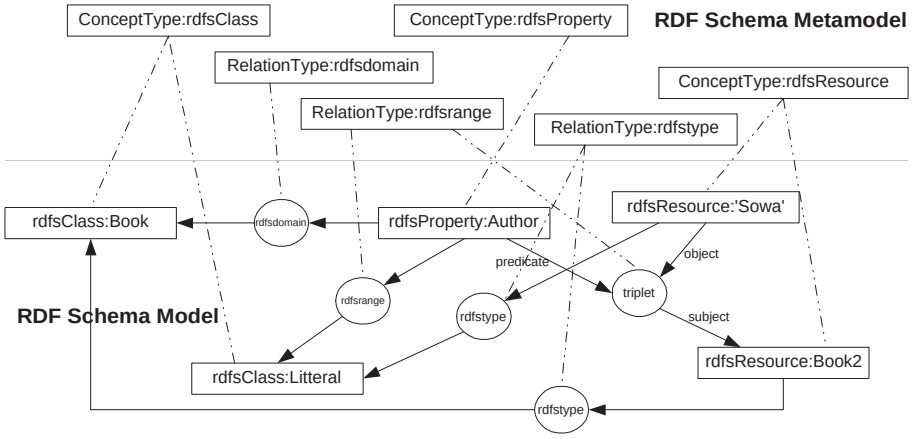

Fig. 15. The representation of "Sowa is the author of the book2".

6 Transformation Rules

For all formalisms we can have a CG representation of their metamodel and we can define transformation rules from one to another[2] as illustrated in this section.

These transformation rules are metarules that map the meta level to the data level. In [23] Sowa describes a mapping between the meta level and the data level. To translate a meta level statement into a data level statement, Sowa introduces two functions τ and ρ. The function τ translates a referent name into a type label. The function ρ has the same behavior as τ on relation types and relations; it translates the name of a relation into a relation type label. In [16] we generalized τ and ρ and defined the function ω as follow:

Definition 1. *The function ω is defined over $\mathcal{C} \rightarrow \mathcal{E}$ where $\mathcal{C}$ is the set of concepts that represent entities of the system and $\mathcal{E}$ is the set of all referenced elements (internal and external elements).*

Applied on a concept the function ω returns the entity represented by the concept. Obviously, the function is defined on the set of concepts that represent entities of the system.

ω([Graph : [Cat]->(on)->[Mat]]) = [Cat]->(on)->[Mat]
ω([ConceptType : *t]) abbreviated in ωt returns the type label t

We show here how we can use CG Metamodeling and metarules to transform models from one formalism to another[3]. To illustrate these transformation rules we present three of them.

The first rule may be applied to transform a RDF Resource into its corresponding conceptual graph model. A RDF Resource corresponds to a CG concept. The rule states that if there exists a RDF Resource x instance of a RDF-S Class y then there is a concept whose type is y and referent is x. Figure 16 presents this metarule.

The second rule may be applied to transform a RDF Schema model with constraints into its corresponding conceptual graph model. As we said before, constraints are not always defined in the same way in RDF Schema and in conceptual graphs. In a RDF Schema constraints are stated between classes and properties. In the conceptual graphs formalism some constraints are stated between concept types but others are stated in concept type definition graphs. Figure 17 presents the metarule.

Constraints on RDF-S classes like rdfssubClassOf are transformed into constraints on concept types. The rdfssubClassOf property between classes is transformed into the csubt relationship between concept types. But constraints like

[2] If the expressiveness of the two formalisms is equivalent. If it is not, there is only a mapping of a subset of constructs of the most expressive formalism to the least expressive one.

[3] As long as metarules do not use functional symbols and do not introduce new elements, one can have a direct mapping between equivalent constructs and rules are not recursive.

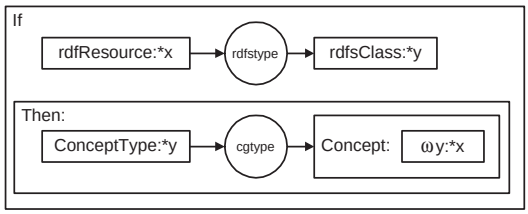

Fig. 16. A RDF Resource is translated in a concept.

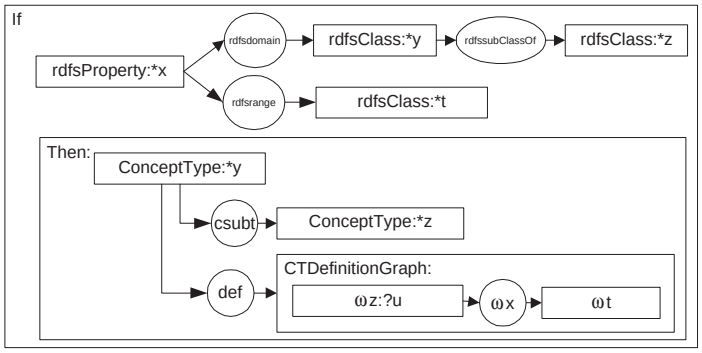

Fig. 17. A RDF Schema Constraint is transformed into a Concept Type Definition.

rdfsrange and rdfsdomain are transformed into constraints between concepts. A RDF-S property `x` having a class `y` for range and a class `t` for domain may be transformed into the fact that any concept of type `t` has a relationship of type `x` with a concept of type `y`.

The third rule illustrates the transformation in the reverse way from a CG model to a RDF Schema model. The concept type definition is transformed in a set of RDF-S constraints. Figure 18 shows the metarule and the Figure 19 presents its application.

Applying metarules presented above we can translate a Concept Type Definition into RDF Schema Constraints and vice-versa as illustrated in Figure 19

7 Related Work

Few researchers has worked on model transformation and metamodeling. Revault and al. propose in [4] a bridge to translate UML-based models to different formalisms. They use a meta-metamodel called PIR3 that is itself an extension of IR3 [22]. Instead of metarules, they use a reduction/normalization algorithm that transforms a UML-based model to a set of constructs; each construct corresponding to a construct of PIR3. Bézivin [5] and Lemesle [21] have the same approach of ours. They use metarules to transform a model with one formalism to another model with a different formalism. Using a first set of metarules they

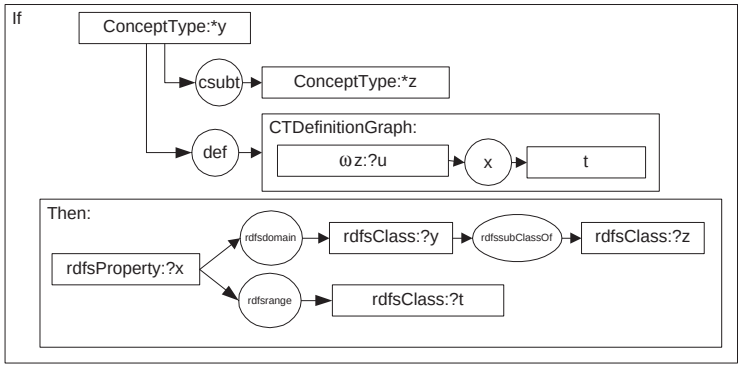

Fig. 18. A Concept Type Definition is transformed into a RDF Schema Constraint.

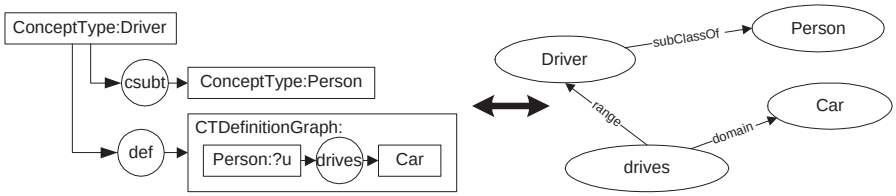

Fig. 19. Transformation example.

translate the source model in sNets, then from this sNets representation they generate a model in the target formalism.

8 Conclusion

This article advocates the use of the CG formalism as an Ontolingua for allowing the automatic translation of knowledge structures from one knowledge representation formalism to the next, thus improving interoperability between these formalisms (especially for those used in Web-oriented applications). In [16] we showed that the CG formalism was an appropriate candidate for such a purpose. Its expressiveness allowed the representation of the various levels of abstraction needed to model some application domain: the data model (M1), its metamodel (M2) and its meta-metamodel (M3). By providing a CG representation of metamodels of different formalisms, we therefore enable a knowledge engineer to write transformation rules that will translate statements provided in a source formalism to a target formalism. Using RDF and RDF-S, we illustrated our proposed methodology. We described the metamodel of RDF and RDF-S using CGs, and showed how statements in RDF would be translatable to other formalisms. Of course our prototype is far from being complete. We need to extend the metamodel under construction so that we cover all aspects of RDF and RDF-S. And we need to implement an analyzer that will convert RDF statements into their CG representation using that metamodel. We also

plan on developing the (CG) metamodel of other formalisms (such as UML). Our ultimate goal is to provide reasoning capabilities on knowledge structures encoded in various documents on the WWW. Improving the interoperability of the knowledge structures that each document contain is therefore relevant. Syntax related considerations are a first and necessary step in that direction. Of course, other considerations such a semantic interpretation of these extracted knowledge structures is also a research issue that is on our agenda; forth-coming papers on the subject will soon present that aspect of our research program.

References

[1] C. Atkinson and T. Kühne. The Essence of Multilevel Metamodeling. In *Proceedings of UML'2001 Conference on Modeling Languages, Concepts and Tools*, Toronto, Ontario, Canada, October 1-5 2001.

[2] T. Berners-Lee. Conceptual Graphs and the Semantic Web. February 2001. available at http://www.w3.org/DesignIssues/CG.html.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[4] X. Blanc, J.F. Perrot, and N. Reveault. Traduction de méta-modèles. In I. Borne and R. Godin, editors, *Langages et Modèles à Objets*, pages 95–111, Le Croisic, France, Janvier 2001. Hermès Science Publications.

[5] J. Bézivin. Objects Everywhere. In *Proceedings of ICEIS*, Setùbal, Portugal, July 2001. Invited Presentation.

[6] J. Bézivin and O. Gerbé. Towards a Precise Definition of the OMG/MDA Frame-Work. In *Proceedings of the 16th Conference on Automated Software Engineering*, pages 273–280, San Diego, USA, November 2001. IEEE Computer Society Press.

[7] P. Chen. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[8] World Wide Web Consortium. *Resource Description Framework (RDF) Schema Specification 1.0*, March 2000.

[9] S. Crawley, S. Davis, J. Indulska, S. McBride, and K. Raymond. Meta-Meta is Better-Better. October 1997.

[10] A. Delteil, R. Dieng, and C. Faron-Zucker. Extension of RDFS Based on the CGs Formalism. In H. Delugach and G. Stumme, editors, *Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001*, pages 275–289, Stanford, CA, USA, July/August 2001. Springer Verlag.

[11] H. Delugach and G. Stumme, editors. *Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001*. Springer Verlag, Stanford, CA, USA, July/August 2001.

[12] P. Eklund and P. Martin. Embedding Knowledge in Web Documents: CGs versus XML-based Metadata Languages. In W. Cyre and W. Tepfenhart, editors, *Proceedings of the 7th International Conference on Conceptual Structures, ICCS 1999*, pages 230–246, Blacksburg, VA, USA, July 1999. Springer Verlag.

[13] J. Esch. Contexts, Canons and Coreferent Types. In J. Dick, J. Sowa, and W. Tepfenhart, editors, *Proceedings of the Second International Conference on Conceptual Structures (ICCS*, pages 185–195, College Park, Maryland, USA, August 1994. Springer Verlag.

[14] B. Ganter and G. Mineau, editors. *Proceedings of the 8th International Conference on Conceptual Structures, ICCS 2000*. Springer Verlag, Darmstadt, Germany, August 2000.

[15] O. Gerbé. Conceptual Graphs for Corporate Knowledge Repositories. In H. Delugach, M. Keeler, D. Lukose, L. Searle, and J. Sowa, editors, *Proceedings of the 5th International Conference on Conceptual Structures (ICCS*, pages 474–488, Seattle, Washington, USA, August 1997. Springer Verlag.

[16] O. Gerbé. *Un modèle uniforme pour la modélisation et la métamodélisation d"une mémoire d"entreprise*. PhD thesis, Université de Montréal, Avril 2000.

[17] O. Gerbé and B. Kerhervé. Modeling and Metamodeling Requirements for Knowledge Management. In J. Bézivin, J. Ernst, and W. Pidcock, editors, *Proceedings of OOPSLA Workshop on Model Engineering with CDIF*, Vancouver, Canada, October 1998.

[18] Object Management Group. *Meta Object Facility (MOF) Specification*, September 1997. OMG Document AD/97-08-14.

[19] Object Management Group. *Unified Modeling Language Specification*, Jume 1999. OMG Document AD/99-06-08.

[20] P. Kocura. Semantics of Attribute Relations in Conceptual Graphs. In B. Ganter and G. Mineau, editors, *Proceedings of the 8th International Conference on Conceptual Structures, ICCS 2000*, pages 235–248, Darmstadt, Germany, August 2000. Springer Verlag.

[21] R. Lemesle. Transformation Rules Based on Metamodeling. In *Proceedings of Second International Enterprise Distributed Object Computing Workshop (EDOC*, pages 113–122, La Jolla, CA, November 1998.

[22] H. Sahraoui. *Application de la méta-modélisation à la génération des outils de conception et de mise en oeuvre des bases de données*. PhD thesis, 1995.

[23] J. Sowa. Relating diagrams to logic. In John F. Sowa Guy W. Mineau, Bernard Moulin, editor, *Proceedings of the First International Conference on Conceptual Graphs (ICCS'93)*, volume 699, pages 1–35, Quebec City, Quebec, Canada, August 1993. Springer-Verlag.

[24] J. Sowa. Ontology, Matadata, and Semiotics. In B. Ganter and G. Mineau, editors, *Proceedings of the 8th International Conference on Conceptual Structures, ICCS 2000*, pages 55–81, Darmstadt, Germany, August 2000. Springer Verlag.

[25] W3C. *Resource Description Framework (RDF) Model and Syntax Specification*, February 1999.

[26] M. Wermelinguer. Conceptual Graphs and First Order Logic. In G. Ellis, R. Levinson, and W. Rich, editors, *Proceedings of the Third International Conference on Conceptual Structures*, pages 323–337, Santa Cruz, CA, USA, August 1995. Springer Verlag.