Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 181938, 11 pages
http://dx.doi.org/10.1155/2014/181938

Research Article

Semantic Consistency Checking in Building Ontology from Heterogeneous Sources

Shihan Yang,[1,2] Hongyan Tan,[3] and Jinzhao Wu[1,2]

[1] *Guangxi Key Lab of Hybrid Computation and IC Design Analysis, Nanning 530006, China*
[2] *School of Information Sciences and Engineering, Guangxi University for Nationalities (GXUN), Nanning 530006, China*
[3] *Institute of Acoustics Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Shihan Yang; dr.yangsh@gmail.com

Received 16 January 2014; Accepted 8 March 2014; Published 15 April 2014

Academic Editor: X. Song

Semantic collision is inevitable while building a domain ontology from heterogeneous data sources (semi-)automatically. Therefore, the semantic consistency is indispensable precondition for building a correct ontology. In this paper, a model-checking-based method is proposed to handle the semantic consistency problem with a kind of middle-model methodology, which could extract a domain ontology from structured and semistructured data sources semiautomatically. The method translates the middle model into the Kripke structure, and consistency assertions into CTL formulae, so a consistency checking problem is promoted to a global model checking. Moreover, the feasibility and correctness of the transformation is proved, and case studies are provided.

1. Introduction

Semantic web has been a great idea and a promising research area for a dozen years [1]. A main challenge of widening semantic web technologies is lack of semantic data, which is named ontology. So lots of researchers focus on how to transform the legacy mass web data into ontology. The legacy web data in varietal forms can be classified roughly into three types: structured data, such as relational databases; semistructured data, such as XML documents and emails; and nonstructured data, such as general text and video. Recently, technologies of automatically transforming semistructured data or structured data into a domain ontology through mediate modeling are promising [2–6]. In these technologies, some semantic collisions appear inevitably when the same domain ontology has been built from multiple data sources. A domain ontology is a formal expression of a domain knowledge, aiming to unify the general knowledge of a special domain in order to share contents, achieve interoperability, or integrate applications without specific authorization. Unfortunately, the unification is very tough even in the same organization, where the general knowledge exists in very different forms, in very distinct interpretation and in very dissimilar usage for most applications. So automatically creating domain ontology from heterogeneous sources becomes a big challenge. The semantical paradox and ambiguity must be of concern during the process of building the domain ontology, which means the validation of semantic consistency. The semantic consistency guarantees correct and concise specific domain ontology for all kinds of semantic web applications with multiple data sources.

In [2, 7–11], researchers transform structured data (relational database schema) into a middle model and then create a domain ontology from the model. They regard that the relational database is the only data source and that the database is well defined (no ambiguity), which is not always practical. Some semantic preserving properties on transforming are proved, but they do not concern the semantic consistency, which is left for the created domain ontology.

As for semistructured data sources, in [4–6], researchers employ a mediate model language for modeling semistructured data and then transform middle models into the domain ontology. The validity checking has been provided

while collisions happen, which is generally a syntax checking. Semantic consistency checking is also missed.

For building domain ontology from heterogeneous data sources, the semantic consistency checking is necessary. In [12, 13], the same middle formal model language has been adopted to model both structured and semistructured data, so the method can be used to build the domain ontology from heterogeneous data sources. In this paper, we focus on the semantic consistency problem based on this method. The literature [14] develops a middle formal language to describe semistructured data for model-checking purpose and [15] employs graph-based formalism to model semistructured data and queries based on the fixed point computation. We are inspired by the model-checking technology [16], translate the mediate model into a Kripke structure, and encode all semantic query problems into CTL formulae and then we transform the semantic consistency checking problem into a global model-checking procedure.

The following Section 2 recalls the mediate-model-based method of building domain ontologies and the model-checking technology. In Section 3, the mediate model language is introduced, and the model equivalence is analyzed. And the model-checking-based consistency checking technology is proposed in Section 4 in detail. Some cases are studied in Section 5. Section 6 gives a conclusion.

2. Mediate-Model-Based Technology and Model Checking

In this section, the method of formally creating domain with the mediate model [12, 13] and a model-checking technology [16, 17] are introduced.

2.1. Formally Creating Domain Ontology. W-graph, a kind of graph-based formal language, is used to model semantic aspect of relationship databases [12]. The main idea is to execute SQL procedures to retrieve the semantic information of the database instances, transform the result sets into a W-graph model, and then transform the model into the ontology autocompletely, which not only maps schemata to the middle model, but also populates the model with data stored in databases. This language is also used to model semantically XML documents in [5]. In [5], they provide XML documents for a semantical interpretation by the W-graph language. The W-graph model created from relationship databases or XML documents can be automatically transformed into a domain ontology (expressed in ontology web language—OWL [18]). And a W-graph is defined as follows.

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 is a finite set of *atomic nodes* depicted as ellipses, N_c is 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 $\langle m, \text{attribute}, n\rangle$, ℓ is defined as $\ell : N \cup E \to \mathscr{C} \times (\mathscr{L} \cup \{\bot\})$, $\mathscr{C} = \{\text{solid}, \text{dashed}\}$, $\mathscr{L}$ is a set of labels, and $\bot$ is a symbol for nothing (empty label, can be 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 follows.

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 and $\ell_{\mathscr{L}}(n) \neq \bot$ for each node n of I.

In Figure 1 a W-instance I is depicted.

$I = \langle N, E, \ell\rangle$, where

$N = \langle\{n_1, n_2, n_3, n_4\}, \{n_5, n_6, n_7, n_8\}\rangle$,

$E = \langle(n_1, (\text{solid}, \text{teaches}), n_3),$

$(n_2, (\text{solid}, \text{teaches}), n_3),$
$(n_4, (\text{solid}, \text{attends}), n_3), (n_1, (\text{solid}, \text{age}), n_5),$
$(n_2, (\text{solid}, \text{age}), n_6), (n_3, (\text{solid}, c\text{Name}), n_7),$
$(n_4, (\text{solid}, \text{name}), n_8)\rangle,$

$\ell(n_1) = \ell(n_2) = (\text{solid}, \text{Teacher}),$

$\ell(n_3) = (\text{solid}, \text{Course}), \ell(n_4) = (\text{solid}, \text{Student}),$

$\ell(n_5) = (\text{solid}, 37), \ell(n_6) = (\text{solid}, 40),$

$\ell(n_7) = (\text{solid}, \text{Database}), \ell(n_8) = (\text{solid}, \text{Smith}).$

It describes the information that two teachers, one 37 years old and another 40 years old, teach database course, and the student Smith attends this course. In W-graph, edge attribute consists of two components, the *color* and the *label*, and the function ℓ returns a *color* and a *label* (possibly empty, $\bot$) for each node. Edge labels are stuck 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 also be implicitly defined 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.

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

The bisimulation semantics of the language is also given as follows.

Definition 3. 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 is said to be a bisimulation between G_0 and G_1 (write $G_0 \overset{b}{\sim} G_1$) 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 $n_i b n_{1-i}$,

(2) for $i = 0, 1$, $\forall n_i \in N_i$, $\forall n_{1-i} \in N_{1-i}$, s.t. $n_i b n_{1-i} \to \ell^i_{\mathscr{L}}(n_i) = \ell^{1-i}_{\mathscr{L}}(n_{1-i}) \vee \ell^i_{\mathscr{L}}(n_i) = \bot \vee \ell^{1-i}_{\mathscr{L}}(n_{1-i}) = \bot$, and

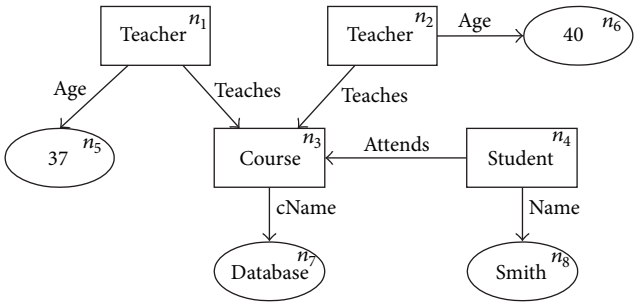

Figure 1: A W-instance I.

(3) for $i = 0, 1$, $\forall n \in N_i$, let $M_i(n) \stackrel{\text{def}}{=} \{\langle m, \text{label}\rangle : \langle n, \langle \text{color}, \text{label}\rangle, m\rangle \in E_i\}$. Then, $\forall n_i \in N_i$, $\forall n_{1-i} \in N_{1-i}$ such that $n_i b n_{1-i}$; 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. $m_i b m_{1-i} \wedge \ell_i = \ell_{1-i}$.

2.2. Model Checking. Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) the model. Here the finite-state model is always called Kripke structure (an automata-like state transition system), and the formal properties are always expressed by computation tree logic (CTL, a logic that is based on a branching-time view) formulae.

Definition 4. A *Kripke structure* $K = \langle \Sigma, \text{Act}, R, I\rangle$ is a transition system over a set Π of atomic propositions, where Σ is a set of states, Act is a set of actions, $R \subseteq \Sigma \times \text{Act} \times \Sigma$ is a transition relations, and $I : \Sigma \to 2^{\Pi}$ is an interpretation.

A path π in a Kripke structure $K = \langle \Sigma, \text{Act}, R, I\rangle$ is an infinite sequence $\pi = \langle \pi_0, a_0, \pi_1, a_1, \pi_2, a_2, \ldots\rangle$ of states and actions (π_i denotes the ith state in the path π), s.t.. For all $i \in N$ it holds that $\pi_i \in \Sigma$ and either $\langle \pi_i, a_i, \pi_{i+1}\rangle \in R$, with $a_i \in \text{Act}$, or there are no outgoing transitions from π_i and for all $j \geqslant i$ it holds that a_j is the special action τ (which is not in Act) and $\pi_j = \pi_i$.

Definition 5. Given the sets Π and Act of atomic propositions and actions, computation tree logic (CTL) formulae are recursively defined as follows:

(1) each $p \in \Pi$ is a CTL formula;

(2) if ϕ_1 and ϕ_2 are CTL formulae, $a \subseteq \text{Act}$, then $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\mathsf{AX}_a(\phi_1)$, $\mathsf{EX}_a(\phi_1)$, $\mathsf{AU}_a(\phi_1, \phi_2)$, and $\mathsf{EU}_a(\phi_1, \phi_2)$ are CTL formulae.

A and E are the universal and existential path quantifiers, while neXt (X) and Until (U) are the linear-time modalities. Composition of formulae of the form $\phi_1 \vee \phi_2$, $\phi_1 \to \phi_2$ can be, respectively, defined by $\neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\neg\phi_1 \vee \phi_2$, and the modalities Finally (F) and Generally (G) can be defined in terms of the CTL formulae: $\mathsf{F}(\phi) = \mathsf{U}(\text{true}, \phi)$ and $\mathsf{G}(\phi) = \neg\mathsf{F}(\neg\phi)$.

Definition 6. Satisfaction of a CTL formula by a state s of the Kripke structure $K = \langle \Sigma, \text{Act}, R, I\rangle$ is defined recursively as follows:

(i) if $p \in \Pi$, then $s \vDash p$ iff $p \in I(s)$. Moreover, $s \vDash \text{true}$, and $s \nvDash \text{false}$;

(ii) $s \vDash \neg p$ iif $s \nvDash p$;

(iii) $s \vDash \phi_1 \wedge \phi_2$ iff $s \vDash \phi_1$ and $s \vDash \phi_2$;

(iv) $s \vDash \mathsf{EX}_a(\phi)$ iff there is a path $\pi = \langle s, x, \pi_1, \ldots\rangle$ s.t. $x \in a$ and $\pi_1 \vDash \phi$;

(v) $s \vDash \mathsf{AX}_a(\phi)$ iff for all paths $\pi = \langle s, x, \pi_1, \ldots\rangle$, $x \in a$ implies $\pi_1 \vDash \phi$;

(vi) $s \vDash \mathsf{EU}_a(\phi_1, \phi_2)$ iff there is a path $\pi = \langle \pi_0, x_0, \pi_1, x_1, \ldots\rangle$, and $\exists j \in \mathbb{N}$ s.t. $\pi_0 = s$, $\pi_j \vDash \phi_2$, and $\forall i < j$, ($\pi_i \vDash \phi_1$ and $x_i \in a$);

(vii) $s \vDash \mathsf{AU}_a(\phi_1, \phi_2)$ iff for all paths $\pi = \langle \pi_0, x_0, \pi_1, x_1, \ldots\rangle$, s.t. $\pi_0 = s$, $\exists j \in \mathbb{N}$, $\pi_j \vDash \phi_2$, and $\forall i < j$, ($\pi_i \vDash \phi_1$ and $x_i \in a$).

Definition 7 (the model-checking problem). *Local model-checking*: given a Kripke structure K, a formula ϕ, and a state s of K, the local model-checking is to verify whether $s \vDash \phi$. *Global model-checking*: given a Kripke structure K, and a formula ϕ, the global model-checking is to find all states s of K such that $s \vDash \phi$.

If Σ is finite, the global model-checking problem for a CTL formula ϕ can be solved in linear running time on $|\phi|\cdot(|\Sigma|+|R|)$, where $|\phi|$ is the length of formula ϕ and $|\Sigma|$ is the number of elements in the set Σ, $|R|$ is the elements number of the transition relations set R [16].

3. Modeling Semantic Inconsistency

When a language has been used to semantically model different data sources for building the same domain ontology, the semantic collision becomes prominent, so the consistency checking is inevitable. Even from a single data source, the incremental procedure of building the semantical model may fail when a new snippet collides semantically with some model segments that existed. Therefore, ambiguities should be detected in order to get correct semantical model during building a domain ontology. The semantic consistency checking is a mechanism for checking whether the model is semantic unambiguity or paradox.

Two kinds of problems would be of concern: redundancy and paradox. The redundancy-free can reduce the size of the model and accelerate modeling procedure. For the middle-model language W-graph, according to Definition 3, two equivalent models (or model segments) cause a redundancy.

As far as the paradox is concerned, there are four types of inconsistency: concept inconsistency, relationship inconsistency, attribute inconsistency, and fact inconsistencies. Before discussing details of these inconsistencies, another special W-graph, so-called W-schema, will be presented. The 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 follows.

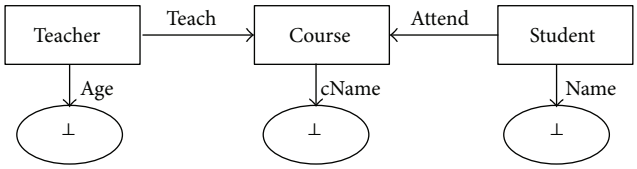

Figure 2: A W-schema S of I.

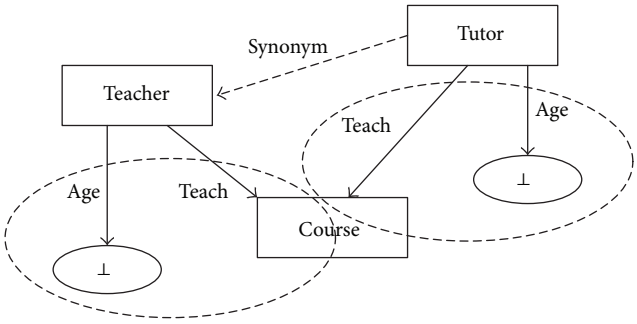

Figure 3: A concept redundancy.

Definition 8. A *W-schema* S is a W-graph such that $\ell_{\mathscr{E}}(e)$ = solid for each edge e of S and $\ell_{\mathscr{E}}(n)$ = solid and $\ell_{\mathscr{L}}(n) = \perp$ for each node n of S; that is, a schema has no values.

For example, Figure 2 depicts the W-schema of the W-instance I in Figure 1. So the W-graph language can be used to model semantic aspects of 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 knowledge. Now we will discuss the details of each inconsistency based on the W-graph.

Concept Redundancy. During the procedure of building semantic model, if a new concept occurs, then we add this concept to the model (add a new node to the W-schema). If a concept paradox occurs, then we also add a new concept to the model. But concepts redundancy will occur if we find two concepts (different names, of course) getting the same attribute set and relationship set. Concepts redundancy always occurs in the W-schema. For example, Figure 3 shows that *Teacher* and *Tutor* are the same concepts here. There is not a new concept *Tutor* to be added into the model, but a new synonym of *Teacher* can be annotated by only one edge labeled "synonym."

We define concept redundancy as follows.

Definition 9. In the W-graph $\langle N, E, \ell \rangle$, two concepts C_1 and C_2 (expressed as nodes n_1, n_2) are redundancy when they get the same adjacent nodes set and edges set:

$$C_1 \approx C_2 \text{ iff } |\{n \mid (n, n_1) \in E\}| = |\{m \mid (m, n_2) \in E\}|, \; |\{n \mid (n_1, n) \in E\}| = |\{m \mid (n_2, m) \in E|$$

and for all m and corresponding n such that $m \sim n$.

Here, $m \sim n$ means that after omitting concepts nodes n_1, n_2 and their adjacent edges the subgraph M including node m bisimulates the subgraph N including node n. Computing $m \sim n$ is an iterative process. For deciding whether $m \sim n$, we delete node m and its adjacent edges from M and node n and its adjacent edges from N and then decide whether the two smaller subgraphs are bisimulation. The iterative process will terminate because the nodes and edges are finite.

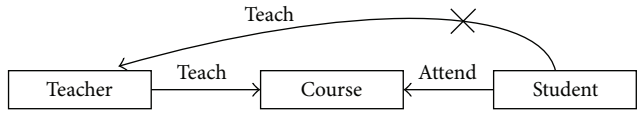

Figure 4: A relationship inconsistency.

Relationships Inconsistency. When a new relationship between concepts during modeling process is found, the relationship cannot be added directly into the model, because sometimes the inconsistency will occur. Firstly, if the relationship is redundant according to the bisimulation, it should be ignored. Secondly, if the relationship is added into the model, we must ensure not to introduce some paradox into the model; otherwise, the relationship inconsistency occurs. For example, Figure 4 shows that the relationship "Student teaches Teacher" should cause a paradox. The relationship may be found from XML documents "Teacher teaches students knowledge, and meanwhile teacher also learns something from students." The relationship "teacher learns from students" may be understood as "Student teaches Teacher," but this contradicts "Teacher teaches Student" relationship, which is reasoned from "Teacher teaches Course" and "Student attends the Course."

The paradox relationship can be formally defined as follows.

Definition 10. In the W-graph $\langle N, E, \ell \rangle$, a paradox relationship between node n_1 and node n_2 is that $(n_1, n_2) \in E$ and $(n_2, n_1) \in E$, where $\ell((n_1, n_2)) = \ell((n_2, n_1))$.

Attribute Inconsistency. The attributes of concepts are another kind of relationship, so-called "isa" or "hasa" relationship. So attribute inconsistency is a kind of relationship inconsistency, but simpler.

Fact Inconsistency. This kind of inconsistency occurs in the W-instance. Facts are the individuals of concepts or attributes. When creating W-instance by populating data from the heterogeneous sources, lots of facts inconsistencies may occur. In Figure 5(a), an attribute fact inconsistency will happen if the fact "age is 40" is added into the model, where the teacher named Charley gets two different ages. And in Figure 5(b), the Teacher2 gets all the same value set of attributes, so Teacher1 and Teacher2 are the concept fact inconsistency. Therefore, the two facts "age is 40" and "Teacher2" cannot be added into the model.

The fact inconsistency is formally defined as follows.

Definition 11. In the W-instance $\langle N, E, \ell \rangle$ where $N = \{N_a, N_c\}$, two facts $n_1, n_2 \in N$ are said to be inconsistent if

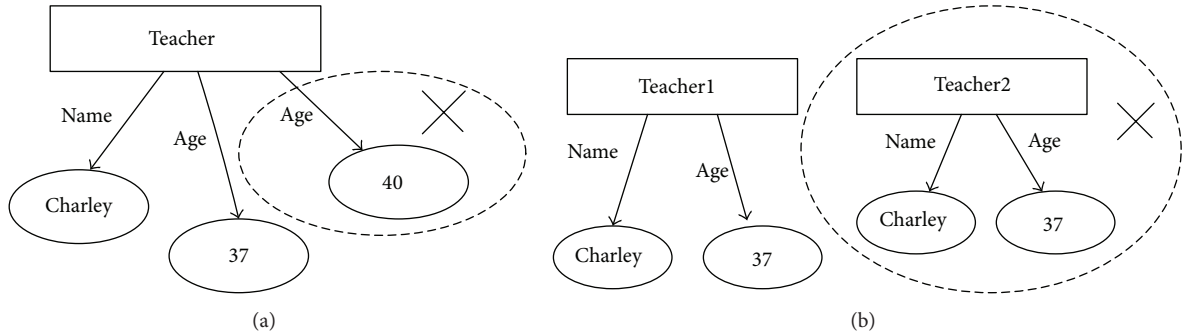

Figure 5: A fact inconsistency.

(1) for $n_1, n_2 \in N_a$, $\exists m \in N_c$ s.t. $\ell((m, n_1)) = \ell((m, n_2))$ and $\ell(n_1) \neq \ell(n_2)$ or

(2) for $n_1, n_2 \in N_c$, $\forall m_1 \in N_a, (n_1, m_1) \in E$ and $m_2 \in N_a, (n_2, m_2) \in E$ s.t. $\ell((n_1, m_1)) = \ell((n_2, m_2))$ and $\ell(m_1) = \ell(m_2)$.

For all these inconsistencies, the core problem is how to discovery the redundancy and paradox in the model. In fact, the procedure of discovery is a subgraph query problem in W-graph.

Definition 12. For a W-graph $G = \langle N, E, \ell \rangle$, a subgraph of G is also a W-graph $G_s = \langle N_s, E_s, \ell_s \rangle$, where $N_s = \{n \in N : \ell_{\mathscr{C}}(n) = \text{solid}\}$ and $E_s = \{(m, (\text{solid}, \ell), n) : m, n \in N_s\}$. Furthermore, given two sets of nodes $S, T \subseteq N$, T is accessible from S if for each $n \in T$ there is a node $m \in S$ such that there is a path in G from m to n.

Definition 13. A W-query is a pointed W-graph, namely, a pair $\langle G, \nu \rangle$ with ν as a node of G (the point). A W-query $\langle G, \nu \rangle$ is accessible if the set N of nodes of G is accessible from $\{\nu\}$.

For example, in Figure 6 three W-queries are depicted. The thick arrows are their *points*, and they are W-queries. Intuitively, the meaning of the first query is to collect all the teachers aged 37. The second query asks for all the teachers that have declared an age (observe the use of an undefined node). The third query, instead, requires collecting all the teachers that teach some courses but not Database, where dashed nodes and lines are introduced to allow a form of negation.

According to Definition 13, an inconsistency checking problem is indeed a query problem in the partial W-graph model. Meanwhile, the semantic equivalence must be of concern during query processing. We call this a *semantical equivalence query problem*. For the incremental procedure of building domain ontology, a semantical equivalence querying should be executed as soon as each new element (concept, relationship, attribute or fact) is added into the model. So, the semantic equivalence querying problem is the basic problem for the consistency checking.

However, the semantical equivalence querying problem for a W-graph model is the subgraph-isomorphism problem, which is NP problem in general. In this paper, we employ model-checking technology to handle semantical equivalence query problem in order to avoid computing subgraph isomorphism.

4. Consistency Checking

In order to employ model-checking technology, we can see a W-graph model as a Kripke structure and a semantical equivalency query as a formula of the temporal logic CTL. In this way, the inconsistency checking problem is reduced to the problem of finding out the states of the model which satisfy the formula (the model-checking problem) that can be done in linear time.

4.1. W-Graph as Kripke Structure. With the following definition, we can build Kripke structure from W-graph.

Definition 14. Let $G = \langle N, E, \ell \rangle$ be a W-graph, the Kripke structure $K_G = \langle \Sigma_G, \text{Act}_G, R_G, I_G \rangle$ over the set of atomic properties Π_G is defined as follows:

(i) Π_G is the set of all node labels of G, $\Pi_G = \{p \mid \forall n \in N, p = \ell(n)\}$.

(ii) The set of states $\Sigma_G = N$.

(iii) The set of actions $\text{Act}_G = \{p \mid \exists m, n \in N, (m, p, n) \in E\} \cup \{p^{-1} \mid \exists m, n \in N, (m, p, n) \in E\} \cup \{\overline{p} \mid \exists m, n \in N, (m, p, n) \in E\}$; that is, the set of actions includes all the edge labels, negative labels (express as $\overline{p}$), and their inverse labels (express as $p^{-1}, \overline{p}^{-1}$); note that negative labels are very different from inverse labels. A negative label edge expresses that there is no special relationship (i.e., "label" relationship) between two nodes, but an inverse label edge expresses that there is a special relationship (i.e., inverse relationship) between this two nodes.

(iv) The ternary transition relation $R_G = E \cup \{(n, p^{-1}, m) \mid (m, p, n) \in E\}$. Moreover, assume that, for each state s with no outgoing edge in E (a leaf in G), a self-loop edge labeled by the special action τ is added.

(v) The interpretation function $I_G(n) = \{\text{true}, \ell_{\mathscr{L}}(n)\}$, where $n \in N$. That is, in each state n the only formulae that hold are the unary atom $\ell_{\mathscr{L}}(n)$ and true.

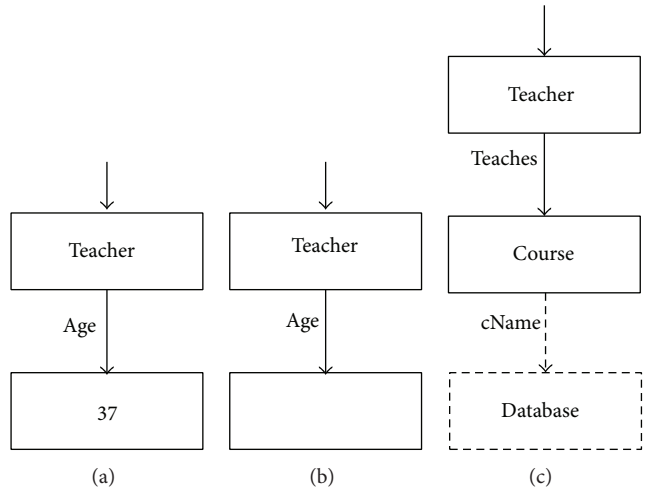

Figure 6: Three W-queries.

For instance, consider the W-graph of Figure 1. It holds the following:

(i) Π_G = {Teacher, Course, Student, 37, 40, Databases, Smith},

(ii) $\Sigma_G = \{n_1, n_2, \ldots, n_8\}$,

(iii) Act_G = {age, age^{-1}, $\overline{\text{age}}$, $\overline{\text{age}}^{-1}$, teaches, teaches^{-1}, $\overline{\text{teaches,teaches}}^{-1}$, attends, attends^{-1}, $\overline{\text{attends,attends}}^{-1}$, name, name^{-1}, $\overline{\text{name}}$, $\overline{\text{name}}^{-1}$, cName, cName^{-1}, $\overline{c\text{Name}}$, $\overline{c\text{Name}}^{-1}$},

(iv) R_G = {(n_1, age, n_5), $(n_5, \text{age}^{-1}, n_1)$, $(n_1, \text{teaches}, n_3)$, $(n_3, \text{teaches}^{-1}, n_1)$, (n_2, age, n_6), $(n_6, \text{age}^{-1}, n_2)$, $(n_2, \text{teaches}, n_3)$, $(n_3, \text{teaches}^{-1}, n_2)$, $(n_4, \text{attends}, n_3)$, $(n_3, \text{attends}^{-1}, n_4)$, $(n_3, c\text{Name}, n_7)$, $(n_7, c\text{Name}^{-1}, n_3)$, (n_4, name, n_8), $(n_8, \text{name}^{-1}, n_4)$, (n_5, τ, n_5), (n_6, τ, n_6), (n_7, τ, n_7), (n_8, τ, n_8)},

(v) I_G = {$I_G(n_1)$ = {true, Teacher}, $I_G(n_2)$ = {true, Teacher}, $I_G(n_3)$ = {true, Course}, $I_G(n_4)$ = {true, Student}, $I_G(n_5)$ = {true, 37}, $I_G(n_6)$ = {true, 40}, $I_G(n_7)$ = {true, Database}, $I_G(n_8)$ = {true, Smith}}.

And this Kripke structure is shown in Figure 7.

4.2. Query as CTL Formula. Reviewing the W-query in Figure 6(a), intuitively, the CTL formula must express the statement "the state Teacher formula is true and there is one next state reachable by an edge labeled age, where the 37 formula is true," this is to say

$$\text{Teacher} \wedge \mathsf{EX}_{\text{age}}(37). \tag{1}$$

For the query in Figure 6(b), the CTL formula should be written as

$$\text{Teacher} \wedge \mathsf{EX}_{\text{age}}(\text{true}), \tag{2}$$

For the query of Figure 6(c), we get the following formula:

$$\text{Teacher} \wedge \mathsf{EX}_{\text{teaches}}(\text{Course}) \wedge \mathsf{AX}_{\text{cname}}(\neg\text{Database}). \tag{3}$$

This formula is true if "there is a node labeled Teacher and there exists one next node labeled Course, which is reachable by an edge labeled teaches, such that for all next to Course nodes labeled Database, the relation cName is not fulfilled". Let us then consider how to encode W-queries in CTL formulae. We study the situation that W-graph and W-query are both acyclic. Firstly, we define an auxiliary function to simply handle labels of nodes and edges in W-graph.

Definition 15. Let $G = \langle N, E, \ell\rangle$ be a W-graph, for all nodes $n \in N$ and for all edges $e = \langle n_1, (c, p), n_2\rangle \in E$; an auxiliary function φ is defined as

$$\varphi(n) = \begin{cases} \ell_{\mathscr{L}}(n), & \text{if } \ell_{\mathscr{L}}(n) \neq \perp, \\ \mathsf{true} & \text{otherwise} \end{cases}$$
$$\varphi(e) = \begin{cases} p, & \text{if } \ell_{\mathscr{C}}(e) = \ell_{\mathscr{C}}(n_2), \\ \overline{p}, & \text{otherwise.} \end{cases} \tag{4}$$

And the query translation can be formally defined as follows.

Definition 16. Let $G = \langle N, E, \ell\rangle$ be a acyclic W-graph, $\nu \in N$, and let $Q = \langle G, \nu\rangle$ be an accessible query. The formula associated with Q is $\Psi_\nu(G)$, where $\Psi_\nu(G)$ is defined recursively as follows:

(i) let $b_1, \ldots, b_h$ $(h \geq 0)$ be the successors of ν, s.t. $\ell_{\mathscr{C}}(b_i)$ = solid,

(ii) let $c_1, \ldots, c_k$ $(k \geq 0)$ be the successors of ν, s.t. $\ell_{\mathscr{C}}(c_i)$ = dashed,

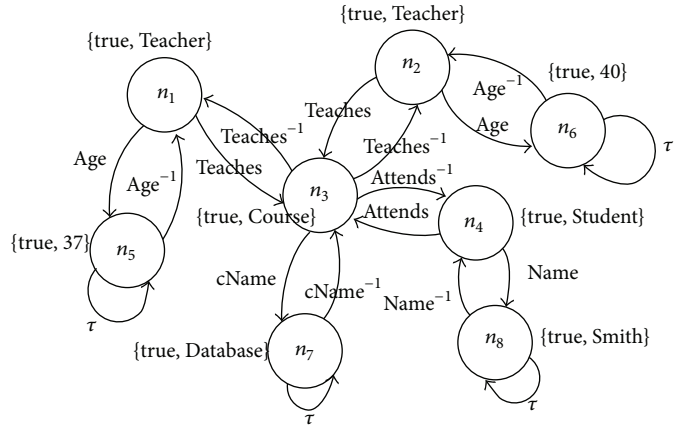

FIGURE 7: The Kripke structure of Figure 1.

(iii) for $i = 1, \ldots, h$ and $j = 1, \ldots, k$, let e_i be the edge which links ν to b_i and e_i' the one which links ν to c_j. If $\ell_{\mathscr{G}}(\nu) = \text{solid}$, then

$$\begin{aligned}\Psi_\nu(G) = \varphi(\nu) \wedge & \bigwedge_{i=1,\ldots,h} \mathsf{EX}_{\varphi(e_i)}\left(\Psi_{b_i}(G)\right) \\ & \wedge \bigwedge_{j=1,\ldots,k} \mathsf{AX}_{\varphi(e_j')}\left(\Psi_{c_j}(G)\right) \\ \text{else } & \left(\ell_{\mathscr{G}}(\nu) = \text{dashed}\right) \Psi_\nu(G) \\ & = \neg\varphi(\nu) \vee \bigvee_{i=1,\ldots,h} \mathsf{AX}_{\varphi(e_i)}\left(\Psi_{b_i}(G)\right) \\ & \vee \bigvee_{j=1,\ldots,k} \mathsf{EX}_{\varphi(e_j')}\left(\Psi_{c_j}(G)\right).\end{aligned} \tag{5}$$

The construction of the formula involves the unfolding of a directed acyclic graph; the size of the formula $\Psi_\nu(G)$ (written as $|\Psi_\nu(G)|$) can grow exponentially with respect to$|G|$. Although that, it is easy to compute the formula without repetitions of subformulae and keep the memory allocation linear w.r.t. $|G|$, so it is natural to compactly represent the formula using a linear amount of memory. This compact representation is allowed by the model-checker NuSMV [19].

Theorem 17. *Given a W-instance I and a W-query $\langle G, \nu\rangle$, let K_I be the Kripke structure associated with I and $\Psi_\nu(G)$ the CTL formula associated with $\langle G, \nu\rangle$. Consistency checking I with $\langle G, \nu\rangle$ can be done in linear time on $|I| \cdot |\Psi_\nu(G)|$.*

Proof. For achieving the consistency checking procedure, there are three steps to follow, translating the W-graph into a Kripke structure, encoding the W-query into a CTL formula, and executing a model-checking process.

Assuming that $|I| = |N| + |E|$, where $|N|$ is the number of nodes in W-instance I and $|E|$ is the number of edges in this graph, notating $K_I = \langle \Sigma_I, \text{Act}_I, R_I, I_I\rangle$, expressing $|\Sigma_I|, |\text{Act}_I|, |R_I|$ as the number of states, actions, and transition relations in $|K_I|$, and writing $|\Psi_\nu(G)|$ as the length of the CTL formula $\Psi_\nu(G)$, then the complexity issues can be analyzed.

Firstly, when creating the Kripke structure, for each node, action, and transitional relation that must be created one by one, the time complexity should be $|\Sigma_I| + |\text{Act}_I| + |R_I| \leq |N| + 4|E| + 3|E| = O(|I|)$, according to Definition 14. Secondly, we consider the length of the encoded CTL formula $|\Psi_\nu(G)| \leq |G| = O(|G|)$, because the worst situation is to encode the whole W-graph into a formula, where $|G|$ is the total number of edges and nodes in this query. The last step is to execute model-checking procedure; the time complexity is $|\Psi_\nu(G)| \cdot (|\Sigma_I| + |R_I|) = O(|G| \cdot |I|)$ according to [16]. So the total time complexity is $O(|I|) + O(|G|) + O(|G| \cdot |I|)$, which is a linear time on $|I| \cdot |\Psi_\nu(G)|$.

If we think that the $|G|$ is always less than or equal to $|I|$ (intuitively, it is always so), then $O(|I|) + O(|G|) + O(|G| \cdot |I|) \leq O(|I|^2) + O(|I|)$, which is a polynomial time over $|I|$. □

The equivalence between two segments of W-graph is the equivalence between two CTL formulae, which can be formally proved in linear time. The consistency checking problem for the W-graph model is promoted to the equivalence proof problem for the CTL formulae with a Kripke structure.

4.3. Checking Consistency. During the procedure of extracting a domain ontology gradually from heterogeneous data sources, semantic consistency has to be checked. According to Definitions 14 and 16, after encoding a W-graph to a Kripke structure and a W-query to a CTL formula, the semantic equivalent query problem on the W-graph has been promoted to a global model-checking problem.

Definition 18. Given a W-instance I and a W-query $\langle G, \nu\rangle$, let K_I be the Kripke structure associated with I and $\Psi_\nu(G)$ the CTL formula associated with $\langle G, \nu\rangle$. The Consistency checking I with $\langle G, \nu\rangle$ amounts to solve the global model-checking problem with the Kripke structure K_I and the CTL formula $\Psi_\nu(G)$, namely, to find all the states s of K_I such that $s \vDash \Psi_\nu(G)$. Algorithm 1 describes this procedure.

Require: I, // W-graph semantic model
$\langle G, v\rangle$ // semantic segment expressed by W-query
Ensure: *consistent or inconsistent*
encoding I *to* K_I; // according to Definition 14
encoding $\langle G, v\rangle$ *to* $\Psi_v(G)$; // according to Definition 16
$S = \{s \mid s \vDash \Psi_v(G)\}$; // model checking
if S is *emptyset* **then**
return *consistent*;
else
return *inconsistent*;
end if

Algorithm 1: Semantic consistency checking.

So semantic consistency checking can be done with model-checking technology. Let us consider each type of inconsistency defined above. As for concepts inconsistency, it is to find out whether there is another concept (equivalent to C) in W-schema S, which has been built to be extended, when a new concept C has been added into the W-schema S. This is to say, a W-query $Q = \langle S, C\rangle$ should be imposed on the S for consistency checking. Let K_S be the Kripke structure of S and let $\Psi_C(S) = \varphi(C)$ be the CTL formula of Q and we should find all states s of K_S such that $s \vDash \varphi(C)$.

As far as relationship inconsistency is concerned, it is to query a W-graph G with a W-query $Q = \langle G, n_1\rangle$ before a new relationship (n_1, n_2) is added into the W-graph model. Let K_G be the Kripke structure of G, and the CTL formula is

$$\Psi_{n_1}(G) = \varphi(n_1) \wedge \mathsf{EX}_{\varphi((n_1,n_2))}(n_2). \tag{6}$$

The consistency checking is to find all states s of K_G such that $s \vDash \Psi_{n_1}(G)$. If the states like this cannot be found, then the model is consistent after adding the new relationship (n_1, n_2); if any state has been found, then the model will be inconsistent and the new relationship cannot be added into the model. This is the relationship redundant inconsistency. As for paradox relationship paradox inconsistency, the CTL formula is

$$\Psi_{n_2}(G) = \varphi(n_2) \wedge \mathsf{EX}_{\varphi((n_1,n_2))}(n_1). \tag{7}$$

As for fact consistency checking, the Kripke structure of the W-instance I is K_I, and the CTL formula for the attribute fact (the instance m of a concept has the concrete attribute value n, i.e., (m, n)) inconsistency is

$$\Psi_m(I) = \varphi(m) \wedge \mathsf{EX}_{\varphi((m,n))}(n). \tag{8}$$

The CTL formula for the concept fact (the instance m of one concept to be added into the model) inconsistency is

$$\Psi_m(I) = \varphi(m) \wedge \bigwedge_{j=1,\ldots,k} \mathsf{AX}_{\varphi((m,n_j))}\left(\Psi_{n_j}(I)\right), \tag{9}$$

where n_j, $j = 1, \ldots, k$ are all successors of m. If some states of K_I satisfy the above formula, then the inconsistency occurs. This is to say, a fact m cannot be added into the model if an equivalent fact has already existed in the model.

5. Cases Study

In this section, we will test the semantic consistency checking technique presented above by using the model checker NuSMV 2.5.4 [20]. NuSMV allows for the representation of finite-state machines (FSMs) and for the analysis of specifications expressed in computation tree logic (CTL) using symbol model-checking techniques. For using the tool, we first rewrite the Kripke structure into a finite-state machine, where edges are not labeled, as follows:

> given a Kripke structure related to a W-instance, replace every labeled edge $m \xrightarrow{\text{action}} n$ by the two edges $m \to \mu$, $\mu \to n$, where μ is a new node labeled action.

The input of the NuSMV tool is represented by a SMV program, which can express both the FSMs and CTL formulae. The Algorithm 2 is the SMV program to describe the Kripke structure of the W-instance in Figure 7 and some CTL specification of consistency checking mentioned above, where the line started with symbol "- -" is comment.

In this SMV program, the set of states of the Kripke structure is chosen by declaring the state variable `state` to assume values {`n0`, ..., `n8`, `teaches`, ..., `tao`}, where actions have also been seen as states. The transition relation of the Kripke structure is expressed by assigning (`ASSIGN`), for each value of the variable `state`, the list of nodes that can be reached from it through one edge. The variables `label` and `nl` are introduced to define the node label of each state identified by the value of the variable `state`. And CTL formulae have been defined by `CTLSPEC`. The formula

> `(nl = Teacher) & EX(state = age) & EX(state = teaches & EX(nl = Course))`

says when a new concept *Teacher*, which has an *age* attribute, has a teaches action, and can teach some Course, is to be added into the model whether there exists an inconsistency. And

> `(nl = Student) & EX(state = attends & EX(nl = Course))`

expresses whether a new relationship Student $\xrightarrow{\text{attends}}$ Course can be added into the model without semantic inconsistency. And

> `(nl = Teacher) & EX(state = age & EX(nl = 40))`

says whether the attribute relationship "the Teacher is 40 years old" can be added into the model without any redundancy and paradox. And

> `(nl = Teacher) & EX(state = age & EX(nl = 37)) & EX(state = teaches & EX(nl = Course & EX(state = cName & EX(nl = Database))))`

expresses whether the fact "the 37-year Teacher teaches Database Course" can be added into the model.

The results we have obtained on the 64-bit windows 7 operation system are shown in Figure 8. The output *true* for a

```
-- SMV program for consistency checking
MODULE main
  VAR
    state:{n1,n2,n3,n4,n5,n6,n7,n8,teaches,inv_teaches,attends,
      inv_attends,age,inv_age,name,inv_name,cName,inv_cName,tao};
    label:{Teacher,Course,Student,Database,Smith,37,40};
  ASSIGN
    init(state) := n1;
    next(state) := case
      state = n1 | state = n2 : {teaches,age};
      state = teaches : n3;
      state = age : {n5,n6};
      state = n3:{inv_attends,inv_teaches,cName};
      state = inv_attends : n4;
      state = inv_teaches : {n1,n2};
      state = cName : n7;
      state = n4 : {attends,name};
      state = attends : n3;
      state = name : n8;
      state = n5 : {inv_age,tao};
      state = inv_age : {n1,n2};
      state = tao : {n5,n6,n7,n8};
      state = n6 : {inv_age,tao};
      state = n7 : {inv_cName,tao};
      state = inv_cName : n3;
      state = n8 : {inv_name,tao};
      state = inv_name : n4;
      TRUE : state;
    esac;
  DEFINE
    nl := case
      state = n1 | state = n2 : Teacher;
      state = n3 : Course;
      state = n4 : Student;
      state = n5 : 37;
      state = n6 : 40;
      state = n7 : Database;
      state = n8 : Smith;
      TRUE : state;
    esac;
  -- concept redundancy checking
  CTLSPEC (nl=Teacher) & EX(state=age) &
          EX(state=teaches & EX(nl=Course))
  -- relationship inconsistency checking
  -- need to be changed to 'init(state) := n4;'
  CTLSPEC (nl=Student) & EX(state=attends & EX(nl=Course))
  -- attribute inconsistency checking
  CTLSPEC (nl=Teacher) & EX(state=age & EX(nl=40))
  -- fact inconsistency checking
  CTLSPEC (nl=Teacher) & EX(state=age&EX(nl=37)) &
          EX(state=teaches & EX(nl=Course &
          EX(state=cName & EX(nl=Database))))
```

Algorithm 2

CTL formula says that at least one inconsistency exists when checking the W-graph model with the new semantic segment (expressed by this formula), so the new segment cannot be added into the model. Otherwise, we can choose another initiate state to check again, until we finish all elements of initiated state set. If we always get a final *false* output, then the inconsistency has not occurred, and the new semantic segment can be added into the model.

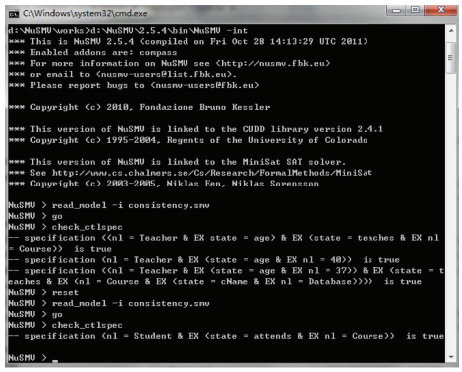

Figure 8: Experiment results.

This test confirms the possibility of solving semantic consistency checking problem by using model-checking on polynomial time methods.

6. Conclusion

For validating semantic consistency during the increasing procedure of building a domain ontology from heterogeneous sources, we employ the model-checking technology to avoid subgraph isomorphism problem, which is NP hard. In order to adopt model-checking method, we formally transform the semantic model into a Kripke structure and the semantic equivalent querying problem into CTL formulae and then the semantic consistency is promoted to the global model-checking problem. The effective experiment with the model-checking tool NuSMV has also been introduced. In the future, the reasoning problem should be considered clearly; for example, some implicative semantic elements would be reasoned from the existing model. If a new semantic segment is equivalent to some implicative semantic elements, the inconsistency also occurs. In the near future, this type of consistency checking should also be regarded.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the Natural Science Foundation of Guangxi under Grants nos. 2011GXNS-FA018154 and 2012GXNS-FGA060003, the Science and Technology Foundation of Guangxi under Grant no. 10169-1, Guangxi Scientific Research Project no. 201012MS274, and the starting fund of GXUN under Grant no. 2011QD017. This paper is supported also by Grant (2012HCIC04) of Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis Open Fund.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] J. Barrasa, Ó. Corcho, and A. Gómez-pérez, "R2o, an extensible and semantically based database-to-ontology mapping language," in *Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB '04)*, pp. 1069–1070, Springer, 2004.

[3] F. Zhang, L. Yan, Z. M. Ma, and J. Cheng, "Knowledge representation and reasoning of XML with ontology," in *Proceedings of the 26th Annual ACM Symposium on Applied Computing (SAC '11)*, pp. 1705–1710, New York, NY, USA, March 2011.

[4] 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 (LID '11)*, pp. 57–57, New York, NY, USA, 2011.

[5] S. Yang, J. Wu, A. He, and Y. Rao, "Derivation of owl ontology from xml documents by formal semantic modeling," *Journal of Computers*, vol. 8, no. 2, pp. 372–379, 2013.

[6] S. Yang and J. Wu, "Mapping relational databases into ontologies through a graph-based formal model," in *Proceedings of the 6th IEEE International Conference on Semantics Knowledge and Grid (SKG '10)*, pp. 219–226, 2010.

[7] C. Bizer, "D2r map—a database to rdf mapping language," in *Proceedings of the 12th ACM International World Wide Web Conference (WWW '03)*, Budapest, Hungary, 2003.

[8] E. Dragut and R. Lawrence, "Composing mappings between schemas using a reference ontology," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pp. 783–800, Springer, Berlin, Germany, 2004.

[9] D. Dejing, L. Paea, K. Shiwoong, and Q. Peishen, "Integrating databases into the semantic web through an ontology-based framework," in *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW '06)*, IEEE Computer Society, p. 54, Washington, DC, USA, 2006.

[10] A. Yuan, B. Alex, and M. John, "Inferring complex semantic mappings between relational tables and ontologies from simple correspondences," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pp. 1152–1169, Springer, Berlin, Germany, 2005.

[11] H.-H. Do and E. Rahm, "Coma: a system for flexible combination of schema matching approaches," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 610–621, VLDB Endowment, 2002.

[12] S. Yang, Y. Zheng, and X. Yang, "Semi-automatically building ontologies from relational databases," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 1, pp. 150–154, July 2010.

[13] S. H. Yang, J. Z. Wu, and A. P. He, "Automatically transforming legacy xml documents into owl ontology," *Applied Mechanics and Materials*, vol. 241, pp. 2638–2644, 2013.

[14] A. Dovier and E. Quintarelli, "Applying model-checking to solve queries on semistructured data," *Computer Languages, Systems and Structures*, vol. 35, no. 2, pp. 143–172, 2009.

[15] S. Cluet, "Modeling and querying semi-structured data," in *Information Extraction a Multidisciplinary Approach to an Emerging Information Technology*, M. Pazienza, Ed., vol. 1299 of *Lecture Notes in Computer Science*, pp. 192–213, Springer, 1997.

[16] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, MIT Press, 1999.

[17] C. Baier and J.-P. Katoen, *Principles of Model Checking*, MIT Press, Cambridge, Mass, USA, 2008.

[18] B. Motik, P. F. Patel-Schneider, and B. Parsia, "Owl 2 web ontology language structural specification and functional-style syntax, W3Cr," 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

[19] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, 2000.

[20] C. Alessandro, R. Marco, C. Roberto et al., "Nusmv2 (version 2.5.4)," 2011, http://nusmv.fbk.eu/.