Derivation of OWL Ontology from XML Documents by Formal Semantic Modeling

Shihan Yang[1,2], Jinzhao Wu[1], Anping He[1], Yunbo Rao[3]
[1]School of Information Sciences and Engineering, Guangxi University for Nationalities, Nanning, China, 530006
[2]Chengdu Documentation and Information Center, Chinese Academy of Sciences, China, 610041
[3]School of information and software engineering, University of Electronic Sci.&Tech. of China, Chengdu, 610054
Email: {dr.yangsh, hapetis, cloudrao}@gmail.com; himrwujz@yahoo.com.cn

Abstract— **The extensible markup language (XML), a standard format of web information, has a clear syntax but unfortunately an ambiguous formal semantics, which results in being not used directly in semantic web applications. So it is tough job to reuse XML-based data intelligently in the semantic web. To address this problem, a new formal technique of obtaining ontology data automatically from XML documents is proposed. We provide the XML a semantical interpretation by developing a graph-based formal language, which then can be automatically mapped into web ontology language OWL with semantics preserved. The semantic validity and entailment problem are also concerned. The automatical mapping tool has also been developed.**

Index Terms— **semantic web, ontology, XML, OWL, formal semantics**

I. Introduction

Extensible markup language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable, but machine-understandable. XML documents are delivered to most transport and store data on the Internet. Most applications of the semantic web, such as semantic-based integrations, intelligent web searching, and internet based knowledge reasoning, can not use XML documents directly, due to that XML formally governs syntax only but not semantics.

There are several ways to address this problem:

- Transforming XML documents into ontology;
  Techniques of artificial intelligence are always adopted to discover knowledge from XML documents, such as pattern discovery [1], text mining [2], classify [3] [4], and fuzzy computing [5]. Those extracting sematic information are just a little part of meaning in XML documents, and are not enough to provide semantic web applications with ontology data. On the other hand, [6] develops a set of mapping rules between XML schemata and ontology to transforming XML documents into ontology directly. The mapping rules are strongly related to XML and ontology specification, ie., it should be changed frequently when XML schemata or ontology languages are changed.
- Researching and reasoning semantically on XML documents by annotating XML documents with semantical information;
  A lot of annotating techniques are used to extracting knowledge from XML documents [7] [8]. Before annotating XML documents, [9] computes sematic similarity between them for an accurate purpose. [10] annotates XML with domain ontology in order to build XML knowledge base. [11] decides whether there exists a semantics-preserving mapping between two XML schemata by defining the semantics of XML data by means of a semantic annotation based on the specific ontology. The annotating technique needs to change XML documents themselves or adds more information to them for semantically annotating. Usually, changing them or giving additions to them is impossible for most legacy XML documents.
- Adding formal structure to unstructured or semi-structured data (XML documents) for reasoning purpose.
  The literature [12] develops a middle formal language to describe semi-structured data for model checking purpose. [13] and [14] employ graph-based formalism to model semi-structured data and query on it based on the fixed point computation. [15] proposes a labeled graph schema to represent semi-structured data, and [16] extended that with constraints. This method focuses on semantic operations on XML documents, but not on semantic data implied in them.

Our purpose is to transform legacy XML documents into ontology for most semantic web applications automatically or semi-automatically. In order to handle legacy XML documents without changing them, we focus on the semantic data hidden in XML documents, and then propose a new technique to transform XML documents to web ontology language OWL2 [17] through middle graph-based formal language, which expression is between XML schemata and OWL2. We do not directly build mapping rules between XML and ontology, but transform XML into a middle language model, then

This work was supported in part by Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis, Nanning, P. R. China.

doi:10.4304/jcp.8.2.372-379

automatically transform the model into ontology. This indirectly transforming decouple dependence between mapping rules and mapping sources/targets. Here the formal language is used as formal semantically interpreting XML documents, but not as reasoning and computing on them. In fact, we employ formal language as middle semantic modeling language, which can be automatically transformed into OWL2, a standard web ontology language.

In the rest of this paper, section II introduces the graph-based formal language W-graph. Section III gives XML documents a model theory semantic interpretation by W-graph. A mapping from semantic model of XML documents to OWL ontology is developed in section IV, and section V shows an automatically transforming tool to obtain ontology from XML documents. Section VI concludes and discusses the future works.

II. Graph-based Formal Language

W-graph is a simple graph-based formal language that we can use to express instances and schemata of data set. The language here is used as a semantically modeling language for XML documents.

A. Syntax

The following definition distinguishes different two kinds of nodes in the original W-graph [12] definition.

Definition 1: A *W-graph* G_w is a directed labeled graph $\langle N, E, \ell\rangle$, where $N = \{N_a, N_c\}$ is a finite set of nodes, N_a a finite set of *atomic nodes*, depicted as ellipses, N_c a finite set of *composite nodes*, depicted as rectangles, $E \subseteq N_c \times (\mathscr{C} \times \mathscr{L}) \times N$ is a set of labeled edges of the form $\langle m, attribute, n\rangle$, ℓ is a function $\ell : N \cup E \to \mathscr{C} \times (\mathscr{L} \cup \{\bot\})$, $\mathscr{C} = \{solid, dashed\}$, $\mathscr{L}$ is a set of labels, and $\bot$ is a symbol for nothing (empty label), read as bottom.

Nodes in W-graph always represent objects, and edges represent relationships between nodes. There are two types of concrete W-graph: instances and schemata. An instance can be formally defined as the following.

Definition 2: A *W-instance* I is a W-graph such that $\ell_{\mathscr{C}}(e) = solid$ for each edge e of I and $\ell_{\mathscr{C}}(n) = solid$, $\ell_{\mathscr{L}}(n) \neq \bot$ for each node n of I.

In Fig.1 a W-instance is depicted. It describes information that two teachers, one 37 years old, one 40 years old, both of them teach database course, and the student Smith attends the same course. In W-graph, edge attributes are made by two components, the *color* and the *label*, and the function ℓ return a *color* and a *label* (possibly empty, $\bot$) for each node. Edge labels are written close to the corresponding edges, and node labels are written inside the rectangles representing the nodes. The set of colors $\mathscr{C}$ denotes how the lines of nodes and edges are drawn (*solid* or *dashed*), and we also call this information the *color* of a node or edge. On the other hand, the function ℓ can be seen as the composition of the two single valued functions $\ell_{\mathscr{C}}$ and $\ell_{\mathscr{L}}$, so ℓ can be implicity defined also on edges: if $e = \langle m, \langle c, k\rangle, n\rangle$, then $\ell_{\mathscr{C}}(e) = c$ and $\ell_{\mathscr{L}}(e) = k$. Two nodes may be connected by more than one edge, provided that edge *attributes* are different.

Two sets $S, T \subseteq N$, T is *accessible* from S if for each node $n \in T$ there is a node $m \in S$ such that there is a path in W-graph G_w from m to n. For example, in the W-instance I of Fig. 1, the set $\{n_3, n_5, n_6, n_7, n_8\}$ is accessible from the set $\{n_1, n_2, n_4\}$.

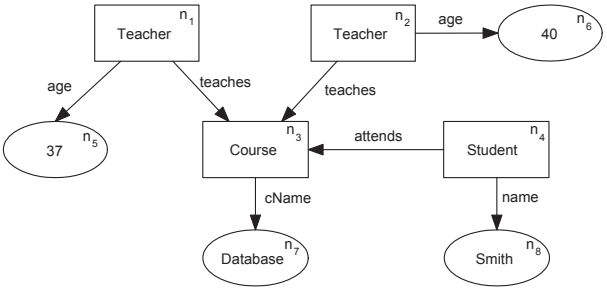

Figure 1. A W-instance I

In the rest of this paper, assume the standard notions of directed path and of subgraph. Denote as $G' \sqsubseteq G$ the fact that G' is a subgraph of G ($G' \sqsubset G$ states that $G' \sqsubseteq G$ and G and G' are different). The *size* of a graph $G = \langle N, E, \ell\rangle$ is $|G| = |N| + |E|$. Moreover, if G is a *W-graph*, then $G_s = \langle N_s, E - s, \ell\,|_{N_s}\rangle$ is the solid subgraph of G, where $N_s = \{n \in N : \ell_{\mathscr{C}}(n) = solid\}$ and $E_s = \{\langle m, \langle solid, \ell\rangle, n\rangle \in E : m, n \in N_s\}$.

A schema gives a pattern to organize data for an instance. The schema of W-instance is also a W-graph, and it could be defined formally as the following.

Definition 3: A *W-schema* S is a *W-graph* such that $\ell_{\mathscr{C}}(e) = solid$ for each edge e of S and $\ell_{\mathscr{C}}(n) = solid$, $\ell_{\mathscr{L}}(n) = \bot$ for each node n of S, i.e., a schema has no values.

So W-graph can be used to describe the knowledge, nodes for concepts, edges for relationships between them, W-schemata for the patterns of the knowledge, and W-instances for the concrete contents of that.

B. Bisimulation-based Semantics

In this subsection, we observe a bisimulation-based semantics for W-graph. The bisimulation provides us the semantic-preserving mapping between XML documents and ontologiey as long as they are precisely encoded by W-graph language.

Definition 4: Given two *W-graphs* $G_0 = \langle N_0, E_0, \ell^0\rangle$ and $G_1 = \langle N_1, E_1, \ell^1\rangle$, a relation $b \subseteq N_0 \times N_1$ is said to be a bisimulation between G_0 and G_1 if and only if:

1) for $i = 0, 1$, $\forall n_i \in N_i$, $\exists n_{1-i} \in N_{1-i}$ such that $n_0 b n_1$,
2) for $\forall n_0 \in N_i$, $\forall n_1 \in N - i$, s.t. $n_0 b n_1 \to \ell^0_{\mathscr{L}}(n_0) = \ell^1_{\mathscr{L}}(n_1) \vee \ell^0_{\mathscr{L}}(n_0) = \bot \vee \ell^1_{\mathscr{L}}(n_1) = \bot$, and
3) for $i = 0, 1$, $\forall n \in N_i$, let
$M_i(n) \stackrel{def}{=\!=} \{\langle m, label\rangle : \langle n, \langle color, label\rangle, m\rangle \in E_i\}$.
Then, $\forall n_0 \in N_0$, $\forall n_1 \in N_1$ such that $n_0 b n_1$,
for $i = 0, 1$, it holds that $\forall \langle m_i, \ell_i\rangle \in M_i(n_i)$,
$\exists \langle m_{1-i}, \ell_{1-i}\rangle \in M_{1-i}(n_{1-i})$, $s.t.\ m_0 b m_1 \wedge \ell_i = \ell_{1-i}$.

Write $G_0 \overset{b}{\sim} G_1$ ($G_0 \overset{b}{\nsim} G_1$) if b is (not) a bisimulation between G_0 and G_1. Write $G_0 \sim G_1$ ($G_0 \nsim G_1$) if there is

(not) a bisimulation between G_0 and G_1, in this case also say that G_0 is bisimilar to G_1.

Condition (1) says that no node in the two graphs can be left out of the relation b. Condition (2) says that two nodes belonging to relation b have exactly the same label, else than the case of dummy nodes, labeled by $\perp$. Condition (3) deals with edge correspondence. If two nodes n_0, n_1 are in relation b, then every edge having n_0 as endpoint should find as a counterpart a corresponding edge with n_1 as endpoint. Notice that output values of the $\ell_{\mathscr{C}}$ function (solid / dashed) are not taken into account in the bisimulation definition.

Based on the bisimulation semantics, we can describe how a *W-instance* is an instance of a *W-schema* as follows.

Definition 5: A *W-instance* I is an instance of a *W-schema* S if $\exists I' \sqsupseteq I$, s.t., $S \sim I'$. S is also said to be a schema for I. I' is said to be a *witness* of the relation schema-instance.

Figure 2 is an example. S is a schema for I (an instance over schema S). To build the witness I', add to I an edge labeled by works linking the entity node *Person* of *Bob* with the entity node *Town*. Moreover, add edges labeled by lives from the two nodes labeled *Person* to the node labeled *Town*, and add also an edge reverse to the father edge. It is easy to check that a bisimulation from S to I' is uniquely determined.

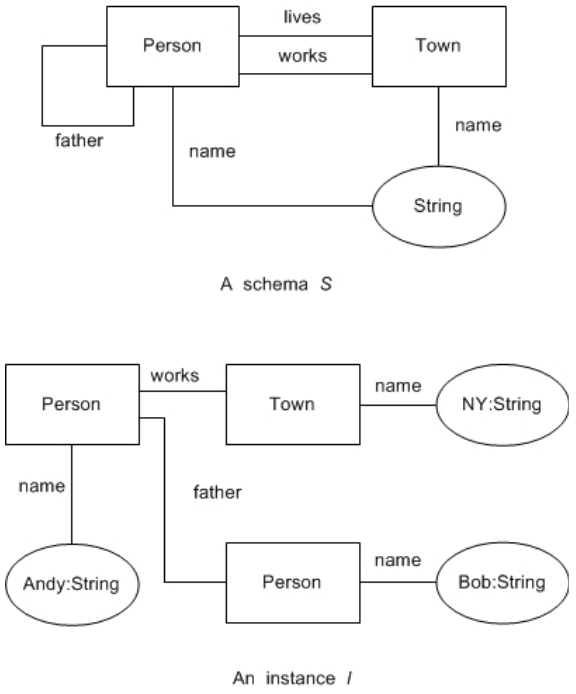

Figure 2. A W-schema S and a W-instance I

C. More Detail Specification

For some concrete transforming tasks, we also give W-Graph more specification in order to depict some details. The structure of nodes and edges in W-graph is shown in figure 3, where the fields are:

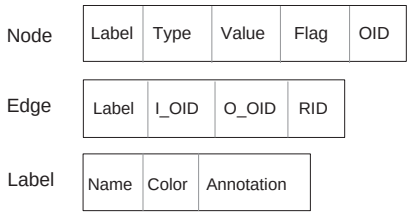

Figure 3. The structure of nodes and edges

- **Label** : A label of node or edge is also a structure, where *Name* is a variable-length character string (describing a concept for a composite node, a data type for an atomic node in the schema, a data value for an atomic node in the instance, and a relationship for an edge), *Color* solid or dashed (the meaning is same as above definition), *Annotation* a comment on node or edge (be null or a variable-length character string). The label gives some readable information on object (node) or relationship (edge).
- **Type** : The data type of the object's value. Each type is either an atomic type (such as integer, string, real number, etc.) for an atomic node, a set type for a composite node in the schema (so this mechanism can handle *nesting* structure), or a concept type (not set type) for a composite node in the instance. The possible concept type must be already occurred in the corresponding schema described by *Name* field of *Label* field of one composite node, and the possible atomic types are not fixed and may vary from information source to information source (here XML documents).
- **Value** : A variable-length value for the object. For composite node, a value is either a set of composite nodes in the schema, or a individual of the concept type in the instance. For atomic node, it is *null* in the schema, but *not null* in the instance.
- **Flag** : A flag is used to distinguish different the type of node, *1* for composite node, *0* for atomic node.
- **OID** : A unique variable-length identifier for the object.
- **I_OID** : The OID of node, which edge is from.
- **O_OID** : The OID of node, which edge is to.
- **RID** : A unique variable-length identifier for the relationship.

We use the conventional *dot seperate expression* to refer to the field of nodes or edges. For example, *N.Label.Name* expresses the name of the label for a node *N*, *M.Value.OID.Label* says the label of a node whose *OID* is in the set / list of value field of the node *M*.

III. XML Semantically Interpreting

XML documents record implicitly rich semantic information, but the information cannot be understood by machines due to absence of formal semantic definition. In this paper, the W-graph is used to interpret formally XML documents, and the interpretation is explicit. The W-

schema interprets the XML schemata, and the W-instance interprets concrete XML documents.

Definition 6: (*Data type*) A data type is a tuple $d = (L, V, \tau)$, where

- L is a finite set of literal, $L \neq \phi$;
- V is a set of values;
- $\tau(d) : L \rightarrow V$ maps a literal with data type d into a corresponding value.

Furthermore, a map $\eta(d) : D \rightarrow N_d$ gives each data type $d \in D$ a name $n \in N_d$, where D is a set of data types, and N_d a set of names of data types.

Definition 7: (*XML model*)A XML model M_{XML} is a two-tuple $M_{XML} = (N, P)$, where

- N is a finite set of nodes. $N = N_e \cup N_a \cup N_d$, N_e is a set of element nodes occurred in XML documents, N_a a set of attribute nodes, and N_d a set of data type names;
- P is a set of binary relationships between nodes, which includes binary relations decided by the order that node occurs in XML documents, $P \subseteq N \times N$.

So a W-graph interpretation can impose on XML schemata and XML concrete documents. The interpretation can obtain as the following two definitions.

Definition 8: (*W-schema interpretation for XML schema*) For a XML model $M_{XML} = (N_x, P)$, a W-schema $WS = \langle N_w, E, \ell \rangle$ is an interpretation based on the following interpretation rule I:

- $N_w = N_x$, where $N_{w_a} = N_{x_d}$, ie., data type node in XML should be interpreted as atomic nodes in W-schema, and $N_{w_c} = N_{x_e} \cup N_{x_a}$, ie., element nodes and attribute nodes in XML should be interpreted as composite nodes in W-schema;
- $E = P$;
- ℓ function satisfies following conditions:
  - $\ell_{\mathscr{C}}(e) = solid, \ell_{\mathscr{C}}(n) = solid, \ell_{\mathscr{L}}(n) = \perp$;
  - $\ell_{\mathscr{L}}(e) =' hasvalue'$, if $p \in P, p \in N_{x_a} \times N_{x_d}$;
  - $\ell_{\mathscr{L}}(e) =' hasattribute'$, if $p \in P, p \in N_{x_e} \times N_{x_a}$;
  - $\ell_{\mathscr{L}}(e) =' relateto'$, others.

Definition 9: (*W-instance interpretation for XML document*) For a XML model $M_{XML} = (N_x, P)$, a W-instance $WI = \langle N_w, E, \ell \rangle$ is an interpretation based on the following interpretation rule I^*:

- $N_w = N_x$, where $N_{w_a} = N_{x_d}$, ie., data type node in XML should be interpreted as atomic nodes in W-instance, and $N_{w_c} = N_{x_e} \cup N_{x_a}$, ie., element nodes and attribute nodes in XML should be interpreted as composite nodes in W-instance;
- $E = P$;
- ℓ function satisfies following conditions:
  - $\ell_{\mathscr{C}}(e) = solid, \ell_{\mathscr{C}}(n) = solid, \ell_{\mathscr{L}}(n) = \tau(d), d \in N_{x_d}$, this is only different from above definition, due to each value of elements should be assigned in the W-instance;
  - $\ell_{\mathscr{L}}(e) =' hasvalue'$, if $p \in P, p \in N_{x_a} \times N_{x_d}$;
  - $\ell_{\mathscr{L}}(e) =' hasattribute'$, if $p \in P, p \in N_{x_e} \times N_{x_a}$;
  - $\ell_{\mathscr{L}}(e) =' relateto'$, others.

So the XML documents could be semantically interpreted with W-graph in terms of the above two definitions. We must notice that the semantical interpretation just transforms the XML model (a tree model) into a W-graph model (a simple graph model) automatically, which is not enough, due to most semantic information still not be drawn out. For solving this problem, we present a visual editor for W-graph language (should be introduced in section V) in order to adjust these semantic information by GUI (Graphic User Interface), so a formally operations on W-graph should be defined here.

Definition 10: Following operators could be imposed on the *W-schema* to change it.

- **Abstracting** : For three composite nodes $m, n, d \in N_c$, if $(m, d), (n, d) \in E$, and $\forall v \in N_c - \{m, n\}, (v, d), (d, v) \notin E$, then drop the node d and atomic nodes associated with it (drop the edges associated with these nodes too), and add two edges $(m, n), (n, m)$ to the W-schema and labeled them uniquely. In practice, some nodes and edges are deleted from the W-schema, and two new edges are added into it. We call this operator *Abstracting*.
- **Supplement** : For a composite node $n \in N_c$, some new composite nodes $n_1, \cdots, n_i$ can be added for *supplementing* it, and these nodes can only have their own atomic nodes, and $(n, n_1), \cdots, (n, n_i)$ are added to E, labeled these edges *supplement*.
- **Rename** : The name of any element in W-schema can be renamed in order to make the element more meaningful.

The operators do not *reduce* the original semantics of the schema. Intuitively, the *abstract* can delete some nodes without reducing basic information, the *supplement* can add some new nodes limitedly, and the *rename* can make the meaning of elements more precise.

IV. Mapping into Ontology

In this section, a mapping function π from W-graph to OWL ontology is provided after introducing simply OWL abstract syntax and semantics, which a transforming algorithm can be based.

A. OWL DL Ontology

OWL DL, the description logic style of using OWL, is based on description logic *SHOIN(D)* [18], [19]. OWL DL can form descriptions of classes, data types, individuals, data values, and axioms that provide information about them.

Definition 11: A OWL DL Ontology is a two-tuple $O = \langle ID, A \rangle$, where

- $ID = ID_C \cup ID_{OP} \cup ID_I \cup ID_{DP} \cup ID_{DR}$ is a finite set of OWL DL identifiers, and $ID_C, ID_{OP}, ID_I, ID_{DP}, ID_{DR}$ are pairwise disjoint. ID_C is a set of class identifiers, ID_{OP} a set of object property identifiers, ID_I a set of individual identifiers, ID_{DP} a set of data type identifiers, and ID_{DR} a set of data range identifiers. Each identifier is a URI (Uniform Resource Identifier) reference, which consists of an absolute URL (Uni-

form Resource Location) or prefix, so-called namespace, and a fragment identifier. For example, the qualified name owl:Thing for the URI reference http://www.w3.org/2002/07/owl#Thing.

- $A = A_C \cup A_P \cup A_I$ is a finite set of OWL DL axioms, and A_C, A_P, A_I are pairwise disjoint. A_C is a set of class axioms, A_P a set of property axioms, and A_I a set of individuals (also called facts).

In table I, the first column gives the OWL DL abstract syntax, while the second column gives the standard description logic syntax.

Definition 12: For an OWL DL ontology $O = \langle ID, A\rangle$, an interpretation [19] of it is $I = \langle \Delta^I, \Delta^I_D, \cdot^I\rangle$, where $\Delta^I \cap \Delta^I_D = \varnothing$, and

- Δ^I is the set of individuals, called individual domain of the interpretation.
- Δ^I_D is the set of data values, called data-value domain.
- $\cdot^I$ is an interpretation function, which is defined in the table I. The interpretation function maps classes into subsets of Δ^I, individuals into elements of Δ^I, data types into subsets of Δ^I_D, data values into elements of Δ^I_D, object properties into subsets of $\Delta^I \times \Delta^I$ and the data type properties into subsets of $\Delta^I \times \Delta^I_D$.

In table I, the third column gives the interpretation-based semantics of OWL DL language, where a symbol $\sharp$ denotes the cardinality of a set.

B. Mapping W-graph into OWL

The W-graph can be automatically transformed into an OWL2 ontology. We build a formal mapping

$$\pi : \Sigma_{W-graph} \rightarrow \Sigma_{OWL}$$

as follows, $\Sigma_{W-graph}, \Sigma_{OWL}$ be finite sets of alphabet of the graph and of the ontology respectively.

Definition 13: For a given W-graph $I = \langle N, E, \ell\rangle$ with specification defined before, it can be formally mapped into a OWL $O = \langle ID, A\rangle$ according to the following function π,

- $\pi(n_c.Label.Name) = id_C, n_c \in N_c \subseteq N, id_C \in ID_C \subseteq ID$, when $n_c.Flag = 1, n_c.Type = set$;
- $\pi(n_a.Label.Name) = id_{DR}, n_a \in N_a \subseteq N, id_{DR} \in ID_{DR} \subseteq ID$, when $n_a.Flag = 0, n_a.Value = null$;
- $\pi(e_{cc}.Label.Name) = id_{OP}, e_{cc} \in N_c \times N_c, id_{OP} \in ID_{OP} \subseteq ID$;
- $\pi(e_{ca}.Label.Name) = id_{DP}, e_{ca} \in N_c \times N_a, id_{DP} \in ID_{DP} \subseteq ID$;
- $\pi(n_c.Value) = id_I, n_c \in N_c, id_I \in ID_I \subseteq ID$, when $n_c.Flag = 1, n_c.Type \neq set$;
- $\pi(n_a.Type) = id_D \in ID$, when $n_a.Flag = 0, n_a.Value = null$;
- $\pi(n_a.Value) = literal \in n^D_a \subseteq ID_{DP}$, when $n_a.Flag = 0, n_a.Value \neq null$;
- $\pi(e_{ca}) = a_{DP} \in A_P \subseteq A, e_{ca} \in N_c \times N_a$, and the axiom like this:
  `DatatypeProperty(`a_{DP}
  `domain(`$\pi(e_{ca}.I\_OID.Label.Name)$`)`
  `range(`$\pi(e_{ca}.O\_OID.Type)$`)`
  `[Functional])`, where `Functional` is optional;
- $\pi(e_{cc}) = a_{OP} \in A_P, e_{cc} \in N_c \times N_c$, and the axiom like this:
  `ObjectProperty(`a_{OP}
  `domain(`$\pi(e_{cc}.I\_OID.Label.Name)$`)`
  `range(`$\pi(e_{cc}.O\_OID.Label.Name)$`)`
  `[Functional/InverseFunctional])`, where `Functional/InverseFunctional` is optional;
- $\pi(\{e_{ca} \mid for\ an_c \in N_c, 1 \leqslant i \leqslant l, \forall n_{a_i} \in N_a, e_{ca} = (n_c, n_{a_i})\}) = a_C \in A_C \subseteq A, e_{ca} \in N_c \times N_a$, and the axiom like this:
  `Class(`a_C `partial`
  `restriction(`$\pi(n_{a_1}.Label.Name)$
  `allValuesFrom(`$\pi(n_{a_1}.Type)$`))`
  ...
  `restriction(`$\pi(n_{a_l}.Label.Name)$
  `allValuesFrom(`$\pi(n_{a_l}.Type)$`))));`
- $\pi(\{e_{cc} \mid for\ an_c \in N_c, and n_c.Type = set, 1 \leqslant i \leqslant k, \forall n_{c_i} \in N_c, e_{cc} = (n_c, n_{c_i})\}) = \{a_{C_i} \mid 1 \leqslant i \leqslant k\} \subseteq A_C, e_{cc} \in N_c \times N_a$, and axioms like these:
  `subClassOf(`$\pi(n_c.Label.Name)$ $\pi(n_{c_1}.Label.Name)$`)`
  ...
  `subClassOf(`$\pi(n_c.Label.Name)$ $\pi(n_{c_k}.Label.Name)$`)`
  or like these:
  `Class(`$\pi(n_c.Label.Name)$ `partial`
  $\pi(n_{c_1}.Label.Name)$`)`
  ...
  `Class(`$\pi(n_c.Label.Name)$ `partial`
  $\pi(n_{c_k}.Label.Name)$`));`
- $\pi(n_c) = a_i \in A_I, n_c \in N_c$, when $n_c.Flag = 1, n_c.Type \neq set$, and the axiom like this:
  `Individual(`a_i `type(`$\pi(n_c.Lable.Name)$`)`
  `value(`$\pi(n_c.Value)$`));`
- $\pi(n_a) = a_i \in A_I, n_a \in N_a$, when $n_a.Flag = 0, n_a.Value \neq null$ and the axiom like this:
  `Individual(`a_i `type(`$\pi(n_a.Lable.Type)$`)`
  `value(`$\pi(n_c.Value)$`)).`

Intuitively the above definition maps each node in the W-schema into a class description of the ontology, each type of atomic nodes into a data type, each composite node of instance into a individual assertion, each atomic node of instance into a property assertion, each value of atomic node into literal (value of data type), each edge between composite nodes into a corresponding class axiom, each edge between composite node and atomic node into a corresponding property axiom, and each annotation into annotation property of the ontology.

Theorem 1: The mapping from W-graph language to OWL ontology by means of W-graph does not reduce the semantics.

Proof: There are two facets to be proved. Firstly, after XML documents are interpreted semantically into W-schemata and W-instances according to definition 8, 9, the W-graph could be dynamically changed only by operators defined in the definition 10, so we prove that these operators do not changed the semantics according

TABLE I.
OWL DL SYNTAX AND SEMANTICS

Abstract Syntax	DL Syntax	Semantics
Description (C)		
A (URI reference)	A	$A^I \subseteq \Delta^I$
owl:Thing	$\top$	Δ^I
owl:Nothing	$\bot$	$\{\}$
intersectionOf(C_1 C_2)	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
unionOf(C_1 C_2)	$C_1 \sqcup C_2$	$C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
complementOf(C)	$\neg C$	$(\neg C)^I = \Delta^I \backslash C^I$
oneOf($o_1, \cdots$)	$\{o_1, \cdots\}$	$\{o_1, \cdots\}^I = \{o_1^I, \cdots\}$
restriction(R someValueFrom(C))	$\exists R.C$	$(\exists R.C)^I = \{x \mid \exists y.(x,y) \in R^I and\ y \in C^I\}$
restriction(R allValueFrom(C))	$\forall R.C$	$(\forall R.C)^I = \{x \mid \forall y.(x,y) \in R^I \rightarrow y \in C^I\}$
restriction(R hasValue(o))	$R : o$	$(\forall R.o)^I = \{x \mid (x, o^I) \in R^I\}$
restriction(R minCardnality(n))	$\geqslant nR$	$(\geqslant nR)^I = \{x \mid \sharp(\{y.(x,y) \in R^I\}) \geqslant n\}$
restriction(R maxCardnality(n))	$\leqslant nR$	$(\leqslant nR)^I = \{x \mid \sharp(\{y.(x,y) \in R^I\}) \leqslant n\}$
restriction(U someValueFrom(D))	$\exists U.D$	$(\exists U.D)^I = \{x \mid \exists y.(x,y) \in U^I and\ y \in D^D\}$
restriction(U allValueFrom(D))	$\forall U.D$	$(\forall U.D)^I = \{x \mid \forall y.(x,y) \in U^I \rightarrow y \in D^D\}$
restriction(U hasValue(v))	$U : v$	$(\forall U.v)^I = \{x \mid (x, v^I) \in U^I\}$
restriction(U minCardnality(n))	$\geqslant nU$	$(\geqslant nU)^I = \{x \mid \sharp(\{y.(x,y) \in U^I\}) \geqslant n\}$
restriction(U maxCardnality(n))	$\leqslant nU$	$(\leqslant nU)^I = \{x \mid \sharp(\{y.(x,y) \in U^I\}) \leqslant n\}$
Data Ranges (D)		
D (URI reference)	D	$D^D \subseteq \Delta_D^I$
oneOf($v_1, \cdots$)	$\{v_1, \cdots\}$	$\{v_1, \cdots\}^I = \{v_1^I, \cdots\}$
Object Properties (R)		
R (URI reference)	R	$R^I \subseteq \Delta^I \times \Delta^I$
	R^-	$(R^-)^I = (R^I)^-$
Datatype Properties (U)		
U (URI reference)	U	$U^I \subseteq \Delta^I \times \Delta_D^I$
Individuals (o)		
o (URI reference)	o	$o^I \in \Delta^I$
Data values (v)		
v (RDF literal)	v	$v^I = v^D \in D^D$
Class Axioms		
Class(A partial $C_1 \cdots C_n$)	$A \sqsubseteq C_1 \sqcap \cdots \sqcap C_n$	$A^I \subseteq C_1^I \cap \cdots \cap C_n^I$
Class(A complete $C_1 \cdots C_n$)	$A = C_1 \sqcap \cdots \sqcap C_n$	$A^I = C_1^I \cap \cdots \cap C_n^I$
EnumeratedClass(A $o_1 \cdots o_n$)	$A = \{o_1, \cdots, o_n\}$	$A^I = \{o_1^I, \cdots, o_n^I\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^I \subseteq C_2^I$
EquivalentClasses($C_1 \cdots C_n$)	$C_1 = \cdots = C_n$	$C_1^I = \cdots = C_n^I$
DisjointClasses($C_1 \cdots C_n$)	$C_i \sqcap C_j = \bot, i \neq j$	$C_i^I \cap C_j^I = \{\}, i \neq j$
Datatype(D)		$D^I \subseteq \Delta_D^I$
Property Axioms		
DatatypeProperty(U		
super(U_1) $\cdots$ super(U_n)	$U \sqsubseteq U_i$	$U^I \subseteq U_i^I$
domain(C_1) $\cdots$ domain(C_m)	$\geqslant 1U \sqsubseteq C_i$	$U^I \subseteq C_i^I \times \Delta_D^I$
range(D_1) $\cdots$ range(D_l)	$\top \sqsubseteq \forall U.D_i$	$U^I \subseteq \Delta^I \times D_i^I$
[Functional])	$\top \sqsubseteq \leqslant 1U$	U^I is functional
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$	$U_1^I \subseteq U_2^I$
EquivalentProperties(U_1 $\cdots$ U_n)	$U_1 = \cdots = U_n$	$U_1^I = \cdots = U_n^I$
ObjectProperty(R		
super(R_1) $\cdots$ super(R_n)	$R \sqsubseteq R_i$	$R^I \subseteq R_i^I$
domain(C_1) $\cdots$ domain(C_m)	$\geqslant 1R \sqsubseteq C_i$	$R^I \subseteq C_i^I \times \Delta^I$
range(C_1) $\cdots$ range(C_l)	$\top \sqsubseteq \forall R.C_i$	$R^I \subseteq \Delta^I \times C_i^I$
[inverseOf(R_0)]	$R = (^-R_0)$	$R^I = (R_0^I)^-$
[Symmetric]	$R = (^-R)$	$R^I = (R^I)^-$
[Functional]	$\top \sqsubseteq \leqslant 1R$	R^I is functional
[InverseFunctional]	$\top \sqsubseteq \leqslant 1R^-$	$(R^I)^-$ is functional
[Transitive])	$Tr(R)$	$R^I = (R^I)^+$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$	$R_1^I \subseteq R_2^I$
EquivalentProperties(R_1 $\cdots$ R_n)	$R_1 = \cdots = R_n$	$R_1^I = \cdots = R_n^I$
AnnotationProperty(S)		
Facts		
Individual(o type($C_1 \cdots$ type(C_n)	$o \in C_i$	$o^I \in C_i^I$
value(R_1 o_1) $\cdots$ value(R_n o_n)	$(o, o_i) \in R_i$	$(o^I, o_i^I) \in R_i^I$
value(U_1 v_1) $\cdots$ value(U_n o_n))	$(o, o_i) \in U_i$	$(o^I, o_i^I) \in U_i^I$
SameIndividual(o_1 $\cdots$ o_n)	$o_1 = \cdots = o_n$	$o_i^I = o_j^I$
DifferentIndividuals(o_1 $\cdots$ o_n)	$o_i \neq o_j, i \neq j$	$o_i^I \neq o_j^I, i \neq j$

to the bi-simulation semantics defined in the definition 4. For *Abstract* operator, two 1:1 relationships are reduced to a m:n relationship, and the information of that is recorded in the new two edges; for *Supplement* operator, more semantic information are added, and nothing is reduced; and for *Rename*, no formal semantic information is changed except for element renamed for more readable.

Secondly, for a legal W-graph G_W and a mapped OWL ontology $O = \pi(G_W)$ (defined in the definition 13), we prove that a mapping $\alpha : G_W \to G_O$ must be existed, such that $G_O = \alpha(G_W)$ is a model of O (G_O is an interpret of the ontology O). We just let $\alpha = \cdot^G \circ \pi$, where $\circ$ is the compound operator of mapping, $\cdot^G$ an interpret function of O. ■

The theorem shows that it does not reduce the semantics when transforming from the semantic interpretation of XML documents to OWL ontology through the medium formal language W-graph.

V. Automatically Transforming Tool

Based on definition 13, a tool has been developed to transform XML documents into ontology automatically. The processing of the tool is shown in figure 4. XML

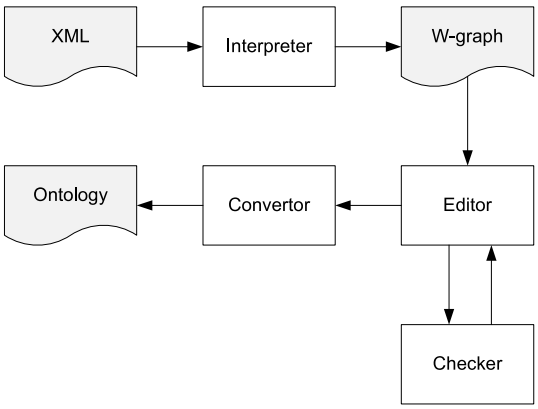

Figure 4. the Processing of the Transforming Tool

documents should be interpreted semantically into W-graph files by the *Interpreter*, and then these W-graph files could be edited in the *Editor* with GUI. During the process of editing, the *Checker* should execute syntax checking and partial semantic checking for W-graph files. At last, the *Convertor* transforms those well-defined W-graph files into OWL2 ontology files automatically.

The *Interpreter* is based on definition 6, 7, 8, and 9. It discovers some semantics from the structure and the contents of XML documents, which semantics information are objective, and maybe not fit to some semantics applications. Then the *Editor* leaves a choice to revise these objective semantics information. The *Editor* is implemented based on operations over W-graph according to definition 10. The semantics information is subjective after editing, and the *Checker* is used to guarantee syntax and partial semantic correctness of these editing. Lastly, the subjective W-graph files are transformed automatically into OWL2 ontology files by the *Convertor*, which is implemented according to definition 13.

The OWL2 files obtained from XML documents can be validated by the online *OWL2 Validator*[1]. The cost of transforming depends on the size of elements of XML documents, which is finite.

VI. Conclusion and Future Works

Based on graph-based formal language W-graph, the method of transforming XML documents into OWL2 automatically has been proposed. The tool for the transforming has also developed. The formal semantics interpretation has been imposed on XML documents by middle formal language W-graph, then the GUI W-graph editor has been provided to revise semantics information, lastly the W-graph after revised has been automatically transformed into OWL2 by the convertor.

In the future, an existed ontology should be introduced to help W-graph edit meaningfully and automatically in a sense. And results of analyzing by AI methods should be added into the process of interpreting XML documents with formal language W-graph, so more meaningful information should be added to middle models.

Acknowledgment

This research is partially supported by the NNSF of China under Grant No. 60873118 and 60973147, the NSF of Guangxi under Grant No. 2011GXNSFA018154, the Sci.&Tech. Foundation of Guangxi under Grant No. 10169-1, Guangxi Scientific Research Project No.201012MS274. This research is also partially supported by the West Light Foundation of the CAS on the project: Automatically extracting semantical metadata of digital document resources on science and technology, by the starting fund of GXUN under Grant No.2011QD017, by the grant of Guangxi Key Laboratory of HCIC Open Fund (HCIC201110), and the Fundamental Research Funds for the Central Universities of Lanzhou University (No.860772).

References

[1] C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic information extraction from semi-structured web pages by pattern discovery," *Decis. Support Syst.*, vol. 35, no. 1, pp. 129–147, Apr. 2003.

[2] C. nan Hsu and C. chi Chang, "Finite-state transducers for semi-structured text mining," in *Proceedings of IJCAI-99 Workshop on Text Mining : Foundations, Techniques and Applications*, Stockholm, Sweden, 1999, pp. 38–49.

[3] M. I. Lam, Z. Gong, and M. Muyeba, "A method for web information extraction," in *Proceedings of the 10th Asia-Pacific Web Conference on Progress in WWW research and development*, ser. APWeb'08, 2008, pp. 383–394.

[4] M. Shaker, H. Ibrahim, A. Mustapha, and L. N. Abdullah, "A framework for extracting information from semi-structured web data sources," in *Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology - Vol. 1*, ser. ICCIT '08, 2008, pp. 27–31.

[1] http://owl.cs.manchester.ac.uk/validator/

[5] P. Ceravolo, M. Nocerino, and M. Viviani, "Knowledge extraction from semi-structured data based on fuzzy techniques," in *Knowledge-Based Intelligent Information and Engineering Systems*, ser. Lecture Notes in Computer Science, M. Negoita, R. Howlett, and L. Jain, Eds. Springer Berlin / Heidelberg, vol. 3215, pp. 328–334.
[6] F. Zhang, L. Yan, Z. M. Ma, and J. Cheng, "Knowledge representation and reasoning of xml with ontology," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11, 2011, pp. 1705–1710.
[7] A. Carlson and C. Schafer, "Bootstrapping information extraction from semi-structured web pages," in *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ser. ECML PKDD '08, 2008, pp. 195–210.
[8] J. Hunter, "Adding multimedia to the semantic web - building an mpeg-7 ontology," in *Proceeding of International Semantic Web Working Symposium (SWWS01)*, Stanford, USA, July 2001.
[9] I.-s. Song, J.-r. Paik, and U.-m. Kim, "Semantic-based similarity computation for xml document," in *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, ser. MUE '07, 2007, pp. 796–803.
[10] H. Li and X. Zhang, "A building method of xml knowledge base using domain ontology," *Information Technology, Computer Engineering and Management Sciences, International Conference of*, vol. 3, pp. 256–259, 2011.
[11] T. Pankowski, "Detecting semantics-preserving xml schema mappings based on annotations to owl ontology," in *Proceedings of the 4th International Workshop on Logic in Databases*, ser. LID '11, 2011, pp. 57–57.
[12] A. Dovier and E. Quintarelli, "Applying model-checking to solve queries on semistructured data," *Comput. Lang. Syst. Struct.*, vol. 35, no. 2, pp. 143–172, July 2009.
[13] S. Cluet, "Modeling and querying semi-structured data," in *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, ser. Lecture Notes in Computer Science, M. Pazienza, Ed. Springer Berlin/Heidelberg, vol. 1299, pp. 192–213.
[14] E. Damiani, B. Oliboni, E. Quintarelli, and L. Tanca, "Modeling semistructured data by using graph-based constraints," in *On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds. Springer Berlin/Heidelberg, vol. 2889, pp. 20–21.
[15] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu, "Adding structure to unstructured data," in *6th Int. Conf. on Database Theory (ICDT'97),LNCS 1186, 336C350*. Springer, 1997.
[16] D. Calvanese and M. Lenzerini, "What can knowledge representation do for semi-structured data," in *Proc. of the 15th Nat. Conf. on Artificial Intelligence*, 1998, pp. 205–210.
[17] B. Motik, P. F. Patel-Schneider, and B. Parsia, "Owl 2 web ontology language structural specification and functional-style syntax," W3C®, 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.
[18] I. Horrocks and U. Sattler, "Ontology reasoning in the shoq(d) description logic," in *IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 199–204.
[19] I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen, "From shiq and rdf to owl: The making of a web ontology language," *Journal of Web Semantics*, vol. 1, pp. 7–26, 2003.

Shihan Yang received the M.S. degree in computer software and theory from Wuhan University in 2001, and the Ph.D. degree in computer software and theory at Chengdu Institute of Computer Application, Chinese Academy of Sciences (CAS), in 2009. He is an assistant professor of School of Information Sciences and Engineering, Guangxi University for Nationalities and an assistant researcher of Chengdu Documentation and Information Center, CAS. He is also a member of Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis. His research interests include formal verification, model checking of hybrid systems, semantic web, software engineer theory, etc.

Jinzhao Wu is a professor of School of Information Sciences and Engineering, Guangxi University for Nationalities. He is chief scientist of Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis. His research interests include symbolic computation, automatic reasoning, designing of logic program, methods of design and analysis of complex concurrent systems, etc. His current research focuses on paraconsistent logic and evaluation and expectation of performance of complex systems, and verification of hybrid systems.

Anping He received the Ph.D. degree in application mathematics from Lanzhou University in 2011. He is an assistant professor of School of Information Sciences and Engineering, Guangxi University for Nationalities. He is also member of Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis. His research interests includes formal specification and verification of software/hardware system, formal semantics, etc.

Yunbo Rao received the B.S. and M.S. degrees from the Sichuan Normal University and the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2003 and 2006, respectively, both in school of computer science and engineering (SCSE). He has as a visiting scholar of electrical engineering of the University of Washington from Oct 2009 to Sep 2011, Seattle, USA. Since 2012, he has been an assistant professor at the school of information and software engineering, University of Electronic Science and Technology of China. His research interests include video enhancement, computer vision, and crowd animation.