A Semantic Mapping Approach for Transforming DTD into RDFS/OWL

Win Lai Hnin, Khin Nweni Tun
University of Computer Studies, Yangon, Myanmar
winlaihnin.84@gmail.com,knntun@gmail.com

Abstract

XML (eXtensible Markup Language) and its schema language are becoming a primary data exchange format on the current web. In the next generation of the Semantic Web, the drawbacks of XML and its schema are appeared. Most data on the Web, however, is in the XML format, but using XML requires applications to understand the format of each data source that they access. To achieve the benefits of the Semantic Web involves transforming XML into the Semantic Web language, RDFS/OWL, a process that generally has manual or only semi-automatic components. This paper proposes a set of rules to map DTD to RDFS ontology. This approach mainly considers the constraints of DTD to map RDFS/OWL. Constraints are a fundamental part of the semantics of the data. So, this approach is to ensure the integrity of the structure and provide more meaning for the original DTD while transforming them into RDF Schema.

1. Introduction

In the recent years, XML has reached a wide acceptance as the relevant standardization for storing and exchanging data on the Web. When two participants agree on the XML data format, they begin to transfer and receive data from each others. To support for this trading, XML documents are often built based on their given schemas, which are expressed in DTD (Document Type Definition) or XML Schema. Actually, a DTD or XML Schema contains the knowledge of a structure, data type and relationship among elements in XML documents. In comparing to XML Schema, DTD is the earlier schema language for XML. It is more compact and higher readable than XML Schema. This method targets on DTD, utilizes its declarations to produce suitable mapping rules.

Although XML plays an important role in structuring the document, it has disadvantages to use in the semantic interoperability. XML mainly focuses on the grammar but there is no way to describe the semantics of the document. So, XML data cannot directly use for the Semantic Web, and need another language to interpret this data.

Though, the meaning in the Semantic Web is mostly represented by Resource Description Framework (RDF). RDF encrypts these meanings in the sets of triples that build meaningful webs about related things. These are recognized by the Universal Resource Identifiers (URIs) which tie meanings to a unique definition so that users can easily find them and their relationships on the web. Our main contribution is a set of rules that transform DTD into RDF Schema.

When a data schema is transformed, the corresponding data instances, queries and constraints must also be adapted. In this paper, XML DTD extend with several classes of the constraints range over keys, foreign keys, fixed constraints, enumeration constraints, content constraints as well as cardinality constraints for capturing the semantics of object identities.
They improve semantic specifications and provide a better reference mechanism for native XML application.

The remainder of this paper is organized as follows. Section 1.1 briefly introduces the related work. Section 2 defines the overview of the proposed system. Section 2.1 describes a set of rules for mapping DTD to RDFS ontology. Section 3 describes the discussion, implementation and evaluation of the proposed system. Finally, section 4 concludes this paper.

1.1 Related Work

Several approaches related to schema mapping have been proposed. This section summarizes and analyzes the strength and weakness of these approaches. Based on such examining, we proposed a more comprehensive and efficient solutions for transforming DTD into RDF Schema.

M. Ferdinand et al. [12] proposed direct mappings from XML Schema to OWL as well as they described mappings from XML to RDF graphs, but these mappings are independent of each other, i.e., the generated instances do not necessarily respect the ontology created from the XML Schema. The XML schema to OWL mapping process is based on a set of interpretation and transforming rules from XML Schema to OWL. The mappings proposed in this approach is intended to be applied for the engineering of web applications. This approach does not tackle the question how to create the OWL model, if no XML Schema is available.

P.T.T. Thuy et al. [13] proposed a procedure for transforming valid XML documents into RDF via RDF Schema. This procedure derived classes and properties from XSD, then matched them with elements in XML documents and interpreted all XML data as RDF statements. However, in order to describe the relationship between parent class and child class, the authors defined new RDF vocabulary, rdfx:contain. This definition is not recognized by the RDF evaluation tools or Semantic Web applications. Therefore, the proposed method use existing RDF vocabularies by using rdfs:Container. So, the result of this approach is used directly on the Web without any changes.

Cruz et al. [4] proposed an approach to integrate heterogeneous XML sources using ontology-based mediation architecture. The ontology integration process contained two steps: schema transformation and ontology merging. In the first step, RDFS is used to model each XML source as a local RDF ontology to achieve a uniform representation basis for the ontology merging step. The transformation from XML to RDF is done as follows: complex-type elements are transformed to rdfs:Class, attributes and simple-type elements are transformed to rdfs:Property, and element-subelement relationship is encoded as a class-to- class relationship using a new defined RDFS predicate “rdfx:contain”. In this work the resulting ontology is somehow semantically-poor, since it is based on RDF, and because of the way used to represent element-subelement relationship (using “rdfx:contain”).

This paper proposes a strategy to map DTD to RDF Schema. The proposed method takes into account complex cases arising from different DTD design style. This method also provides a set of mapping bridges between the entities of the XML source and the created RDF ontology.

2. Overview of the Proposed System

The transforming framework of DTD2RDFS/OWL is shown in Figure 1. Having DTD as input, a mapping process converts all DTD components to RDFS/OWL which captures the semantic and maintains the constraints of the element names, attribute names, data types and other declaration of DTD. Moreover, RDFS/OWL is better than the DTD by adding definition of the meaning and relationship between elements in DTD. During this stage, the proposed method also checks and solves the problem whether the next element has the same name with the previous one, if it does, these elements are renamed.

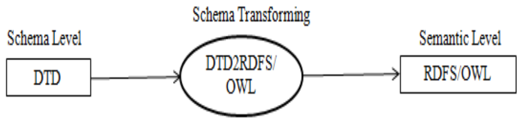


Figure 1: A Framework for Transforming XML into RDF

2.1 Mapping DTD to RDFS/OWL

In this section, the propose method presents the rules for the mapping of the DTD to RDFS/OWL. This method tends to convert every DTD element and attribute to class and property in the RDFS/OWL. The result of this mapping is an RDFS/OWL that maintains the structure and captures the semantics of the DTD.

Creating RDF Schema includes:

(a) Class description: containing rdfs:comment (class name + “class”) - human readable description of the resource – and rdfs:Container (describing the resource is a subclass of a class).

(b) Property description: holding rdfs:domain- indicates the class which this property is described for – and rdfs:range – indicates a class which values of the property must be members or a data type.

The idea of this step is as follows:

Root element: Element defined by <!DOCTYPE> in DTD is mapped to the root-class of RDF schema, which is the first classdeclared by rdfs:Class.

Class (rdfs:Class): A DTD is made up of three main building blocks: ELEMENT, ATTLIST and ENTITY. ELEMENT is the main building block of XML documents. In the DTD, XML elements are

declared with an ELEMENT. An element definition has the following syntax:
<! ELEMENT *element-name* (*element-content*)>

element-content may be EMPTY, or data type, or sequences of children. Because ELEMENT is used to describe elements of a document and each element can contain children elements, the function of these elements is like a class in a structure program, therefore this element will be considered as RDF class. Each rdfs:Class is represented by a unique identifier, rdf:ID.

```
<!DOCTYPE   catalog[
<!ELEMENT  catalog (journal+)>
<!ELEMENT  journal (article|name)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT  article (title, author)>
<!ELEMENT  title (#PCDATA)>
<!ATTLIST  catalog  publisher  #FIXED “O’Reilly”>
<!ATTLIST  catalog title  CDATA  #REQUIRED>
<!ATTLIST catalog photo ENTITY #REQUIRED>
<!ENTITY mt-catalog-1 “mt-catalog1.jpg”>
<!ATTLIST  journal date CDATA  #REQUIRED>
<!ATTLIST  author gender (Male/Female)  #REQUIRED>
<!ATTLIST  article  aid  ID  #REQUIRED>
<!ATTLIST  author  id IDREF #REQUIRED>
]>
```

Figure 2. Definition of Complex Classes in DTD

Moreover, DTD attributes normally contain constraints. For instance, “REQUIRED” means that the value must be appeared. On the contrary, value of an attribute with “IMPLIED” means that it is not demanded. And “NOTATION” means that attribute’s value is a comment. Furthermore, in order to depict the element’s appearing times, this procedure borrows the OWL expressions, such as owl:maxCardinality and owl:minCardinality. Table 1 shows the mapping of DTD constraints to RDFS/OWL concepts.

Table 1. The Mapping of DTD Constraints into RDFS/OWL

DTD	RDFS/OWL
#REQUIRED	owl:minCardinality (=1)
#IMPLIED	owl:Cardinality (=0)
+	owl:minCardinality (=1)
?	owl:minCardinality (=0)
*	owl:minCardinality(=0) owl:maxCardinality (=unbounded)

For nested elements, this procedure don’t use the rdfs:subclassOf, which is available in RDF syntaxes. The reason is because some nested elements in DTD are not actually the sub-class of their parent element. Therefore, rdfs:Container is defined to establish the relationship between child node and parent node For example, the relationship of parent element and child element between three classes, “catalog”, “journal”, and “article” in Fig.2 are described as RDFS/OWL concepts shown in Fig.3.

```
<rdfs:Class   rdf:ID= “catalog”>
     <rdfs:comment> catalog class </rdfs:comment>
</rdfs:Class>
<rdfs:Class  rdf:ID= “journal” owl:minCardinality=  “1”>
      <rdfs:comment> journal class </rdfs:comment>
      <rdfs:Container    rdf:resource= “catalog”/>
</rdfs:Class>
<rdfs:Class  rdf:ID= “article”>
        <rdfs:comment> article class </rdfs:comment>
        <rdfs:Container  rdf:resource= “journal”/>
</rdfs:Class>
```

Figure 3. RDFS/OWL Declaration in Figure 2

Content constraints are relationship of subelements. For instance, the relationship of B and (C|D) is “and”, which means (B≠ null□(C|D)≠null), while the relationship of C and D is ‘or”, which means ((C=null □ D≠null) □ (C≠null □D=null)). In Fig.2, article and name are subelements of journal and title and author are subelements of article. For instance, these are mapped to RDFS/OWL concepts in Fig.4.

```
rdfs:Class>
    <owl:intersectionOf   rdf:parseType= “Collection”>
        <rdfs:Class   rdf:ID= “article”/>
        <rdfs:Class   rdf:ID= “name”/>
    </owl:intersectionOf>
          <rdfs:comment> Class Collection </rdfs:comment>
          <rdfs:Container   rdf:resource= “journal”/>
</rdfs:Class>
<rdfs:Class>
    <owl:unionOf    rdf:parseType= “Collection”>
        <rdfs:Class   rdf:ID= “title”/>
        <rdfs:Class   rdf:ID= “author”/>
    </owl:unionOf>
         <rdfs:comment> Class Collection </rdfs:comment>
         <rdfs:Container    rdf:resource= “article”/>
 </rdfs:Class>
```

Figure 4. RDFS/OWL Declaration in Figure 2

Property (rdf:Property) : For the case an element in DTD is described by <!ELEMENT> tag but its element-content contains data type (#PCDATA or #CDATA), this element will be considered as RDF property, rdf:Property. The property’s domain is the parent class of this property, and its range is the data type of this property. On the other hand, #PCDATA and #CDATA are used for declare character data in XML, so this procedure maps them to “String” data type in RDFS/OWL. For instance, in Fig 5, element “name” has a data type, so it is mapped to RDFS/OWL concepts in Fig.6.

```
<!ELEMENT  name (#PCDATA)>
```

Figure 5. Definition of Complex Classes in DTD

```
<rdf:Property  rdf:ID= "name">
     <rdfs:domain  rdf:resource= "journal"/>
     <rdfs:range   rdf:resource= "http://www.w3.org/1999/02/22-
rdf-syntax-ns#Literal"/>
</rdf:Property>
```

Figure 6. RDFS/OWL Declaration in Figure 5

ATTLIST provides extra information about elements so its function is to describe the property of a class. The attribute definition has the following syntax:
<! ATTLIST *element-name attribute-name attribute-type default-value*>

element-name is the name of element (class) and *attribute-name* is a name of the attribute, in the propose method, it is a name of the property. *attribute-type* is a data type and *default-value* specifies default value of the attribute. For instance, one simple attribute in Fig.7 is mapped to RDFS/OWL concepts in Fig.8.

```
<!ATTLIST  journal date CDATA  #REQUIRED>
```

Figure 7. Definition of Complex Class in DTD

```
<rdf:Property   rdf:ID= "date">
     <rdfs:domain   rdf:resource= "#journal"/>
     <rdfs:range     rdf:resource= "http://www.w3.org/1999/02/22-
rdf-syntax-ns#Literal"/>
</rdf:Property>
```

Figure 8: RDFS/OWL Declaration in Figure 6

If *attribute-type* contains "FIXED" constraints, *default-value* will be considered as the range of its property. In Fig.9, attribute "publisher" has data type "FIXED", so it is mapped to RDFS/OWL concepts in Fig.10.

```
<!ATTLIST  catalog  publisher  #FIXED "O'Reilly">
```

Figure 9. Definition of Complex Classes in DTD

```
<rdf:Property   rdf:ID= "publisher">
      <rdfs:domain   rdf:resource= "#catalog"/>
      <rdfs:range     rdf:resource= "#O'Reilly"/>
</rdf:Property>
```

Figure 10. RDFS/OWL Declaration in Figure 9

There is another notice that XML syntax allows, elements with the same name in a document, but RDFS/OWL does not. RDFS/OWL requires each element has a unique identifier. Since there are two elements that have the same name, **title,** the second repeated name is renamed by adding "has" and its parent name in front of its name as in Fig 11.

```
<rdf:Property   rdf:ID= "has-article-title">
     <rdfs:domain  rdf:resource= "#article">
     <rdfs: range    rdf:resource=
'http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
```

Figure 11: RDFS/OWL Declaration in Figure 2

Beside these, another constraint in DTD is enumeration constraint. Its purpose is to declare a list of possible value of its attribute and attributes in the document must be assigned a value from this list. Furthermore, in order to depict the attribute's enumerated type, this procedure borrows the OWL expressions, such as owl:oneOf. For example, in Fig.12, attribute "gender" has enumeration constraint, so it is mapped to RDFS/OWL concepts in Fig.13.

```
<!ATTLIST  author gender (Male/Female)  #REQUIRED>
```

Figure 12. Definition of Complex Classes in DTD

```
<rdf:Property   rdf:ID= "gender"   owl:oneOf
                      rdf:parseType= "Resource">
     <rdfs:domain   rdf:resource= "#author"/>
     <rdfs:range    rdf:resource= "#Male"/>
     <rdfs:range    rdf:resource= "Female"/>
</rdf:Property>
```

Figure 13. RDFS/OWL Declaration in Figure 12

ENTITY is used to define a shortcut for a common text in XML. Its syntax is as follows:

<! ENTITY *name definition*>

In this case, name is the name of ENTITY and definition is its definition. Because of the function in the DTD, this procedure handles name as a variable and definition as its value. When this procedure meets this variable in the document, its value will be called. For example, one simple entity in Fig 2 is mapped to RDFS/OWL concepts in Fig 14.

```
<rdf:Property   rdf:ID= "photo">
       <rdfs:domain  rdf:resource= "#catalog">
       <rdfs:range   rdf:resource= "mt-catalog1.jpg"^
   http://www.w3.org/1999/02-22-rdf-syntax-ns#mt-catalog-1>
</rdf:Property>
```

Figure 14: RDFS/OWL Declaration in Figure 2

In DTD, there are two kinds of keys constraints such as ID and IDREF. ID is key and IDREF is foreign key. For instance, ID/IDREF in Fig.2 is mapped to RDFS/OWL concepts in Fig.15.

```
<rdf:Property    rdf:ID= "aid">
     <rdfs:domain   rdf:resource= "article"/>
     <rdfs:range     rdf:resource=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
</rdf:Property>
<rdf:Property   rdf:ID= "id">
     <rdfs:domain   rdf:resource= "author"/>
     <rdfs:range    rdf:resource=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal"/>
</rdf:Property>
```

Figure 15. RDFS/OWL Declaration in Figure 2

For instance, in Figure 2, catalog.dtd, root class is *catalog* that contains information of *journal* element. Element *journal* contains three properties, *title*, *publisher* and *date* and two classes; *article* and *name*. A class *article* includes the *title* and *author* elements. But a class *name* is defined by element, but does not contain any other elements, so it has considered as attribute. Since there are two elements that have the same name, *title*, the second repeated name is renamed by adding “has” and its parent name in front of its name. The resulted RDFS ontology as following:

```
< ?xml version= “1.0”?>
    <rdf:RDF  xmlns:rdf=
“http://www.w3.org/1999/02/22-rdf-syntax-ns#”
                  xmls:rdfs=
http://www.w3.org/2000/01/rdf-schema#>
<rdfs:Class   rdf:ID= “catalog”>
    <rdfs:comment> catalog class </rdfs:comment>
</rdfs:Class>
<rdfs:Class  rdf:ID= “journal” owl:minCardinality=  “1”>
     <rdfs:comment> journal class </rdfs:comment>
     <rdfs:Container    rdf:resource= “catalog”/>
</rdfs:Class>
<rdfs:Class  rdf:ID= “article”>
       <rdfs:comment> article class </rdfs:comment>
       <rdfs:Container  rdf:resource= “journal”/>
</rdfs:Class>
<rdfs:Property rdf;ID= “journal” owl:minCardinality= “1”>
     <rdfs:domain rdf:resource= “#catalog”/>
     <rdfs:range  rdf:resource= http://www.w3.org/1999/02/22-rdf-
syntax-ns#Literal/>
</rdfs:Property>
rdfs:Class>
   <owl:intersectionOf   rdf:parseType= “Collection”>
       <rdfs:Class   rdf:ID= “article”/>
       <rdfs:Class   rdf:ID= “name”/>
   </owl:intersectionOf>
<rdfs:comment> Class Collection </rdfs:comment>
        <rdfs:Container   rdf:resource= “journal”/>
</rdfs:Class>
<rdf:Property  rdf:ID= “name”>
    <rdfs:domain  rdf:resource= “journal”/>
    <rdfs:range   rdf:resource= “http://www.w3.org/1999/02/22-
rdf-syntax-ns#Literal”/>
</rdf:Property>
<rdfs:Class>
   <owl:unionOf    rdf:parseType= “Collection”>
       <rdfs:Class   rdf:ID= “title”/>
       <rdfs:Class   rdf:ID= “author”/>
   </owl:unionOf>
     <rdfs:comment> Class Collection</rdfs:comment>
     <rdfs:Container    rdf:resource= “article”/>
 </rdfs:Class>
<rdf:Property   rdf:ID= “has-article-title”>
    <rdfs:domain  rdf:resource= “#article”>
    <rdfs: range    rdf:resource=
‘http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal”/>
<rdf:Property   rdf:ID= “publisher”>
     <rdfs:domain   rdf:resource= “#catalog”/>
     <rdfs:range     rdf:resource= “#O’Reilly”/>
</rdf:Property>
<rdf:Property   rdf:ID= “photo”>
       <rdfs:domain  rdf:resource= “#catalog”>
       <rdfs:range   rdf:resource= “mt-catalog1.jpg”^
   http://www.w3.org/1999/02-22-rdf-syntax-ns#mt-catalog-1>
</rdf:Property>
```

```
<rdf:Property   rdf:ID= “date”>
    <rdfs:domain   rdf:resource= “#journal”/>
    <rdfs:range     rdf:resource= “http://www.w3.org/1999/02/22-
rdf-syntax-ns#Literal”/>
</rdf:Property>
<rdf:Property   rdf:ID= “gender”   owl:oneOf
                 rdf:parseType= “Resource”>
    <rdfs:domain   rdf:resource= “#author”/>
    <rdfs:range    rdf:resource= “#Male”/>
    <rdfs:range    rdf:resource= “Female”/>
</rdf:Property>
<rdf:Property    rdf:ID= “aid”>
     <rdfs:domain   rdf:resource= “article”/>
     <rdfs:range     rdf:resource=
“http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal”/>
</rdf:Property>
“http://www.w3.org/1999/02/22-rdf-syntax-ns#Literal”/>
</rdf:Property>
</rdf:RDF>
```

Clearly, in the above RDF Schema document, there are three classes, *catalog*, *journal*, and *article*. Each class is added a description by using rdfs:comment. The nested class is described by using rdfs:Container. The attribute *title* of the class article is changed to has_article_title. Each property is supported by rdfs:domain and rdfs:range which restrict the anterior and posterior values of a property.

3. Discussion, Implementation and Evaluation

In this section, discuss the reason why this system chooses the RDF for the destination transforming. Of course, other ontology languages than RDF can be used to describe the meaning of the XML, too. However, this system targets on the RDF Schema since it is currently the foundation ontology language for the Semantic Web. Moreover, currently there are some tools supporting for it are available, such as Protégé, Altova and some other reasoning tools.

This system is notably different from other related approaches. First, this system translates between the schemas and updates the changing element during mapping step. While mapping, this system uses the existing RDF Schema vocabularies, especially to express the relationship among nesting classes. Since this approach is based on the DTD definitions and exploits the RDF syntaxes, the transformation process is done automatically without any user intervention.

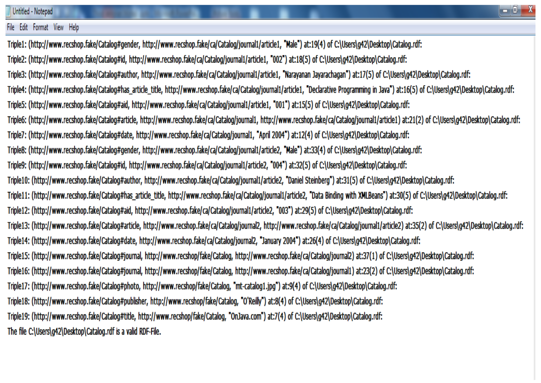
The program transforms DTD document into RDF Schema including *.rdfs. File .rdfs stores descriptions of classes and the relationships between properties and classes as well as the data-types of these properties. The output results are generated using Intel (R) Core (TM) i3 CPU (2.40GHz and 2GB) under Windows 7 running Sun Java v.1.6. Java programming language is used to transform XML DTD into RDFS ontology.

To access the performance of the schema transformation process, four datasets with HDF5.dtd, mondial-3.0.dtd, SigmodRecord.dtd and yahoo.dtd are used. Table 2 is the time performance evaluation of the proposed system.

Table 2. Time Performance Evaluation

Schema name (XML file)	Size in (XML file)	Size out (RDF data)	Execution Time for processing
HDF5	9.34KB	18KB	0.325s
yahoo	1.44KB	3.5KB	0.115s
SigmodRecord	4.00KB	8.2KB	0.257s
Mondial-3.0	4.20KB	8.56KB	0.269s

In order to validate the RDF output result, this paper uses the ICS-FORTH VRP Validation tool. The RDF output results obtained by using this method always pass this validation tool service. The validation result of RDF Schema is shown in Fig. 16.



```
Triple1: (http://www.recshop.fake/Catalog#gender, http://www.recshop.fake/ca/Catalog/journal1/article1, "Male") at:19(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple2: (http://www.recshop.fake/Catalog#id, http://www.recshop.fake/ca/Catalog/journal1/article1, "002") at:18(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple3: (http://www.recshop.fake/Catalog#author, http://www.recshop.fake/ca/Catalog/journal1/article1, "Narayanan Jayarachagan") at:17(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple4: (http://www.recshop.fake/Catalog#has_article_title, http://www.recshop.fake/ca/Catalog/journal1/article1, "Declarative Programming in Java") at:16(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple5: (http://www.recshop.fake/Catalog#aid, http://www.recshop.fake/ca/Catalog/journal1/article1, "001") at:15(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple6: (http://www.recshop.fake/Catalog#article, http://www.recshop.fake/ca/Catalog/journal1, http://www.recshop.fake/ca/Catalog/journal1/article1) at:21(2) of C:\Users\g42\Desktop\Catalog.rdf:
Triple7: (http://www.recshop.fake/Catalog#date, http://www.recshop.fake/ca/Catalog/journal1, "April 2004") at:12(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple8: (http://www.recshop.fake/Catalog#gender, http://www.recshop.fake/ca/Catalog/journal1/article2, "Male") at:33(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple9: (http://www.recshop.fake/Catalog#id, http://www.recshop.fake/ca/Catalog/journal1/article2, "004") at:32(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple10: (http://www.recshop.fake/Catalog#author, http://www.recshop.fake/ca/Catalog/journal1/article2, "Daniel Steinberg") at:31(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple11: (http://www.recshop.fake/Catalog#has_article_title, http://www.recshop.fake/ca/Catalog/journal1/article2, "Data Binding with XMLBeans") at:30(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple12: (http://www.recshop.fake/Catalog#aid, http://www.recshop.fake/ca/Catalog/journal1/article2, "003") at:29(5) of C:\Users\g42\Desktop\Catalog.rdf:
Triple13: (http://www.recshop.fake/Catalog#article, http://www.recshop.fake/ca/Catalog/journal2, http://www.recshop.fake/ca/Catalog/journal1/article2) at:35(2) of C:\Users\g42\Desktop\Catalog.rdf:
Triple14: (http://www.recshop.fake/Catalog#date, http://www.recshop.fake/ca/Catalog/journal2, "January 2004") at:26(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple15: (http://www.recshop.fake/Catalog#journal, http://www.recshop/fake/Catalog, http://www.recshop.fake/ca/Catalog/journal2) at:37(1) of C:\Users\g42\Desktop\Catalog.rdf:
Triple16: (http://www.recshop.fake/Catalog#journal, http://www.recshop/fake/Catalog, http://www.recshop.fake/ca/Catalog/journal1) at:23(2) of C:\Users\g42\Desktop\Catalog.rdf:
Triple17: (http://www.recshop.fake/Catalog#photo, http://www.recshop/fake/Catalog, "mt-catalog1.jpg") at:9(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple18: (http://www.recshop.fake/Catalog#publisher, http://www.recshop/fake/Catalog, "O'Reilly") at:8(4) of C:\Users\g42\Desktop\Catalog.rdf:
Triple19: (http://www.recshop.fake/Catalog#title, http://www.recshop/fake/Catalog, "OnJava.com") at:7(4) of C:\Users\g42\Desktop\Catalog.rdf:
The file C:\Users\g42\Desktop\Catalog.rdf is a valid RDF-File.
```

Fig 16. Validated Result of the RDF Document

This means that RDF Schema output can be used directly by other RDF editors or Semantic Web applications.

4. Conclusion

In this paper, the propose method has presented a set of rules to generate RDFS/OWL from DTD. This method is based on DTD to automatically generate the RDFS/OWL, as well as, a set of mapping bridges. This paper shows that it is possible to mine DTD sources to extract enough knowledge to build semantically reduces the effort to convert the Web into a Semantic Web, empower the data integration and can provide a useful input to more complex and ambitious systems.

References

[1] S.Decker, S.Melnik, F.V.Harmelen, D.Fensel, M.Klein, J.Broekstra, M.Erdmann, and I.Horrocks, "The Semantic Web: The Roles of XML and RDF", 2000, IEEE Internet Computing.

[2] Sergey Melnik, " Bridging the gap between RDF and XML", 1999, available at http://www.db.stanford.edu/melnik/rdf/syntax.html.

[3] Michel Klein, "Interpreting XML via an RDF Schema", 2002, Database and Expert Systems Applications.

[4] I.F. Cruz, H. Xiao and F. Hsu, "An Ontology- Based Framework for XML Semantic Integration" In IDEAS'04: Proceedings of the International Database Engineering and Application Symposium, pp. 217-226, (2004).

[5] P.T.T. Thuy,Young-Koo Lee and Sungyong Lee, "XSD2RDFS and XML2RDF Transforming: a Semantic Approach", 2nd International Conference on Emerging Databases, 2010.

[6] Peter Patel-Schneider and Jerome Simeon, "The Yin/Yang Web: XML syntax and RDF Semantics", 11th International WWW conference, Hawaii, 2002.

[7] Refsnes Data, "Introduction to DTD", 1999-2007, available at: http://www.w3schools.com/dtd/dtd-into.asp.

[8] Dan Brickley, R.V Guha and Brian McBride, "RDF Vocabulary description language 1.0:RDF Schema", W3C, Feb 2004, available at: http://www.w3.org/TR/2004/REC-rdf-schema.

[9] Frank Manola, Eric Miller, "RDF Primer", W3C Recommendation, February 2004, available at: http://www/w3.org/TR/REC-rdf-syntax.

[10] T.Bray, J.Paoli and C.M. Sperberg-McQueen, "eXtensible Markup Language(XML) 1.0", W3C Recommendation, Feb 1998, available at: http://www.w3.org/TR/REC-xml.

[11] P.T.T Thuy, Young-Koo Lee and Sungyoung Lee, "A Semantic Approach for Transforming XML Data into RDF Ontology", Advanced Information Technologies in Future Computing Environments, 2013.

[12] M. Ferdinand, C. Zirpins, and D. Trastour, "Lifting XML Schema to OWL", In Web Engineering- 4th

International Conference, ICWE 2004, Munich, Germany, 2004, pp.354- 358.
[13] P.T.T. Thuy, Young-Koo Lee, Sungyoung Lee and Byeong-Soo Jeong, “Transforming Valid XML Documents into RDF via RDF Schema”, International Conference on Next Generation Web Services Practices, IEEE, October 2007.