Refining multiple ontologies: A mappings-based approach*

Jianjiang Lu[1, 2], Baowen Xu[1, 2, 3], Peng Wang[1], Yanhui Li[1], Dazhou Kang[1], Jin Zhou[1]
[1]Department of Computer Science and Engineering, Southeast University, Nanjing, 210096, China
[2]Jiangsu Institute of Software Quality, Nanjing, 210096, China
[3]State Key Laboratory of Software, Wuhan University, Wuhan 430072, China
jjlu@seu.edu.cn

Abstract. In a distributed environment like Semantic Web, complex applications often need to handle multiple ontologies, where the heterogeneity of multiple ontologies arises. In order to reconcile these ontologies, ontology mappings are proposed to provide the interoperating rules between multiple ontologies. However, introducing mappings may cause inconsistencies and redundancies, which will break the soundness and balance of the original ontologies. This paper proposes an approach to eliminate inconsistencies and redundancies. First the mappings are classified into eleven kinds and multiple ontologies are modeled as a graph. Then the changes of the graph, when introducing different kinds of mappings orderly, are analyzed. Following refining steps include semantic checking to keep multiple ontologies sound, and semantic refinement to avoid multiple redundancies. All the steps in our approach can be performed in polynomial time. Our method has high efficiency and is feasible to applications about multiple ontologies in a distributed environment.

Key words: Multiple ontologies, ontology mapping, ontology refinement

(Received April 5, 2005 / Accepted September 8, 2005)

1 Introduction

Ontology is a formal, explicit specification of a shared conceptualization [1]. It represents the common knowledge in domain through defining the concepts, relations, axioms and instances formally. As a powerful model to solve the information sharing problems, ontology plays a key role in the fields including knowledge representation, information retrieval, and so on. Ontology is also the core of Semantic Web [2], which envisions a world-wide distributed architecture where, on the basis of semantic marking up of web resources, data and computational resources will easily interoperate. In the past several years, the development of Semantic Web has improved the popularity of ontology greatly.

To improve their performance and efficiency, ontologies are adopted by more and more applications to represent semantic information. In a distributed environment, even a single application is often required to handle the information from multiple domains. Therefore, multiple ontologies are used frequently in practical cases, such as semantic annotation based on multiple ontologies [3]. Usually, multiple ontologies are produced by different communities or come from different fields. These reasons cause the ontologies frequently heterogeneous, and that has been a major difficulty to utilize multiple ontologies. To solve this problem, ontology integration and mapping are two solutions [4]. Whereas, ontology integration lacks automatic methods to support it and the integration process is required to be executed repetitively when changing or evolving the ontologies. On these accounts, the ontology integration is too high cost and inflexible. And in fact, most applications just need the interoperation between multiple ontologies, ontology mapping is just a way to establish the exchange rules between ontologies, which can realize the ontology interoperation. And naturally the cost of generating mapping is lower than that of integration.

When introducing the mappings, the original independent multiple ontologies are connected as a weak integrated big ontology. However, the structural and semantic balances of singleton ontology would be destroyed meanwhile. In mapping introduction, unexpected redundancies and clashes may appear in multiple ontologies, which will lead to false conclusions

*This work was supported in part by the NSFC (60373066, 60425206 and 90412003), National Grand Fundamental Research 973 Program of China (2002CB312000), National Research Foundation for the Doctoral Program of Higher Education of China (20020286004).

and lower performances. Therefore, it is very necessary to refine the redundancies and clashes to keep the multiple ontologies sound and simple.

In this paper, we propose an approach to eliminate inconsistencies and redundancies. In the first step, the mappings are classified into eleven kinds, and graph theory is used to model the multiple ontologies environment, and then changes of graph are analyzed when introducing different kinds of mappings respectively and orderly. Following refining step includes semantic checking and semantic refinement. The former keeps the multiple ontologies sound, and the latter assures that the multiple ontologies would be irredundant.

This paper is organized as follows: Section 2 gives some principles about ontology and ontology mapping. Then after introducing the mappings into multiple ontologies, the refining problems are analyzed in section 3. Section 4 discusses the method of semantic checking and semantic refining for the multiple ontologies. Section 5 discusses related work, and section 6 makes the conclusions.

2 Principles

In this section, we will give some basic knowledge about ontology. Then we introduce ontology mapping issues among heterogeneous ontologies, and here we focus on mapping classification, which is the base of the subsequent refining process.

2.1 Ontology

Many definitions of ontologies have been given in the last decade, and the most prevalent definition of them is presented by Gruber: an ontology is a formal, explicit specification of a shared conceptualization [1]. In our framework, we consider an ontology as the following definition. This definition is expressive enough to represent ontology in most ontology applications.

Definition 1. (*Ontology*) An ontology is a six-tuple $O=(C, A^C, R, A^R, H, X)$, where C is a set of concepts; A^C is a collection of attribute sets about concepts; R is a set of relations, each relation associates to a pair of concepts; A^R is a collection of attribute sets about relations; H represents a concept hierarchy; and X is the set of axioms.

In above definition, if c_i is a concept in C, and its attributes can be denoted by $A^C(c_i)$. Each relation $r_i<c_p,c_q>$ in R represents a binary association between the individuals in concept c_p and c_q, and the attribute of it can be denoted by $A^R(r_i)$. H is a concept hierarchy derived from C and it is a set of superclass-subclass relations; $<c_p,c_q> \in H$ if c_p is a superclass of c_q. Each axiom in X is a constraint about the concepts, relations and attributes.

2.2 Ontology mapping

Ontology mapping can provide a common layer from which information can be accessed and exchanged in the semantically sound manners [5]. The mismatch is the essential reason that causes ontologies heterogeneous. The ontology mismatch can be divided into two levels: ontology language and meta-model level [6]. The former level of mismatches includes syntax, logical representation, semantic primitives and expressivity of the languages. The second level includes conceptual models, interpretations and terms. Classifying ontology mismatches is important to denote which kind of mismatches can be resolved with a mapping formalism or detected by a matching algorithm. However, in terms of the practical applications, the above classification is far more abstract. In fact, most of practical ontology mappings are just based on the ontology's components (with one exceptions: axioms usually don't need mapping.) directly. Based on the above idea, mappings can be classified into two kinds: concept mappings and relation mappings.

Concept mappings can be divided into seven subclasses.

(1) *Cequal* (synonym mapping). This kind of mappings represents identical or very close concepts in different ontologies. For example, the 'PC' and 'Computer' in different ontologies express the same meaning.

(2) *Cdiffer* (polysemous mapping). This kind of mapping represents homonymous concepts in different ontologies with different meanings. For example the concept 'Doctor' in different ontologies may denote a man having Ph.D degree or a man curing disease.

(3) *Cisa* (hypernym or 'is-a' mapping). This mapping represents the possible hierarchy relations of concepts in different ontologies, for example the 'is-a' relations between 'hammer' and 'tool' in different ontologies.

(4) *Cinstanceof* (hyponym mapping). This mapping expresses the inverse hierarchy relations against the hypernym mapping.

(5) *Chasa* (meronym/holonym mapping). This mapping represents part-whole relations between concepts in different ontologies. For instance, 'Tree' in an ontology has 'Root' and 'Leaf' in another ontology.

(6) *Ccover* (cover mapping). This mapping expresses that the disjunction of several concepts in different ontologies can cover another disjunction of several concepts in different ontologies.

(7) *Copposed* (opposed mapping). This mapping expresses that if a concept can be divided into two disjoint sub-concepts from different ontologies, then they are opposed. For example, two sub-concepts of concept 'Student' in different ontologies: 'MaleStudent' and 'FemaleStudent' are opposed.

Four subclasses are divided for the relation mappings as well.

(8) *Rsubsume* (subsume mapping). This mapping expresses the relations in different ontologies having subsumption relationships. For example, the relation 'ancestor-descender' in an ontology can subsume the relation 'father-son' in another ontology.

(9) *Requal* (relation equal mapping). This mapping denotes the two relations from different ontologies are equal.

(10) *Rinverse* (inverse mapping). The mapping represents that two relations in different ontologies are inverse. For example, the 'Teach' in an ontology and the 'TaughtBy' in another ontology are inverse relations.

(11) *Rcompose* (composed mapping). Two relations of different ontologies may compose a new relation. For example, the relation 'brother' in an ontology and the relation 'father-son' in another ontology can compose a new relation 'uncle-nephew' by that brother(x, y) and father-son(y, z) can imply uncle-nephew(x, z).

The above eleven kinds of mappings are enough to express most common relations among heterogeneous ontologies. And classifying these mappings will be favorable to refining process, since we can deal with kinds of mappings respectively.

3 Analyzing and dividing the refining problems

The mappings connect the isolating multiple ontologies, but meanwhile, it may cause some unavoidable semantic redundancies and conflicts. Redundancies may cause low performance of the multiple ontologies application; and conflict may derive false conclusions and even lead the system to crash. Therefore, checking semantic consistency and reducing semantic redundancies will improve the reasoning capability and soundness for multiple ontologies.

In our view, seven kinds of mapping: *Cequal*, *Cisa*, *Cinstanceof*, *Chasa*, *Ccover*, *Rsubsume* and *Requal* are transitive relations, which can introduce semantic redundancies and conflicts by implying new conclusion from their transitive property. The other four kinds of mappings including *Cdiffer*, *Copposed*, *Rinverse* and *Rcompose*, are not transitive and can not cause reasoning failure and semantic redundancies. Therefore, the semantic checking and reducing of semantic redundancies are aimed at the former seven kinds of transitive mappings.

To be more convenient to refine the multiple ontologies, we divide these mappings further. In terms of concept mappings, *Cinstanceof* is a reverse mapping of *Cisa*, and can be translated to *Cisa* mappings easily. In addition, we can use the similar method of refining *Cisa* to deal with *Chas*. Besides the concept mappings, the relation mappings are needed to be considered here. The way to solve relation mappings is similar to the concepts: *Rsubsume* is similar to *Cisa*; *Requal* is similar to *Cequal* as well. Based on the above analyses, our refining method only focuses on the *Cequal*, *Cisa*, *Ccover* mappings, which can be extended to suit other mappings.

4 Refining multiple ontologies

In order to deal with the problem conveniently, we use graph theory as the basic mathematical model of the multiple ontologies. Then we use graph-based matrix to deal with the mappings in order.

4.1 Mathematical model

We use a direct graph to represent the concept direct inherited relations in multi-ontologies, such a graph is called multi-ontologies direct inherited graph.

Definition 2. The multi-ontologies direct inherited graph G is denoted as $G=(C, E)$, where C is a set of concepts; all concepts in C are numbered form c_1 to c_n according to their ontologies, where $n=|C|$. E is the set of direct inherited edges. If c_1 is a direct child of c_2, $(c_1, c_2) \in E$. G is a direct graph and obviously $(c, c) \notin E$.

A matrix $U_{n\times n}$ is used to describe the set of direct inherited edges E, The values of U_{ij} denote the direct inherited concept relations inside singleton ontology and among multi-ontologies. The value of U_{ij} can be got from the following expression:

$$U_{ij} = \begin{cases} 1 & (C_i, C_j) \in E \\ 0 & (C_i, C_j) \notin E \end{cases}$$

Before any mapping is imported to multi-ontologies, matrix U only has the direct inherited relations in the inner inside every ontology, hence the values 1 are dense near the diagonal of U, as Figure 1 (a) shown, where [O_i] denotes all direct inherited relations in the *i-th* ontology.

In this graph, we will add a global concept *Thing*, and top concepts in singleton ontology will be considered as direct children of *Thing*. And then we change this graph when importing mappings '*Ccover*', '*Cisa*' and '*Cequal*' orderly.

1) *Importing global concept Thing*

We will add *Thing* to the concept set C, and the matrix U becomes $(n+1)\times(n+1)$.

Since every top concept in ontologies is a direct child of *Thing*, the intersections of the top concepts and *Thing* in U are filled with 1, and other intersections with 0. And for *Thing* does not inherit any other concepts, so its corresponding row is filled with 0. Now the change of U is shown as Figure 1 (b).

2) *Importing the Ccover mappings*

The reason of our processing *Ccover* first is that this kind of mappings will add some new concepts to the graph G.

Given a mapping $C_{cover}((a_1, a_2, ..., a_m), (b_1, b_2, ..., b_n))$, where $1 \le m, n \le |C|$. According to the different values of m and n, we will discuss four different processes as follows.

(1) $m = n = 1$. The *Ccover* mapping degenerates into *Cisa* mapping, and relevant discuss is given later.

(2) $m = 1$, $n \ge 2$. We add a new concept c_p in C, and its semantic meaning is $\bigcup_{1\le i\le n} b_i$. New edges $\forall 1 \le i \le n$, (b_i, c_p) and (c_p, a_1) are added to E.

(3) $m \ge 2$, $n = 1$. We add a new concept c_q in C, its semantic meaning is $\bigcup_{1\le i\le m} a_i$. new edges $\forall 1 \le i \le m$ (a_i, c_q), (b_1, c_q) and $(c_q, Thing)$ are added to E. Notice that for the concept c_p in situation (2) has a direct parent a_1, so $(c_p, Thing)$ can not be added to graph G.

(4) $m \ge 2$, $n \ge 2$. We add two new concepts c_p and c_q to graph G, and add a new edge (c_p, c_q) and all other new edges in the situation (2) and (3) as well. Figure 1 (c) shows the change of U in situation (4).

3) Importing the Cisa mappings

The *Cisa* mappings do not change the set of concepts, but add the direct inherited edges to graph G. Given a mapping *Cisa*(c_u,c_v), it implies $(c_u, c_v) \in E$. From the definition of *Cisa* mapping, c_u and c_v belong to different ontologies. We use E_{ij} to denote all direct inherited concept relations in ontology O_i and O_j: $E_{ij} \subseteq E$, $\forall (c_u, c_v) \in E_{ij}$ can infer that c_u in O_i and c_v in O_j. Now the changes of U are shown as Figure 1 (d).

4) *Importing the Cequal mappings*

A *Cequal* mapping declares two concepts of different ontology is synonym. To keep the whole multi-ontologies semantic irredundant, one of the two concepts in a *Cequal* mapping should be removed, and the remaining concept should represent all the semantic meanings of the two original concepts. The concepts reducing algorithm is given as follows. For each mapping *Cequal*(c_k, c_l), in algorithm 1, all semantic information of concept c_l is transferred to concept c_k.

Algorithm 1. Reducing synonym concepts

For each mapping *Cequal*(c_k, c_l), we rewrite U in the following steps:

Step1. $U_{ki}=U_{ki}+U_{li}$, where $1 \le i \le |U|$, if the result $U_{ki} > 0$, let U_{ki}=1. Delete the *l-th* row.

Step2. $U_{ik}=U_{ik}+U_{il}$, where $1 \le i \le |U|$, if the result $U_{ik} > 0$, let U_{ik}=1. Delete *l-th* column.

Next mapping

Through these processes discussed above, we complete the basic mathematical processing for the multi-ontologies graph G when importing ontology mapping.

4.2 Semantic checking

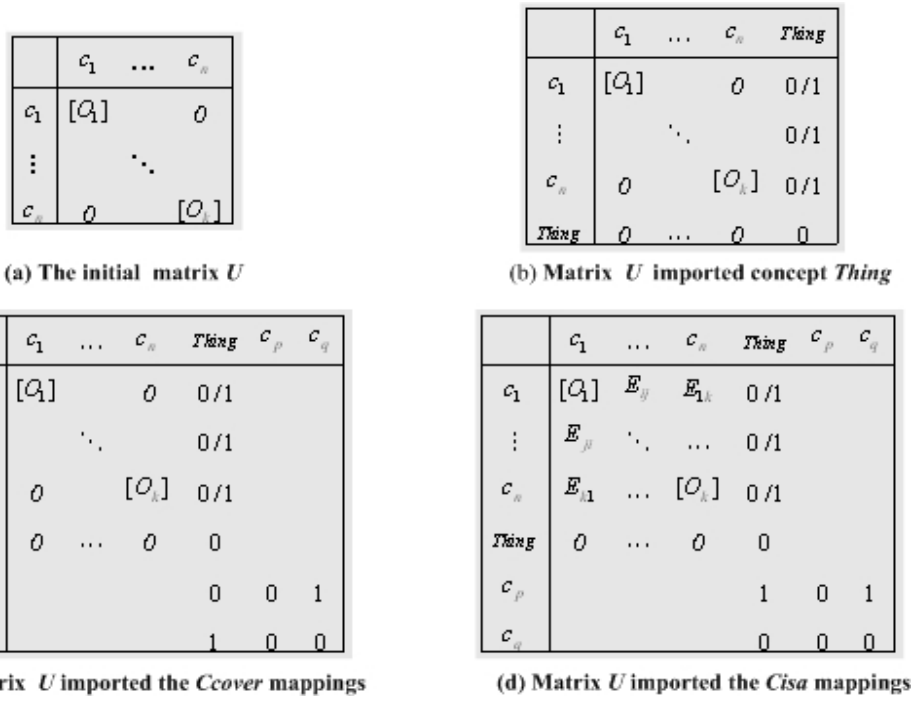
We assume that no cycle exists in the original single ontology [O_i]. If a cycle appears in graph G, the reasons may be the following two.

1) *Healthy cycles*. The concepts or relations in the cycle are equal; just we still have not found the corresponding *Cequal* or *Requal* mappings. Therefore, we can announce all synonym concepts or equal relations in the cycles and use Algorithm 1 to delete the redundant ones.

2) *Ill cycles*. The semantic conflicts may exist in the original ontology and bring about semantic error cycles.

It is disappointing that the computer just can find the cycles, but it is not able to distinguish the two types of

cycles above. Therefore, when the cycles are detected in the graph by the computer, people may judge the types of the cycles to decide whether using Algorithm 1 to deal with them automatically.



(a) The initial matrix U

(b) Matrix U imported concept *Thing*

(c) Matrix U imported the *Ccover* mappings

(d) Matrix U imported the *Cisa* mappings

Figure 1: The changes of matrix U by the introduction of mappings

4.3 Semantic refinement

After semantic checking, *G* is acyclic. In semantic refinement, we will reduce redundant direct inherited relation and assure the new graph *G'* would not lose any semantic information in *G*.

To be more formal, semantic refinement must satisfy two goals: (1) The new graph *G'* is minimal (see definition 3); (2) The refining operations do not change the connectivity of graph *G*, which means *G'* is an equivalent connective graph of *G* (see definition 4).

Definition 3. $G=(C,E)$ is a minimal graph, if for $\forall(c_x,c_y)\in E$, there does not exist a directed path from c_x to c_y: $p_{xy}=(c_x,a_1,...,a_s,c_y)$, where $s>0$, and $1\le i\le s-1$, $(c_x,a_1),(a_s,c_y),(a_i,a_{i+1})\in E$. If the path $p_{xy}=(c_x,a_1,...,a_s,c_y)$ exists, we call it a substitute path of the edge (c_x,c_y). And the length of a path is the number of concept in is minus 1.

Definition 4. $G'=(C,E')$ is an equivalent connective graph of *G*=(*C*, *E*), iff $\forall c_x,c_y\in C$, if *G*=(*C*, *E*) has a directed path $p_{xy}=(c_x,...,c_y)$, then $G'=(C,E')$ must has a directed path from c_x to c_y too.

Definition 5. If $G'_{min}=(C,E'_{min})$ is a minimal graph, and it is an equivalent connective graph of *G*=(*C*, *E*), we call $G'_{min}=(C,E'_{min})$ a minimal equivalent connective graph of *G*=(*C*, *E*).

Formally, the goal of the refinement is to seek for the minimal equivalent connective graph of *G*=(*C*, *E*). From definition 3, 4 and 5, we have three conclusions:

Conclusion 1. After removing all edges having substitute paths in *G*=(*C*, *E*), we can get the minimal graph $G_{min}=(C,E_{min})$.

Conclusion 2. For an edge $(c_x,c_y)\in E$, if it has a substitute path $p_{xy}=(c_x,a_1,...,a_s,c_y)$, we can infer that there are more than one path from c_x to c_y (at least, one substitute path and c_x,c_y), and vice versa.

Conclusion 3. After removing all direct inherited edges having more than one path from c_x to c_y in *G*=(*C*, *E*), we can get the minimal graph $G_{min}=(C,E_{min})$.

Then we discuss the methods to find all edges having substitute paths in graph as follows.

Let matrix U denotes all edges in $G=(C,E)$, and $U^1=U$ and $U^n=U^{n-1}\times U^1$, where $n\geq 2$.

Theorem 1. $\forall c_i,c_j\in C$, U_{ij}^k is the number of *k*-length paths from c_i to c_j.

***Proof*:** We use mathematical induction here.

(1) Obviously, for $\forall c_i,c_j\in C$, U_{ij}^1 denotes the number of 1-length paths from c_i to c_j.

(2) Assume $\exists l\geq 1$, for $\forall c_i,c_j\in C$, U_{ij}^l denotes the number of *l*-length paths from c_i to c_j. For $U^{l+1}=U^l\times U$, we can get $U_{ij}^{l+1}=\sum_{s=1}^{n}U_{is}^l\times U_{sj}$. And the (*l*+1)-length paths from c_i to c_j are composed by *l*-length path from c_i to c_s and 1-length path from c_s to c_j, where $i\neq s\neq j$. Therefore, for $\forall c_i,c_j\in C$, U_{ij}^{l+1} is the number of (*l+1*)-length paths from c_i to c_j.

Let $W=\sum_{k=2}^{n}U^k$. According to conclusion 2 and theorem 1, if (c_i,c_j) have substitute paths, $U_{ij}=1$ and $W_{ij}\geq 1$.

Definition 6. If an edge in *E* has substitute paths, we call it redundant edge and let E_d be the set of them.

$E_d=\{(c_i,c_j)\,|\,(c_i,c_j)\in E \text{ and } W_{ij}\geq 1\}$.

Definition 7. Let p_{mk} be a substitute path of (c_m, c_k), where $p_{mk}=(c_m,a_1,..,a_p,c_x,c_y,b_1,..,b_q,c_k)$, and $p>0$ or $q>0$. Notes (c_x,c_y) is a part of the substitute path p_{mk} of (c_m, c_k), we define a new relation '$\prec$' to denote this relation between them as $(c_x,c_y)\prec(c_m,c_k)$.

Theorem 2. The relation $\prec$ is irreflexive, antisymmetric and transitive.

***Proof*:**

(1) **Irreflexivity.** Assume $(c_x,c_y)\prec(c_x,c_y)$, $\exists$ a directed path $p_{xy}=(c_x,a_1,..,a_p,c_x,c_y,b_1,..,b_q,c_y)$. For $p>0$ or $q>0$, so $\exists$ a cycle $(c_x,a_1,..,a_p,c_x)$ or $\exists$ another cycle $(c_y,b_1,..,b_q,c_y)$. It contradicts with the acyclic premise, hence $\prec$ is irreflexive.

(2) **Antisymmetry.** Assume $(c_x,c_y)\prec(c_m,c_k)$ and $(c_m,c_k)\prec(c_x,c_y)$, where $c_x\neq c_m$ or $c_y\neq c_k$, $\exists$ two directed paths: $p_{xy}=(c_x,a_1,..,a_p,c_m,c_k,b_1,..,b_q,c_y)$ and p_{mk}=(c_m, c_1, …, c_r, c_x, c_y, d_1, …, d_l, c_k), so we get: $(c_x,a_1,..,a_p,c_m,c_1,..,c_r,c_x,c_y,d_1,..,d_l,c_k,b_1,..,b_q,c_y)$ is a directed path. For $p>0$ or $r>0$, and $l>0$ or $q>0$, apparently, there must be a cycle from c_x to c_x or c_y to c_y. That contradicts with the acyclic premise, hence $\prec$ is antisymmetric.

(3) **Transitivity.** Assume $(c_x,c_y)\prec(c_m,c_k)$ and $(c_m,c_k)\prec(c_s,c_t)$, from definition 7, $\exists$ two directed path: $p_{mk}=(c_m,a_1,..,a_p,c_x,c_y,b_1,..,b_q,c_k)$ and p_{st}=(c_s, c_1, …, c_r, c_m, c_k, d_1, …, d_l, c_t). So we can get: $(c_s,c_1,..,c_r,c_m,a_1,..,a_p,c_x,c_y,b_1,..,b_q,c_k,d_1,..,d_l,c_t)$ is a directed path, it implies that $(c_x,c_y)\prec(c_s,c_t)$. Hence $\prec$ is transitive.

Definition 8. We divide E_d into two disjoint subsets: $E_b=\{(c_i,c_j)\,|\,(c_i,c_j)\in E_d,\neg\exists(c_x,c_y)\in E_d,(c_x,c_y)\prec(c_i,c_j)\}$; and $E_x=E_d-E_b$.

Definition 9. For any path $p_{xy}=(c_x,a_1,...,a_s,c_y)$, we define the set $E(p_{xy})$ of edges in p_{xy} as follows:

$$E(p_{xy})=\{(c_x,a_1)\bigcup_{i=1}^{s-1}\{(a_i,a_{i+1})\bigcup(a_s,c_y)\}.$$

Algorithm 2. Construct the substitute path

Input: a substitute path p'_{xy} and its composing edges set $E(p'_{xy})$, and the edges in $E(p'_{xy})$ are stored in turn from c_x to c_y.

Step1. For each edge e_i in $E(p'_{xy})$.

Step2. If $e_i\in E_b$, there must be a substitute path P_l, all edges in $P_l\notin E_d$, use the edges of P_l to substitute e_i in turn, go to Step 5.

Step3. If $e_i\in E_x$, there must be a substitute path P_m, using the edges of P_m to substitute e_i in turn. Return to Step2.

Step4. If $e_i\notin E_b$ and $e_i\notin E_x$, go on to Step 5.

Step5. Next edge

Output: A substitute path p_{xy}, in which all edges $\notin E_d$.

Theorem 3. For any $(c_x,c_y)\in E_d$, it must have a substitute path $p_{xy}=(c_x,a_1,...,a_s,c_y)$, where $s>0$, $1\leq i\leq s-1$, and $E(p_{xy})\bigcap E_d=\varnothing$.

***Proof*:**

(1) For any $(c_x,c_y)\in E_b$, obviously the theorem is valid.

(2) For any $(c_x,c_y)\in E_x$, it must has a substitute path $p'_{xy}=(c_x,a_1,...,a_s,c_y)$. We use algorithm 2 to construct a substitute path p_{xy} from p'_{xy}.

(3) From (1) and (2), theorem 3 is proved.

Theorem 4. After removing all edges in E_d, $G'=(C,E')$ is the equivalent connective graph of the original graph $G=(C,E)$.

***Proof*:** For any $c_x, c_y \in C$, if there is a directed path p'_{xy} from c_x to c_y in $G=(C,E)$, from theorem 3, p'_{xy} can be substituted by a new path p_{xy} satisfying $E(p_{xy}) \bigcap E_d = \varnothing$. Therefore, there must be a directed path from c_x to c_y in $G'=(C,E')$.

According to the conclusion 3 and theorem 4, we can conclude that: after deleting all edges in E_d, we can get the minimal equivalent connective graph $G'=(C,E')$ of $G=(C,E)$.

By the methods discussed in this section, we complete the semantic refinement for multi-ontologies with ontology mapping. During the refining process, all the steps in our methods can be performed in polynomial time obviously. That assures our method has high efficiency and is feasible to the practical cases. Although our discussion focuses on the concept mappings, the methods to deal with the relation mappings also can be derived from our approach easily.

5 Related works

Ontology mapping is considered as an important solution to the heterogeneity of multiple ontologies. Developing mappings has been the focus of a variety of works originating from diverse communities over a number of years. Some frameworks have been proposed. The IFF ontology mapping framework is based on the Information Flow Theory, which could describe the dynamic and stability about ontology [7]. In MAFRA framework, Maedche et al. used the semantic mapping to specify the translation between individuals and properties in different ontology [8].

Finding mappings is the key and also a difficult issue in ontology mapping. Till now, there is not an efficient way. To avoid the high cost of generating mappings manually, semi-automatic algorithm is necessary. These algorithms often use the natural language processing or machine learning, and discover the matching patterns by calculating the similarity. PROMPT algorithm first computes the concepts similarity, and then provides the possible mappings to the knowledge engineer to refine manually [9]. GLUE evaluates the similarity probability of the concepts or relations through instance learning [10]. The learning process of the approach uses multi-strategy learning mechanism. In [11], we also provided two semi-automatic methods for generating the mappings. First way discovers mappings by calculating the similarity based on the ontology structure, which includes synonym concept set, concept features and semantic neighborhood concept set. Another method is based on the mutual instances set of the ontologies and mappings which can be extracted by set operations. Although so much work has done for the ontology mapping, few of them consider the refinement after generating mappings.

In Ontology generation, discovering the relations between concepts is also an intricate task. Maedche and Staab used the idea of generalized association rule algorithm to detect non-taxonomic relations between concepts [12]. However, they didn't consider ontology refinement. Since the ontology learning algorithms often extract a number of taxonomic relations and general binary relations [13], errors and inconsistencies often exist, so the refining step is a necessary step in the ontology generation. Shamsfard and Barforoush proposed an automatic ontology building approach to extract lexical and ontological knowledge from Persian texts [14]. Their work involves refining and reorganizing ontology to eliminate redundancies, remove superfluities and unnecessary parts, but they did not give more detail algorithm. Lonsdale aims to build a new domain ontology reusing an existing big one [15]. Refining the results is the last step in his approach. But he does not give the detail algorithm too, and his method is manual.

Refining ontology is an important step both in the ontology building and multiple ontologies' applications. Its aim is to keep the ontologies sound and well organized. However, as far as we know, except our work, we have not found out any similar and specific work about on this issue. So we can believe that our work is one of the earliest approaches to investigate this issue.

6 Conclusions

Multiple ontologies often need to be accessed by the applications today. Ontology mappings provide the interoperating rules between multiple ontologies in order to reconcile these ontologies. However, introducing the mappings may break the soundness and balance of the original ontologies. And some mappings may be redundant or erroneous, which

would cause the multiple inconsistent and redundant. For the purpose to deal with the problem, this paper proposes an approach to eliminate the redundancies and errors in multiple ontologies. First, to solve the problem more conveniently, we classify the mappings into eleven kinds and model the multiple ontologies as graph. Then we analyze the changes of graph when introducing different kinds of mappings respectively and orderly. Following refining step includes semantic checking and semantic refinement. The former keeps the multiple ontologies sound, and the latter assures that the multiple ontologies are irredundant. Since this method is efficient, so it is feasible in the distributed environments.

We are implementing our methods as a tool in a system, whose aim is to solve the applications based on multiple ontologies. We have tried to use our refining method to refine the ontologies when extracting sub-ontology from the multiple ontologies [16]. The method works well, and the results are good. Additionally, we believe our approach is of benefit to the ontology generation as well, especially when the generation derived from ontology learning, our method can help to refine the ontology, provide suggestions for the knowledge engineers when finding the cycles, and improve the quality of the ontology.

References

[1] Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, v.5, p.199-220, 1993.

[2] Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Scientific American*, v.284, p.34-43, 2001.

[3] Wang, P; Xu, B. W.; Lu, J. J.; Li, Y. H. & Kang, D. Z. Using Bridge Ontology for Detailed Semantic Annotation Based on Multi-ontologies. *International Journal of Electronics and Computer Science*, v.6(2), p.19-29, 2004.

[4] Wache, H.; Vögele, T.; Visser, U.; et al. Ontology-based integration of information- a survey of existing approaches. In: *IJCAI'01 Workshop: Ontologies and Information Sharing*, p.108-117, 2001.

[5] Kalfoglou, Y.; Schorlemmer, M. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, v. 18(1), p.1-31, 2003.

[6] Klein, M. Combining and relating ontologies: an analysis of problems and solutions. In: *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, 2001.

[7] Kent, R. The information flow foundation for conceptual knowledge organization. In: *Proc. of the 6th International Conference of the International Society for Knowledge Organization*, 2000.

[8] Maedche, A.; Motik, B.; Silva, N. & Volz, R. MAFRA - A mapping framework for distributed ontologies. In: *Proc. of the 13th European Conference on Knowledge Engineering and Knowledge Management, EKAW-2002*, Madrid, Spain, 2002.

[9] Noy, N. F.; Musen, M. PROMPT: algorithm and tool for automated ontology merging and alignment. In: *Proc. of the 17th National Conference on Artificial Intelligence (AAAI'00)*, 2000.

[10] Ehrig, M.; Staab, S. QOM-Quick Ontology Mapping. In: *International Semantic Web Conference*, Hiroshima, Japan, LNCS, v.3298, p.683-697, 2004.

[11] Wang, P.; Xu, B. W.; Lu, J. J. et al. Theory and semi-automatic generation of bridge ontology in multi-ontologies environment. In: *On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops*, Lecture Notes in Computer Science, v.3292, p.763-767, 2004.

[12] Maedche, A.; Staab, S. Discovering Conceptual Relations from Text. In: *Proc. of the 14th European Conference on Artificial Intelligence*, Berlin, 2000.

[13] Maedche, A.; Staab, S. *Ontology Learning*. Handbook on Ontologies, Springer, p.173-190, 2004.

[14] Shamsfard, M.; Barforoush, A. A. Learning ontologies from natural language texts. *Int. J. Human-Computer Studies*, v.60, p.17-63, 2004.

[15] Lonsdale, D.; Ding, Y.; et al. Peppering knowledge sources with SALT: boosting conceptual content for ontology generation. In: *Proc. of the AAAI Workshop on Semantic Web Meets Language Resources*, Edmonton, Alberta, Canada, 2002.

[16] Kang, D. Z.; Xu, B. W.; Lu, J. J.; Wang, P.; Li, Y. H. Extracting sub-ontology from multiple ontologies. In: *On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops*, Lecture Notes in Computer Science, v.3292, p.731-740, 2004.