Validation of XML Documents with SWRL

Jesús M. Almendros-Jiménez

Validation of XML Documents with SWRL *

Jesús M. Almendros-Jiménez

Dpto. de Lenguajes y Computación.
Universidad de Almería. Spain. Email: jalmen@ual.es

Abstract. In this paper we describe how XML documents are mapped into an OWL ontology and how SWRL rules are used to validate the semantic content of XML documents. XML completion and data constraints are specified with SWRL. The semantic completion of the XML document can be mapped into a semantic completion of the corresponding ontology. Besides, SWRL serves for specifying and reasoning with data constraints. We will illustrate our approach with an example that shows that user intervention is vital to XML mapping and completion and SWRL helps to detect relevant data constraints. The approach has been tested with the well-known Protégé tool.

1 Introduction

In the database scientific community many efforts have been achieved to give semantic content to data sources. The entity-relationship (ER) model, functional dependences and integrity constraints are key elements of database design. ER model gives semantics to data relations and restrict cardinality of relations. Functional dependences establish data dependences and key values. Integrity constraints impose restrictions over data values in a certain domain.

The *eXtensible Markup Language (XML)* [W3C07] is equipped with data definition languages (*DTD* [W3C99] and *XML Schema* [W3C09c]) whose aim is to describe the syntactic structure of XML documents. Well-formed XML documents conform to the corresponding DTD and XML Schema. However, even when XML documents can be well-formed, the content can be redundant which can lead to inconsistency, as well as the content can violate imposed restrictions over data.

Most available XML resources lack in the description of semantic content. But it also happens for syntactic content: DTD and XML schemas might not available in Web resources. It makes that data exchange fails due to bad understanding of XML content. Nevertheless, XML records can follow the same pattern and the user could help to discover and recover the semantic content of XML resources. This task is not easy and should be supported by tools. Such tools should be able to analyze the semantic content of XML data revealing fails on interpretation.

* This work has been supported by the Spanish Ministry MICINN under grant TIN2008-06622-C03-03, Ingenieros Alborada IDI under grant TRA2009-0309, and the JUNTA de ANDALUCÍA (proyecto de excelencia) ref. TIC-6114.

In order to help tools in the interpretation, the user could provide the intended meaning of resources.

RDF(S) [KC04] and OWL [MPSP+08] emerge as solutions for equipping Web data with semantic content. Unfortunately, most of database management systems do not offer exporting facilities to RDF(S) and OWL, although some efforts have been carried out (see [KSM08] for a survey). XML has been usually adopted as database exchange format. For this reason, some authors have described how to map XML into RDF(S)/OWL and add semantic content to XML documents.

On one hand, most proposals focus on the mapping of the XML schema into RDF(S)/OWL. In some cases, the mapping is exploited for reasoning (conformance and completion, among others). On the other hand, in many cases, the mapping, when the XML schema is present, can be automatically accomplished. Nevertheless, some of them work when the XML schema is missing, requiring the user intervention, who specifies a set of mapping rules. Finally, most of cases tools have been developed with this end, based on XML transformation and query languages like XSLT and XPath.

In this paper we describe how XML documents are mapped into an OWL ontology and how SWRL rules are used to validate the semantic content of XML documents. SWRL rules enable to express constraints on XML data, and they can be triggered in order to validate the constraints. The approach has been tested with the well-known Protégé tool.

We can summarize the main contributions of our proposal as follows:

- XML into OWL mapping is carried out by specifying mappings from tags and attributes into concepts, roles and individuals of the ontology. Such mappings are defined with XPath. The ontology is created from the mapping, firstly, at instance level and, secondly, by adding ontology axioms with SWRL.
- SWRL is used for two main tasks:
  (a) to add new semantic information to the ontology instance generated from the XML document. Such information can be considered as a completion of the XML model. In particular, such rules can create additional classes and roles, and therefore extending the ontology initially created from the mapping with new axioms and instance relations. In other words, SWRL serves as ontology definition language in the phase of completion; and
  (b) to express data constraints on the XML document. SWRL can be used for expressing relations that the ontology instance has to satisfy. Therefore SWRL is a vehicle for reasoning with the semantic content and therefore for analyzing XML resources.

Our approach aims to provide a method for specifying a transformation rather than to consider automatic mapping from the XML Schema. XML completion and data constraints are specified with SWRL. The semantic completion of the XML document can be mapped into a semantic completion of the corresponding ontology. Besides, SWRL serves for specifying and reasoning with data constraints. We will illustrate our approach with an example that shows

that user intervention is vital to XML mapping and completion and SWRL helps to detect relevant data constraints.

The drawbacks of our approach are the following. The kind of ontology we can handle is limited to SWRL expressivity. Since we create the target ontology with SWRL, OWL meta-data axioms are defined with SWRL rules. In other words, SWRL is taken as ontology definition language in the line of [KMH11,KRH08]. However, therefore there are some completions/data constraints that cannot be specified. In particular, those involving universally quantified constraints cannot be specified.

One feasible extension of our work is to move to SQWRL [OD09], which extends SWRL to a more powerful query language. It is considered as future work. With regard to the choice of SWRL as ontology definition language, we could express completions/data constraints with a suitable OWL 2 fragment (for instance, EL, RL and QL) [W3C09a,W3C09b]. However, in such a case, we would also have a limited expressivity.

1.1 Related Work

XML Schema mapping into RDF(S)/OWL has been extensively studied.

In an early work [FZT04] XML Schema is mapped into RDF(S) meta-data and individuals. In [BA05], they propose an small set of mapping rules from XML Schema into OWL and define a transformation with XSLT. In [TC07], they also propose an XSLT based transformation from XML Schema to OWL that allows the inverse transformation, i.e. convert individuals from OWL to XML. The inverse transformation is usually called *lowering*, and the direct one is called *lifting*. Besides, in [XC06], XML schemas are mapped into RDF(S) and they make use of the mapping for query processing in the context of data integration and interoperation. This is also the case of [BMPS$^+$11], in which the authors propose a set of patterns to automatically transform an XML schema to OWL. Such patterns are obtained from pattern recognition. Manual mapping has been considered in [TLLJ08], in which the authors translate XML schemas and data to RDF(S) with user intervention.

In some cases the target ontology has to be present, that is, the mapping from XML into OWL aims to populate the ontology with individuals. For instance, in the proposals [RRC08,AKKP08,RRC06,VDPM$^+$08], the authors transform XML resources to RDF(S) and OWL format, in the presence of an existing ontology and with user intervention. The mapping rules can be specified with Java (for instance, in [RRC08]), and with domain specific languages (for instance, with XSPARQL [AKKP08]).

Manual mappings are in many cases based on XPath expressions. In [GC09], they describe how to map XML documents into an ontology, using XPath for expressing the location in the XML document of OWL concepts and roles. In [OD11] they employs XPath to map XML documents into OWL, extending the set of OWL constructors to represent XML resources. Besides XPath is used to describe mappings in the XSPARQL framework [AKKP08].

Finally, some authors have explored how to exploit the mapping for XML validation. For instance, in [WRC08], they describe how to map the XML Schema into Description Logic (DL) and they make use of DL for model validation and completion and as query language. Besides, in [ZYMC11], they map XML into OWL (and DL) and they study how to reason about the XML schema, in the sense of that, to check conformance of XML documents, and to prove inclusion, equivalence and disjointness of XML schemas. SWRL has been employed in [LSD+09] for data normalization, encoding schemas and functional dependences with DL.

Comparing our work with existent proposals we find some similarities with the work described in [OD11] in which they employ XPath for manual mapping and a rich OWL fragment for describing concept and role relations.

1.2 Structure of the Paper

The structure of the paper is as follows. Section 2 will present a running example. Section 3 will describe how to map XML documents into OWL. Section 4 will focus on how to complete the semantic content of XML documents. Section 5 will show how to validate XML constraints and, finally, Section 6 will conclude and present future work.

2 Running example

Let us suppose that we have the XML resource of Figure 1. The document lists *paper*s and *researcher*s involved in a *conference*. Each *paper* and *researcher* has an identifier (represented by the attribute *id*), and has an associated set of labels: *title* and *wordCount* for *paper*s and *name* for *researcher*s. Furthermore, they have attributes: *studentPaper* for *paper*s and *isStudent*, *manuscript* and *referee* for *researcher*s. The meaning of the attributes *manuscript* and *referee* is that the given researcher has submitted the paper of number described by *manuscript* as well as (s)he has participated as reviewer of the paper of number given by *referee*.

It is worth observing that the document uses identifiers for cross references between papers and researches. It is just for simplifying the example, and it is not real restriction of our approach. It does not mean that interesting examples come from resources where cross references are given.

Now, let us suppose that we would like to analyze the semantic content of the XML document. We would like to know whether some paper violates the restriction on the number of words of submissions. In order to do this we could execute the following XPath query:

```
/conference/papers/paper[wordCount>10000]
```

and it gives us the papers whose attribute *wordCount* is greater than 10000. This is a typical restriction that can be analyzed with XPath.

Fig. 1. Running Example

```
<?xml version='1.0'?>
<conference>
<papers>
<paper id="1" studentPaper="true">
<title> XML Schemas </title>
<wordCount> 1200 </wordCount>
</paper>
<paper id="2"  studentPaper="false">
<title> XML and OWL </title>
<wordCount> 2800 </wordCount>
</paper>
<paper id="3" studentPaper="true">
<title> OWL and RDF </title>
<wordCount> 12000 </wordCount>
</paper>
</papers>
<researchers>
<researcher id="a" isStudent="false" manuscript="1"
        referee="1">
<name>Smith </name>
</researcher>
<researcher id="b" isStudent="true" manuscript="1"
        referee="2">
<name>Douglas </name>
</researcher>
<researcher id="c" isStudent="false" manuscript="2"
        referee="3">
<name>King </name>
</researcher>
<researcher id="d" isStudent="true" manuscript="2"
        referee="1">
<name>Ben</name>
</researcher>
<researcher id="e" isStudent="false" manuscript="3"
        referee="3">
<name>William</name>
</researcher>
</researchers>
</conference>
```

However, we can complicate the situation when papers are classified as student and senior papers which have different restrictions of length. Fortunately, each paper has been labeled with this information, that is, with the attribute *studentPaper*. However, it is redundant in the document. That is, we have information about submitters and whether they are student or not. In the case

papers are not labeled with the attribute *studentPaper*, the XPath query becomes more complex. In general, missing and redundant information in XML resources makes XPath based analysis more complex to make.

The goal of our approach is to be able to extract from the XML document the semantic content and validate the content. In particular, it involves to analyze cross references. Besides, the process of extraction is guided by the user who knows (or at least (s)he can suspect) the meaning of the given labels and attributes.

In XML Schema based translations, such semantic information could not be specified in the document therefore the user intervention is also required. Semantic validation of XML documents ranges from restrictions imposed over data such as "the attribute *wordCount* has to be smaller than 10000" to properly integrity constraints as "the reviewer of a paper cannot be the submitter". Assuming that senior papers cannot have students as authors, inconsistent information comes from senior papers having an student as author.

Validation can be improved by completing the XML model. Completion means to add new information that can be deduced from the original resource.

For instance, in the running example, we can add the *author* for each *paper*, which can be obtained as the inverse of the value of the *manuscript* attribute of each *researcher*. In the case the attribute *studentPaper* was not included, we can add this information from the information about *researcher*s using the *isStudent* attribute. Besides, the semantic extraction can be more accurate. For instance, we can define the classes *Student*, *Senior* as subclasses of *Researcher*, and *PaperofSenior* and *PaperofStudent* as subclasses of *Paper*. *PaperofSenior* class is defined as the subclass of papers whose value *studentPaper* is false and *Student* can be defined as the subclass of researchers whose value *isStudent* is true. We can also define the *Reviewed* class as the subclass of papers which have at least one *referee*.

The definition of such ontology based completion facilitates the description of data constraints. For instance, *author*s of *PaperofSenior* cannot be *Student*s. In order to express data constraints we have adopted a simple solution in our approach. We will define new ontology classes that represent data constraints. For instance, the class *BadPaperCategory* can be defined as the class of senior papers having an student as author, while the class *NoSelfReview* can be defined as the class of papers having a referee which is also the author of the paper. When the classes *BadPaperCategory* and *NoSelfReview* are not empty, the XML document violates the required constraints.

Our approach has in the spirit to provide a methodology for XML validation. Our proposal distinguishes three steps: mapping, completion and validation. The following sections will describe each one of these steps using the running example.

3 Mapping XML into OWL

The first step consists in the XML into OWL mapping with rules. Mapping rules establish a correspondence from XPath expressions to ontology concepts:

$$xp_1 \mapsto C_1, \ldots, xp_n \mapsto C_n$$

and from pairs of XPath expressions to ontology roles:

$$(xp'_1, xp''_1) \mapsto r_1 \ldots, (xp'_m, xp''_m) \mapsto r_m$$

Mapping works at the instance level, creating instances of concepts and roles from XML items. Concepts and roles belong to the target ontology.

For instance, let us suppose that the programmer wants to transform the running example. The programmer, firstly, has to define paths to access to individuals and property values:

```
(a) doc('papers.xml')//papers/researcher/@id
(b) doc('papers.xml')//papers/researcher/name
(c) doc('papers.xml')//papers/researcher/isStudent
(d) doc('papers.xml')//papers/researcher/@manuscript
(e) doc('papers.xml')//papers/researcher/@referee
```

and, secondly, has to map them into ontology concepts and roles as follows: $\mathsf{a} \mapsto \mathsf{Researcher}$, $(\mathsf{a}, \mathsf{b}) \mapsto \mathsf{name}$, $(\mathsf{a}, \mathsf{c}) \mapsto \mathsf{isStudent}$, $(\mathsf{a}, \mathsf{d}) \mapsto \mathsf{manuscript}$ and $(\mathsf{a}, \mathsf{e}) \mapsto \mathsf{referee}$. The same can be done from papers:

```
(f) doc('papers.xml')//papers/paper/@id
(g) doc('papers.xml')//papers/paper/title
(h) doc('papers.xml')//papers/paper/wordCount
(i) doc('papers.xml')//papers/paper/@StudentPaper
```

in which the mapping is as follows: $\mathsf{f} \mapsto \mathsf{Paper}$, $(\mathsf{f}, \mathsf{g}) \mapsto \mathsf{title}$, $(\mathsf{f}, \mathsf{h}) \mapsto \mathsf{wordCount}$ and $(\mathsf{f}, \mathsf{i}) \mapsto \mathsf{StudentPaper}$.

The mapping obtains and ontology for researchers (see Figure 2) and an ontology for papers (see Figure 3).

4 Semantic Completion

The second step consists in the XML completion using SWRL rules. The completion is defined in terms of the target OWL ontology.

SWRL rules are used to define new concepts and roles $C'_1, \ldots, C'_s$, $r'_1, \ldots, r'_l$ from $C_1, \ldots, C_n$ and $r_1, \ldots, r_m$. Completion aims to structure the semantic information and to infer new information.

SWRL allows to express "and" conditions by means of "$\wedge$", and OWL classes and properties can be used as atoms in the antecedent and the consequent: $C(?x)$ means that $?x$ is an individual of the class C, and $P(?x, ?y)$ means that the property P holds for $?x$ and $?y$, where $?x$ and $?y$ are variables that in SWRL starts with "?". Moreover, SWRL admits the use of the built-ins *"greaterThanOrEqual"* and *"lessThan"* whose role is to restrict the numeric value of the variables.

For instance, let us suppose that the programmer wants to specify the completion of the running example by defining the concepts *PaperofSenior*, *PaperofStudent* and *Reviewed* as subclasses of the concept *Paper*:

```
<rdf:RDF>
<owl:Class rdf:about="#Researcher"/>
<rdf:Property rdf:about="#manuscript"/>
<rdf:Property rdf:about="#referee"/>
<owl:DatatypeProperty rdf:about="#name"/>
<owl:DatatypeProperty rdf:about="#isStudent"/>
 <owl:Thing rdf:about="#a">
    <rdf:type rdf:resource="#Researcher"/>
    <name>Smith</name>
    <isStudent>false</isStudent>
    <manuscript rdf:resource="#1"/>
    <referee rdf:resource="#1"/>
 </owl:Thing>
 <owl:Thing rdf:about="#b">
    <rdf:type rdf:resource="#Researcher"/>
    <name>Douglas</name>
    <isStudent>true</isStudent>
    <manuscript rdf:resource="#1"/>
    <referee rdf:resource="#2"/>
 </owl:Thing>
 <owl:Thing rdf:about="#c">
    <rdf:type rdf:resource="#Researcher"/>
    <name>King</name>
    <isStudent>false</isStudent>
    <manuscript rdf:resource="#2"/>
    <referee rdf:resource="#3"/>
 </owl:Thing>
 </rdf:RDF>
```

Fig. 2. Ontology for researchers

```
PaperofSenior(?x) -> Paper(?x)
Reviewed(?x) -> Paper(?x)
studentPaper(?x,false) -> PaperofSenior(?x)
studentPaper(?x,true) -> PaperofStudent(?x)
referee(?x,?y) -> Reviewed(?x)
```

Moreover, (s)he defines the inverse relations of *manuscript* and *referee* (defined as *author* and *submission*), and the classes *Student* and *Senior* as subclasses of *Researcher* as follows:

```
manuscript(?x,?y) -> author(?y,?x)
referee(?x,?y) -> submission(?y,?x)
isStudent(?x,true) -> Student(?x)
isStudent(?x,false) -> Senior(?x)
Student(?x) -> Researcher(?x)
Senior(?x) -> Researcher(?x)
```

```
<rdf:RDF>
<owl:Class rdf:about="#Paper"/>
<owl:DatatypeProperty rdf:about="#studentPaper"/>
<owl:DatatypeProperty rdf:about="#title"/>
<owl:DatatypeProperty rdf:about="#wordCount"/>
 <owl:Thing rdf:about="#1">
    <rdf:type rdf:resource="#Paper"/>
    <studentPaper>true</studentPaper>
    <title>XML Schemas</title>
    <wordCount>1200</wordCount>
 </owl:Thing>
 <owl:Thing rdf:about="#2">
    <rdf:type rdf:resource="#Paper"/>
    <studentPaper>false</studentPaper>
    <title>XML and OWL</title>
    <wordCount>2800</wordCount>
 </owl:Thing>
 <owl:Thing rdf:about="#3">
    <rdf:type rdf:resource="#Paper"/>
    <studentPaper>true</studentPaper>
    <title>OWL and RDF</title>
    <wordCount>12000</wordCount>
 </owl:Thing>
</rdf:RDF>
```

Fig. 3. Ontology for papers

Let us remark that SWRL is used for describing meta-data relationships, and some of them correspond to OWL 2 relationships. For instance, they can be expressed as

$$\mathsf{PaperofSenior} \sqsubseteq \mathsf{Paper}$$

$$\exists \mathsf{studentPaper}.\{\mathsf{false}\} \sqsubseteq \mathsf{PaperofSenior}$$

$$\exists \mathsf{referee}.\top \sqsubseteq \mathsf{Reviewed}$$

However, we make use of SWRL as ontology definition language for expressing ontology relationships.
The semantic completion will obtain the ontologies of Figures 4 and 5. We have classified researchers and papers as seniors and students, and papers as papers of students and seniors, and reviewed.

5 Validation

The last step consists in the definition of data constraints with SWRL. New concepts are defined from $C_1, \ldots, C_n$, $C'_1, \ldots, C'_s$, $r_1, \ldots, r_m$ and $r'_1, \ldots, r'_l$, and individuals of such concepts violate data constraints.

```
<rdf:RDF>
    <owl:Thing rdf:about="#a">
      <submission rdf:resource="#1"/>
      <rdf:type rdf:resource="#Senior"/>
    </owl:Thing>
    <owl:Thing rdf:about="#b">
      <submission rdf:resource="#2"/>
      <rdf:type rdf:resource="#Student"/>
    </owl:Thing>
    <owl:Thing rdf:about="#c">
      <submission rdf:resource="#3"/>
      <rdf:type rdf:resource="#Senior"/>
    </owl:Thing>
    <owl:Thing rdf:about="#d">
      <submission rdf:resource="#1"/>
      <rdf:type rdf:resource="#Student"/>
    </owl:Thing>
    <owl:Thing rdf:about="#e">
      <submission rdf:resource="#3"/>
      <rdf:type rdf:resource="#Senior"/>
    </owl:Thing>
</rdf:RDF>
```

Fig. 4. Completion of researchers

```
<rdf:RDF>
<owl:Thing  rdf:about="#1">
      <rdf:type rdf:resource="#PaperofStudent"/>
      <rdf:type rdf:resource="#Reviewed"/>
      <author rdf:resource="#a"/>
      <author rdf:resource="#b"/>
    </owl:Thing>
    <owl:Thing rdf:about="#2">
      <rdf:type rdf:resource="#PaperofSenior"/>
      <rdf:type rdf:resource="#Reviewed"/>
      <author rdf:resource="#c"/>
      <author rdf:resource="#d"/>
    </owl:Thing>
    <owl:Thing rdf:about="#3">
      <rdf:type rdf:resource="#PaperofStudent"/>
      <rdf:type rdf:resource="#Reviewed"/>
      <author rdf:resource="#e"/>
    </owl:Thing>
</rdf:RDF>
```

Fig. 5. Completion of papers

For instance, in the running example, the programmer can define the concepts *PaperLength*, *NoSelfReview*, *NoStudentReview* and *BadPaperCategory* with the following rules:

```
wordCount(?x,?y) ^ greaterThanOrEqual(?y,10000)
                          -> PaperLength(?x)
manuscript(?x,?y) ^ submission(?x,?y)
                          -> NoSelfReview(?x)
Student(?x) ^ submission(?x,?y)
                          -> NoStudentReviewer(?x)
manuscript(?x,?y) ^ Student(?x)
    ^ PaperofSenior(?y)   -> BadPaperCategory(?x)
```

Finally, they can be triggered, obtaining the following results:

```
<result>
    <owl:Thing  rdf:about="#3">
      <rdf:type rdf:resource="#PaperLength"/>
    </owl:Thing>
    <owl:Thing   rdf:about="#a">
      <rdf:type rdf:resource="#NoSelfReview"/>
    </owl:Thing>
    <owl:Thing  rdf:about="#e">
      <rdf:type rdf:resource="#NoSelfReview"/>
    </owl:Thing>
    <owl:Thing   rdf:about="#b">
      <rdf:type rdf:resource="#NoStudentReviewer"/>
    </owl:Thing>
    <owl:Thing   rdf:about="#d">
      <rdf:type rdf:resource="#NoStudentReviewer"/>
    </owl:Thing>
    <owl:Thing  rdf:about="#d">
      <rdf:type rdf:resource="#BadPaperCategory"/>
    </owl:Thing>
    <owl:Thing  rdf:about="#2">
      <rdf:type rdf:resource="#BadPaperCategory"/>
    </owl:Thing>
</result>
```

The result shows that the paper length is exceeded by paper `#3`, the authors of `#a` and `#e` have reviewed their own paper, `#b` and `#d` are students that review a paper, and finally the researcher `#d` is an student with a senior paper and `#2` is a senior paper with an student as author.

We have tested our approach with the Protégé tool (version 4.1) using the Hermit reasoner (version 1.3.4). The Hermit reasoner has been used for triggering the completion rules and data constraints from the mapping of XML into OWL. Figure 6 shows an snapshot of the validation results obtained from Hermit and Protégé.

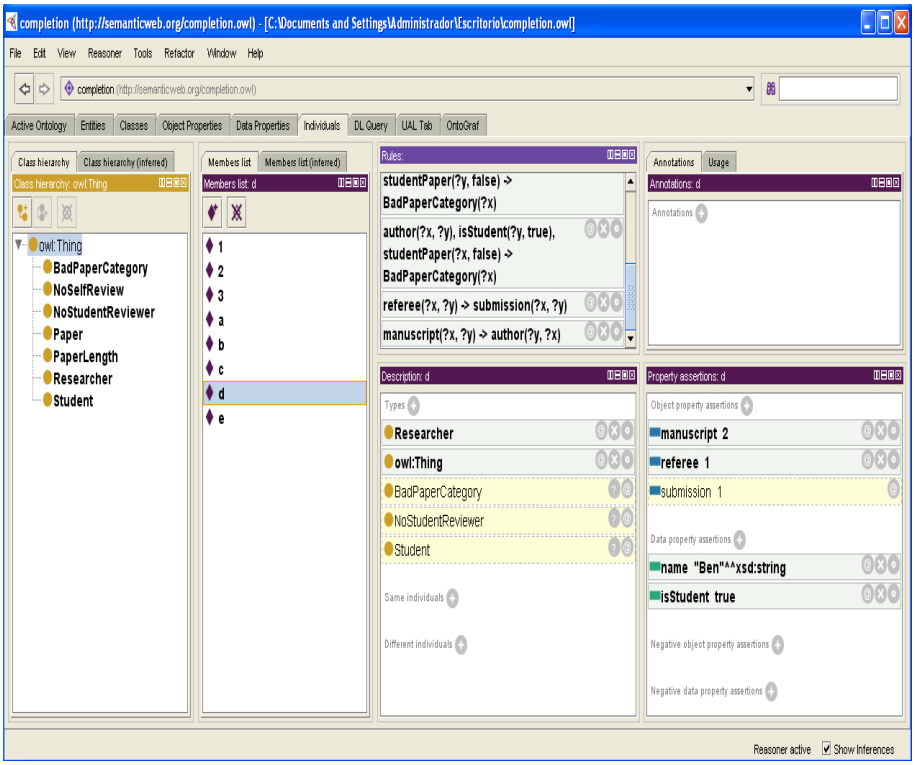

Fig. 6. Visualization of Validation in Protegé

6 Conclusions and Future Work

In this paper we have studied how to validate XML documents from a mapping into OWL and the use of SWRL. We have described how to complete XML/OWL models and how to specify data constraints with SWRL.

As future work, we would like to extend our work in the following directions. Firstly, we could move to a more expressive language, SQWRL, in order to be able to express more complex data constraints. Secondly, we would like to study how to map XML into OWL by using the XML Schema. Finally, we would like to fully integrate our proposal with the Protégé tool. We have in mind the development of a Protégé plugin for the edition and execution of transformations in Protégé. The plugin would allow to validate XML documents in Protégé, and the use of OWL constructors/SWRL to specify completions/data constraints.

References

[AKKP08] W. Akhtar, J. Kopeckỳ, T. Krennwallner, and A. Polleres. XSPARQL: Traveling between the XML and RDF worlds–and avoiding the XSLT

pilgrimage. *The Semantic Web: Research and Applications*, pages 432–447, 2008.

[BA05] H. Bohring and S. Auer. Mapping XML to OWL ontologies. *Leipziger Informatik-Tage*, 72:147–156, 2005.

[BMPS+11] Ivan Bedini, Christopher J. Matheus, Peter F. Patel-Schneider, Aidan Boran, and Benjamin Nguyen. Transforming XML Schema to OWL Using Patterns. In *ICSC*, pages 102–109. IEEE, 2011.

[FZT04] M. Ferdinand, C. Zirpins, and D. Trastour. Lifting XML Schema to OWL. *Web Engineering*, pages 776–777, 2004.

[GC09] R. Ghawi and N. Cullot. Building Ontologies from XML Data Sources. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, pages 480–484. IEEE, 2009.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, 2004.

[KMH11] A. Krisnadhi, F. Maier, and P. Hitzler. OWL and Rules. *Reasoning Web. Semantic Technologies for the Web of Data*, pages 382–415, 2011.

[KRH08] M. Krötzsch, S. Rudolph, and P. Hitzler. Description Logic Rules. In *Proceeding of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 80–84. IOS Press, 2008.

[KSM08] N. Konstantinou, D.E. Spanos, and N. Mitrou. Ontology and database mapping: A survey of current implementations and future directions. *Journal of Web Engineering*, 7(1):1–24, 2008.

[LSD+09] Y.F. Li, J. Sun, G. Dobbie, S. Lee, and H.H. Wang. Verifying semistructured data normalization using SWRL. In *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, pages 193–200. IEEE, 2009.

[MPSP+08] B. Motik, P.F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, et al. OWL 2 web ontology language: Structural specification and functional-style syntax, `http://www.w3.org/TR/owl2-syntax/`. Technical report, www.w3.org, 2008.

[OD09] M.J. O'Connor and A.K. Das. SQWRL: a query language for OWL. In *OWL: Experiences and Directions (OWLED), Fifth International Workshop*, 2009.

[OD11] M.J. O'Connor and A.K. Das. Acquiring OWL ontologies from XML documents. In *Proceedings of the Sixth International Conference on Knowledge Capture, New York*, 2011.

[RRC06] T. Rodrigues, P. Rosa, and J. Cardoso. Mapping XML to Exiting OWL ontologies. In *International Conference WWW/Internet*, pages 72–77, 2006.

[RRC08] T. Rodrigues, P. Rosa, and J. Cardoso. Moving from syntactic to semantic organizations using JXML2OWL. *Computers in Industry*, 59(8):808–819, 2008.

[TC07] C. Tsinaraki and S. Christodoulakis. XS2OWL: a formal model and a system for enabling XML schema applications to interoperate with OWL-DL domain knowledge and semantic web tools. In *Proceedings of the 1st international conference on Digital libraries: research and development*, pages 124–136. Springer-Verlag, 2007.

[TLLJ08] P. Thuy, Y.K. Lee, S. Lee, and B.S. Jeong. Exploiting XML Schema for Interpreting XML Documents as RDF. In *Services Computing, 2008. SCC'08. IEEE International Conference on*, volume 2, pages 555–558. IEEE, 2008.

[VDPM+08] D. Van Deursen, C. Poppe, G. Martens, E. Mannens, and R. Walle. XML to RDF conversion: a Generic Approach. In *Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on*, pages 138–144. IEEE, 2008.

[W3C99] W3C. Document Type Definition. Technical report, `http://www.w3.org/TR/html4/sgml/dtd.html`, 1999.

[W3C07] W3C. Extensible Markup Language (XML), `http://www.w3.org/XML/`. Technical report, www.w3c.org, 2007.

[W3C09a] W3C. OWL 2 Web Ontology Language Direct Semantics, `http://www.w3.org/TR/owl2-direct-semantics/`. Technical report, www.w3.org, 2009.

[W3C09b] W3C. OWL 2 Web Ontology Language RDF-Based Semantics, `http://www.w3.org/TR/owl2-rdf-based-semantics/`. Technical report, www.w3.org, 2009.

[W3C09c] W3C. W3C XML Schema Definition Language (XSD), `http://www.w3.org/XML/Schema`. Technical report, www.w3.org, 2009.

[WRC08] X. Wu, D. Ratcliffe, and M.A. Cameron. XML Schema Representation and Reasoning: A Description Logic Method. In *Services-Part I, 2008. IEEE Congress on*, pages 487–494. IEEE, 2008.

[XC06] H. Xiao and I. Cruz. Integrating and exchanging XML data using ontologies. *Journal on Data Semantics VI*, pages 67–89, 2006.

[ZYMC11] F. Zhang, L. Yan, ZM Ma, and J. Cheng. Knowledge representation and reasoning of XML with ontology. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1705–1710. ACM, 2011.