Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp901-906)

A New Method of Building Ontology Using Inheritance

ZHONG MING[1], SHUBIN CAI[2], SHIXIAN LI[2] and XINHONG ZENG[1]

[1] Information Engineering Faculty, Shenzhen University, Shenzhen, 518060, China
[2] Computer Science Department, SUN Yat-sen University, 510275, Guangzhou, China

Abstract: Ontology is the explicit representation of domain concept model. It can well express the complicated relationships between objects. Since formal ontology is the foundation of domain knowledge sharing and reusing, the formal ontology and associated concepts are stated in this paper. Previously research on ontology construction is high-cost and time-consuming, a new method of building ontology using inheritance is presented. The constructive algorithm of ontology inheritance is investigated to make the construction of ontology in an engineering way and to improve the quality of ontology. The methods to deal with conflicts and inconsistencies due to multi-inheritance and inheritance with exceptions are proposed.

Key-Words: Ontology, Ontology Building, Ontology Construction, Inheritance

1 Introduction

In the computer science community, more and more researches on ontology are reported now. Neches[2] defines ontology as "An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary". Gruber gives an more prevalent definition, i.e. "An ontology is a specification of a conceptualization." Ontology is now gaining a specific role in areas such as Artificial Intelligence, Computational Linguistics, and Databases. Its importance has been recognized in fields as diverse as knowledge engineering, knowledge representation, qualitative modeling, language engineering, database design, information integration, object-oriented analysis, information design.

The accumulated ontologies gives rise to the need for a suitable engineering method to organize, share and reuse these ontologies because creating ontologies is cost and time-consuming. Few ontologies in DAML Ontology Library share or reuse the same definition by now. It's not easy to share or reuse them. Our approach is to develop an ontology knowledge base, which focus on the mechanism to deal with inheritance and maintenance of ontologies. The inheritance mechanism of this ontology knowledge base consists of three layer of inheritance, i.e. the inheritance in attribute and method layer, the object layer and the ontology layer. As shown in figure 1.1.

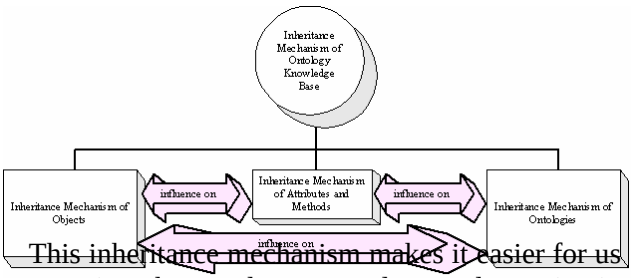

Fig. 1.1

This inheritance mechanism makes it easier for us to organize, share and reuse ontology and to maintain the ontology knowledge base.

Section 2 introduces the related work. Some formal definitions such as ontology, ontology skeleton, domain model etc will be given in Section 3. The inheritance graph of ontology will be elaborated on in section 4. Section 5 introduces the method of creating ontology using inheritance mechanism. Some examples are presented in section 6. Finally, we get some conclusions.

2 Related Work

A genetic approach for building ontology was presented by R.LU[1]. This methodology uses the semantic relevance of existing domain models and domain ontologies, and proposes the possibility of building ontologies following the view of semantic match. The basic operations are selection, clone, mutation, crossover synthesis transgenic. Since the early 90's, we took part in the research leaded by Professor R.LU engaged in a national key project, i.e., the Eagle project[12], which is a part of the Jade Bird project. In this project, we have taken a knowledge-based approach. This approach is aiming at automatically generating information systems. Ontology is a key concept for the knowledge base.

Ying Dong and Mingshu Li presented another method TEMPPLET[2], which combing two words "template" and "applet" together. Firstly it follows a "template ontology" of the domain; then fitting "applet ontology" into the chosen template when customizing. Thomas R.Gruber suggests a set of criteria to guide the development of ontologies[3]. Mike Uschold and Martin King introduce their methodology for building ontologies in [4]. They envisage a comprehensive methodology for developing ontologies to include 4 stages, i.e. Identify Purpose, Building the Ontology, Evaluation and Documentation.

Our approach is somewhat different from those approaches mentioned above. We propose a new method of creating ontology using inheritance. One of our purposes to develop an ontology knowledge base is to construct new ontology based on inheritance mechanism. Thus, the inheritance mechanism considered not only support inheritance exception and multi-inheritance, but also have methods to deal with conflicts and inconsistence caused by inheritance exception and multi-inheritance.

3 Formal Definition of Ontology

3.1 Main Idea About Ontology

We start our research based on following ideas[1]:

1. Relations is Independent Knowledge Units.
2. Organize the Objects in Ontology
3. Take Objects as Basic Elements of an Otology.
4. Let Ontology Form Inheritance Hierarchies
5. Allow Ontology to be Nested

Based on above ideas, the formal definitions about ontology is carefully defined in [1]. We will cites the most important ones here

3.2 Formal Definition of Ontology

Definition 3.1 Ontology Skeleton

An ontology skeleton (D, P, U, V, H) is a connected, finitely nested acyclic cybergraph[1], where the cardinality of U∪V is at least 2. The descriptors in D are called attributes. The processors in P are called methods. The simple vertexes in U are called object skeletons. The complex vertexes in V are ontology skeletons themselves and the hyperlanes in H are relations among the simple and complex vertices.

Definition 3.2 Ontology

If $O_1=(D_1, P_1, U_1, V_1, H_1)$is an ontology skeleton. Replace the elements in D_1 and P_1 by real attributes and methods such as those in an object. Replace the ontology skeletons in V_1 and H_1 by corresponding ontologies. The result is called an ontology

The ontology we use here is different from other ontologies. An ontology can have its own inheritance relationships other than import from other ontology.

Definition 3.3 Acyclic Cybergraph with relevance Degrees

A Generalized acyclic cybergraph is a sixtuple (D, P, U, V, H, R), where D, P and U are the same as in definition 1. H is a set of hyperplane, where each node of each hyperplane is a simple node or a compound node. A simple node is an element of U∪V. A compound node may be an and-node and(S) or an or-node or(T), where S and T are subsets of U∪V. An element of V is itself a generalized acyclic cybergraph.

R is a mapping. It maps each member m of H to a decimal number ranged from 0 to 1, called relevance degree of m with respect to this generalized acyclic cybergraph. Further, it maps each node n of each member m of H to a decimal number ranged from 0 to 1, called irrelevance degree of n with respect to m.

Each vertex v in an and-node has the relevance degree 1 with respect to the node it belongs. Each vertex v in an or-node has the quasi relevance degree 1 with respect to the node it belongs. That means, none of the vertexes contained in an or-node is necessary to that node (and to the hyperplane it belongs). But the set of vertexes in an or-node as a whole is necessary to that node. If h has the relevance degree d with respect to some entity E, then v has the quasi irrelevance degree d with respect to E.

Definition 3.4 Relevance Degree

Denote the relevance degree of x with respect to y as RD(x,y), the quasi relevance degree of x with respect to y as QRD(x,y) and the lower bound as LB. Then we have:

(1) min(RD(x,y)，RD(y,z)) is a LB（RD(x,z)）

(2) The above point is still valid if we replace RD(x,y)with LB(RD(x,y)) and/or RD(y,z) with LB(RD(y,z)).

(3) For any x contained in y

RD(x,y)=max{l|l is a LB(RD(x,y))}

(4) Min(QRD(x,y),QRD(y,z)) is a LB(QRD(x,z)).

(5) The point 4 is still valid if we replace QRD(x,y) with LB(QRD(x,y)) and/or QRD(y,z) with LB(QRD(y,z)).

(6) The point 4 is still valid if we replace QRD(x,y) with RD(x,y) or LB(RD(x,y)), or replace QRD(y,z) with RD(y,z) or LB(RD(y,z)). But not both.

(7) QRD(x,y)=max{l|l is a LB(QRD(x,y))}

(8) In any other case, the relevance degree of x with respect to y is 0.

(9) RD(x,y)=1 if y is an ancestor of x.

If RD(x,y)=d,(0≤d≤1),then the larger the relevance degree d is, the more is the relevance. D=1 means x is necessary for y.

Definition 3.5 Weighted Ontology Skeleton

A weighted ontology skeleton is a connected, finitely nested generalized acyclic cybergraph, where the cardinality of U∪V is at least 2. The descriptors are attributes. The processors are methods. The simple nodes are objects and/or object classed. The compound nodes are weighted ontology skeletons. And the hyperplanes are the relations among members of U∪V.

4 Ontology in Inheritance Graph

4.1 Nodes in Inheritance Graph

Based on the formal ontology definition, the inheritance of ontology is addressed in directive graph. The nodes are classified into 7 types:

Attributes Node It's an element of D in Ontology.

Method Node It's an element of P in Ontology.

Object Node It's an element of V in Ontology. An Object Node can have inner and outer structures, Which indicate its own Attributes and Methods. The inner attributes and methods are encapsulated. Other nodes know nothing about them, and descendent nodes can't inherit them with exception. The outer attributes and methods are sensable to others and descendent nodes may inherit them with exception.

Ontology Node It's an element of U in Ontology. An Ontology Node also has inner and outer structures, and the difference between which is as the same of Object Nodes.

Relation Node It's an element of H in Ontology. An Relation Node have inner structure, which indicates the cardinality of relations, the name of relations, the name of entities attached with the relations, and the constraint of the relations.

And-Node It's an and-node in Weighted Ontology.

Or-Node: It's an or-node in Weighted Ontology.

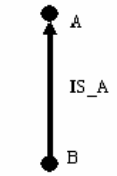

Figure 2.1

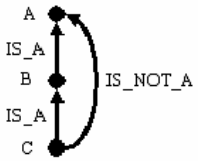

Figure 2.2

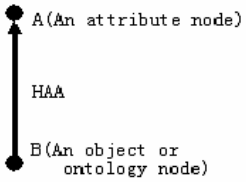

Figure 2.3

4.2 Directive Edges in Inheritance Graph

There are 7 kinds of directive edges, IS_A, IS_NOT_A, HAA (Has_An_Attribute), HAM (Has_A_Method), HAObj (Has_An_Object), HAOnt (Has_An_Ontology) and HAR (Has_A_Relation) in inheritance grahp representing the correspoding inheritance and containment relationships between nodes in an ontology. Figure 2.1 to 2.7 is simple examples of them respectively.

A virtual node represent the define area of a relation. It has no ownership with others and is of the same type of the nodes it has the IS_A edge point to. Edge of relation represent other relations than the above 7 types of relations between nodes. It has the same name as the name of the relation node it points from. Every edge of relation points to virtual object (ontology) node. Assume there is a relation node R and have a relation edge point to a virtual object node VObj, The define area (DA as short) of VObj in R is consist of all the nodes x that VObj IS_A x and x's descendent nodes. In figure 2.8, it represent a relation R(A,B,C),DA(A)={A'}, DA(B)={B'}and DA(C)={C',D}. If DA(A) is not empty, then A is called valid for relation R, else A is called invalid for relation R. If all the virtual object (ontology) nodes that R point to is valid, then relation R is called valid. Otherwise R is invalid.

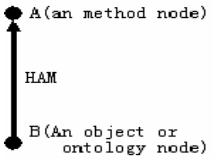

Figure 2.4

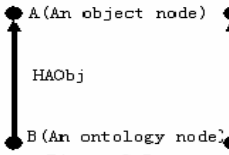

Figure 2.5

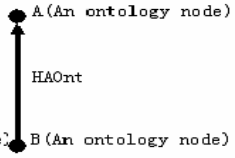

Figure 2.6

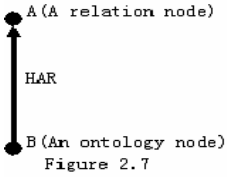

Figure 2.7

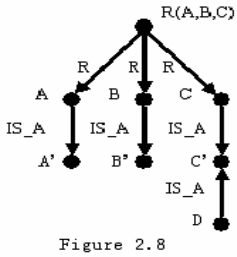

Figure 2.8

As shown in figure 2.9 and 2.10, the representation of "And-Node" and "Or-Node" is the same as the representation of relation R above.

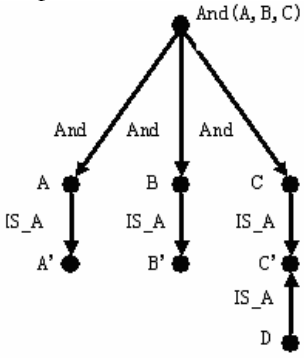

Figure 2.9

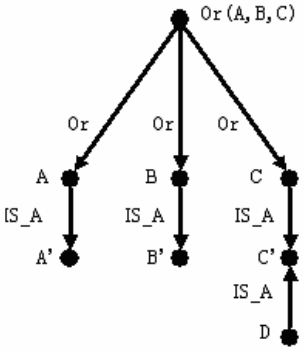

Figure 2.10

Every edge has a weight, which equals to the Relevance Degree. The weight of an IS_A edge is 1; weight of all ownership edges from object node is 1.

The Ontology Knowledge Base as a whole becomes a huge inheritance graph. And we can investigate the inheritance of ontology, construction of ontology and maintenance of ontology knowledge base based on this inheritance graph.

4.3 Inheritance Concepts

In the inheritance graph, we have no copies of what the ancestor ontologies have as elements in the

descendent ontology. That is to say, a descendent ontology node only owns nodes that have different relations from it and its ancestor ontologies nodes. We can see Figure 2.11 for example. OntoA has an attribute A. OntoB IS_A OntoA and has an attribute B, and implicitily has attribute A from inheritance .

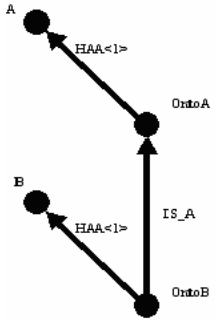

Figure 2.11

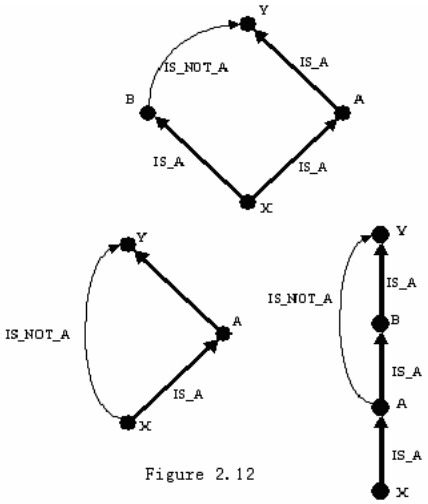

Figure 2.12

It is very easy to construct a new ontology by inheritance in the inheritance graph, though it is not easy to find out what descendent ontology owns since IS_NOT_A exists. Before showing what the retrieve algorithm is, here are notions used in inheritance graph to develop the retrieve algorithm.

Definition 4.1 Inheritance Path

In Inheritance Graph G, a path consisted of IS_A edges from node x to node y is called an inheritance path from x to y. We used an order $<v_1,\cdots,v_n>$ to represent an inheritance path, and this order is called an inheritance order.

Definition 4.2 Contradict

Inheritance Graph G contradicts inheritance order $<x_1, \cdots, x_n>$ iff there $\exists i$, such that IS_NOT_A(x_1, x_i) $\in$G($1<i<=n$).

The definition of contradict is used to enable the exception of inheritance.

Definition 4.3 intermediary

In Inheritance Graph G, node y is an intermediary of inheritance order $<x_1, \cdots, x_i, \cdots, x_n\rangle$ iff there $\exists i$, such that $y= x_i$ or G contains a order $< x_1, \cdots, x_i, y_1, \cdots, y_m, x_{i+1}\rangle$ and $\exists j$, $y= y_j$, $1\leqslant j\leqslant m$ and $1\leqslant i\leqslant n$。

Definition 4.4 preclude

Inheritance Graph G preclude order $\sigma = \langle x_1, \cdots, x_n \rangle$ iff $\exists y$, such that IS_NOT_A(y, x_n), y is an intermediary of σ.

Definition 4.5 Inheritance Conclusion Set

The inheritance conclusion set of inheritance graph G C(G) is all the inheritance orders <x,y> and orders $<x, \cdots, y>$ that are not contradicted or precluded by G.

Definition 4.6 Ancestors Set

The Ancestors Set of node x is AS(x)={y|$\exists<x, \cdots, y>\in$ C(G)};

Definition 4.7 Inheritance Hierarchy

Assume that inheritance order set IOS(x, x_n)={$\forall<x, \cdots, x_n>\in$C(G)} and intermediary set IS(x, x_n)={y|$\exists\ \sigma \in$ IOS(x, x_n),y is a intermediary of σ }, the inheritance hierarchy of x_n respect to x IH(x_n, x)=1+Max(IH(y, x),$\forall y\in$ IS(x, x_n)). Especially, if there exists no intermediary y, then IH(x_n, x)=1.

Defition 4.8 Inheritance Hierarchy Conflict:

For $\forall$y1,y2, y1,y2$\in$ AS(x), if $\exists$ IS_NOT_A(y1,y2), then we call it an inheritance hierarchy conflict.

The inheritance hierarchy conflict occurs when we say X IS_A Y and X IS_NOT_A Y both. But what figure 2.13 shows is not a conflict. Because Y is not in AS(X). When detecting an inheritance hierarchy conflict, we delete the one that has a larger IH value from AS(x) and inform the user of this. If IH values are equal, we delete one randomly, and warn the user. After solving the inheritance hierarchy conflict, we call AS(X) as Modified AS(X).(MAS(X) for short). Similarly, we defined Modified Descendent Set(X) as MAS(X),which means the set of descendents of x.

Using definitions above, the algorithm to generate MAS(x) is implemented, called UpScan algorithm. Similarly, the algorithm to generate MDS(x) is called DownScan algorithm.

Definition 4.9 Inheritance Similarity Degree:

If y1 and y2 are nodes that x inherit from its ancestor nodes. If y2 in MAS(y1) and y1 inherit y2 without exception then ISD(y1,y2)="contain" (or "contained_by" in reverse. Otherwise, ISD(y1,y2)= (2*number of the same nodes x inherit from y1 and y2) / (number of nodes x inherit from y1+number of nodes x inherit from y2)*100%.

If ISD(y1,y2)= "contain" or "contained_by", nodes y1 and y2 will be merged automatically. The merging opeation is done as:.

Set up a new node y3

Link IS_A edges from y3 to y1 and y2

Link the same type of edge as that from x to y1 from x to y3

RD(y3,x)=Max(RD(y1,x),RD(y2,x).

RD(y1,x)=RD(y2,x)=0

4.4 Inheritance of Attribute and Method

The inheritance mechanism of ontology consists of three parts, inheritance of ontology, inheritance of object and inheritance of attribute and method, which have influence on each other.

The inheritance of attribute and method is very

simple; an example will explain it clearly. See figure 2.14. “English Name” and “Chinese Name” IS_A “Name”. What we want to know in this layer is that MAS(English Name)={Name}=MAS(Chinese Name). And MDS(name)={English Name, Chinese Name}. The inheritance mechanism of attribute and method layer is to generate MAS(X) and MDS(X) for any x is an attribute node or a method node.

4.5 Inheritance of Object

See figure 2.15. Using UpScan algorithm, we have MAS(Chinese Boy)={Person, Chinese}. In inheritance graph, An object has HAA edges point to its attributes and HAM to methods. Scaning all the HAA and HAM edges start from nodes in MAS(X), we get an attribute set of object X called ASObj(X) and a method set of object X called MSObj(X).

For $\forall a \in ASObj(X)$, if $\exists b$, ISD(a,b)= “contain” and $b \in$ ASObj(X), then delete b from ASObj(X). After doing this, the attribute set is called Modified Attribute Set of Object x. (MASObj(x) for short) For $\forall m \in MSObj$, if $\exists n$, ISD(m,n)= “contain” and $n \in$ MSObj, then delete n from MSObj. After doing this, the method set is called Modified Method Set of Object x. (MMSObj(x) for short) The algorithm to generate MASObj(x) and MMSObj(x) is called “object ownership relation scan”

Apply object ownership relation scan to figure 2.15, we get MASObj(Chinese Boy)={Chinese Name}. Attribute “Name” is in ASObj(Chinese Boy), but not in MASObj(Chinese Boy).

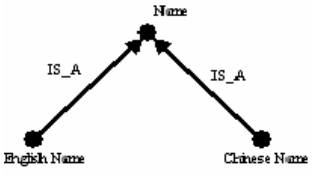

Figure 2.14

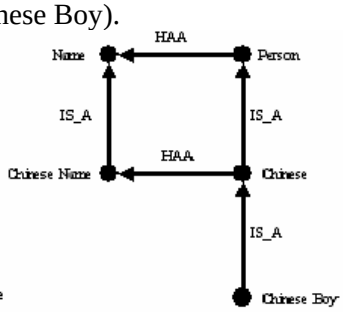

Figure 2.15

4.6 Inheritance of Ontology

The method used to find what an ontology has from inheritance mechanism is called “Ontology ownership relation scan algorithm”. It is as bellow:

After using UpScan algorithm to generate MAS(x),similar to the object ownership relation scan algorithm, we can get MASOnto(x) and MMSOnto(x),MObjS(x)(Modified Object Set of ontology) and MOntoS(x)(Modified Ontology Set of ontology) and RS(x)(Relation Set of ontology). The relation node form no inheritance hierarchy and there is no need to modify the relation set of ontology.

For every object node in MObjS(x), try object inheritance mechanism to generate what it has.

For every ontology node in MOntoS(x), try ontology inheritance mechanism recursively to generate what it has.

For every relation node in RS(x), try DownScan algorithm to find its Define Area. Checking changes of relevance degree.

The checking of changes of relevance degree is:

If x is a ontology node, for $\forall y1,y2$, $y1,y2 \in MAS(x)$,if $\exists z$ owned by x ,and $RD(z,y1) \neq RD(z,y2)$， if y1 is a intermediary of y2, then let RD(z,x)=RD(z,y1),otherwise if y2 is a intermediary of y1, then let RD(z,x)=RD(z,y2), otherwise it is an Inheritance Relevance Degree Conflict.

When inheritance relevance degree conflict occurs, we let RD(z,x)=MAX(RD(z,y1),RD(z,y2)) and inform the user.

5 Building Ontology

Building ontology using ontology inheritance mechanism shall follow the four steps below:

1. Search Ontologies to Be Inherited

We can search ontology by name, keyword or description (comment) or elements name of the desired ontology. The ontologies is orgainzed in inheritance hierarchy to make it easier for use.

2. Choose Ontologies to Be Inherited

And we can set the merge condition by x% or filter condition by RD d. The tool will merge nodes that have the ISD>x%, and at last, only show those nodes that have larger RD then d.

3. Modification on System-Generated Ontology

“Ontology ownership relation scan" algorithm is called to generate a temporary ontology. User makes modification on the temporary ontology to build their own ontology.

4. Save Modified Ontology to Knowledge-Base

According to the user’s modification, make changes to the inheritance graph. If user inserts an attribute y into ontology x, if y already exists in the base, then link a HAA edge from x to y, else a new node y is created and link the edge. If user change attribute z to y, then node y is created and link an IS_A edge from y to z, and a HAA edge from x to y. If user delete an attribute y, then a HAA edge from x to y is created with RD(y,x)=0. The modification of method, object, ontology and relation nodes is the same as attribute nodes.

6 Case Study

We will use inhertiance graphs to illustrate the construction procedure of ontology using inhertiance mechanism argued above.

Fig. 6.1 is example of inheritance with exception. NewOnto inheriting Demo while it IS_NOT_A DemoAncestor, which is the ancestor of Demo.

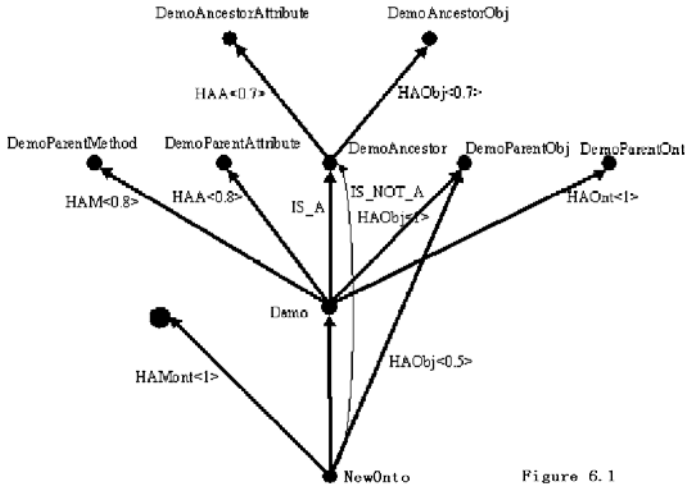

Figure 6.1

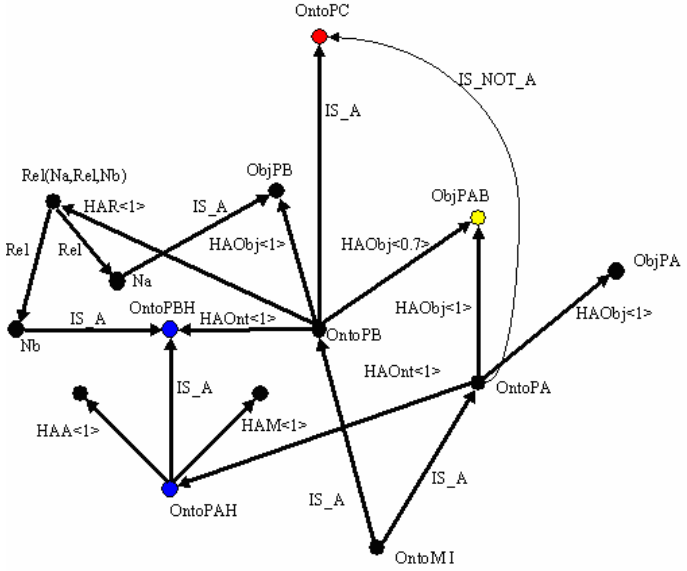

Figure 6.2

Fig. 6.2 is an ontology named "OntoMI" inherit "OntoPA"and "OntoPB". As we can see, "OntoPAH" IS_A "OntoPBH", so "OntoMI" should merge OntoPBH and OntoPAH into one ontology. Since OntoMI IS_A OntoPA and has HAObj<1> to ObjPAB, and OntoMI IS_A OntoPB and has HAObj<0.7> to ObjPAB, and because OntoPA and OntoPB isn't intermediate of each other, so it is a conflict. The default setting of the mechanism will choose the bigger one. OntoPC is an inheritance hierarchy conflict.

7 Conclusion

We organize multiple shared ontologies on the basis of the similarities that they conceptualise the common domain. Ontology clusters are then organized in a hierarchical fashion thus permitting ontologies to be described at different levels of abstraction. One want to create an ontology, he/she first search the library to find an ontology that most similar to his/her acquire. Then he can inherit the ontology and edit it. After his customization, he can save it to the library for further use.

A important feature of Inheritance Graph is that descendent nodes haven't copied what ancestors have. Thus, if the content of an ancestor node changed, the descendent nodes will know about it automatically. When the algorithm of ontology ownership relation scan is applied, the changes will show in the descendent nodes. The descendent nodes can effect the define area of a relation in ancestor nodes. When adding a new descendent node, the define area of relations which contain the ancestor nodes will extend. The descendent node will be included in the define area automatically. The result of this feature of inheritance graph is very useful. It makes the maintenance of knowledge base very easy.

When introducing the inheritance graph, we have mentioned that object nodes and ontology nodes have inner and outer structure, which are used to descript the content of the object and ontology. The inner structure is encapsulated, other nodes including the descendent nodes can not make any modification to the content of inner structure. The outer structure represents the share content of an object or ontology. The content in outer structure is represented as nodes, which allow difference between relations of it with ancestor nodes and descendent nodes.

Using the inheritance appraoch presented in this paper, the construction of ontolotgy is becoming more easier and less time-consuming.

8 Acknowledge

This research is supported by National Social Science Foundation of China (#05CTQ001), Guangdong Natural Science Foundation (#04011304), and Shenzhen Science Technology Plan (#200422)

References:

[1] R.LU, Z.JIN. Domain Modeling-Based Software Engineering. Kluwer Academic Publishers. 2000

[2] Jag Sodhi, Prince Sodhi. Software Reuse: Domain Analysis and Design Processes. McGraw-Hill Companies, Inc. 1998

[3] Jacobson I, Griss M, Josson P. Software Reuse: Architecture Process and Organization for Business Success. 1998

[4] Scott A. DeLoach, Thomas C.Hartrum. A Theory-Based Representation for Object-Oriented Domain Models. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.26,NO.6,JUNE 2000

[5] Trygve Reenskaug, P.Wold and O.A.Lehne, The OOram Software Engineering Method,1996