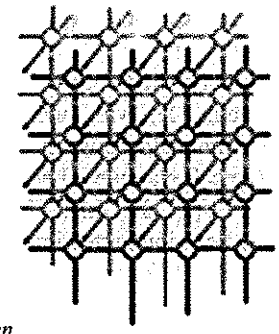
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2003; **15**:845–869 (DOI: 10.1002/cpe.732)

Enhancing OODB semantics to support browsing in an OODB vocabulary representation



Li-min Liu[1], James Geller[2,*,†] and Yehoshua Perl[2]

[1]*Department of Applied Mathematics, Chung Yuan Christian University, 22, Pu-Zen, Pu-Chung Li, Chung-Li, Taiwan, Republic of China*
[2]*Computer Science Department, New Jersey Institute of Technology, Newark, NJ 07102, U.S.A.*

SUMMARY

In previous work, we have modeled a vocabulary given as a semantic network by an object-oriented database (OODB). The OODB schema thus obtained provides a compact abstract view of the vocabulary. This enables the fast traversal of the vocabulary by a user. In the semantic network vocabulary, the IS-A relationships express the specialization hierarchy. In our OODB modeling of the vocabulary, the SUBCLASS relationship expresses the specialization hierarchy of the classes and supports the inheritance of their properties. A typical IS-A path in the vocabulary has a corresponding shorter SUBCLASS path in the OODB schema.

In this paper we expose several cases where the SUBCLASS hierarchy fails to fully correspond to the IS-A hierarchy of the vocabulary. In these cases there exist traversal paths in the semantic network for which there are no corresponding traversal paths in the OODB schema. The reason for this failure is the existence of some IS-A relationships between concepts of two classes, which are not connected by a SUBCLASS relationship. This phenomenon weakens the accuracy of our modeling.

To rectify the situation we introduce a new OODB semantic relationship IS-A′ to represent the existence of IS-A relationships between concepts of a pair of classes which are not connected via a SUBCLASS relationship. The resulting schema contains both SUBCLASS relationships and IS-A′ relationships which completely model the IS-A hierarchy of the vocabulary. We define a mixed-class level traversal path to contain either SUBCLASS or IS-A′ relationships. Consequently, each traversal path in the semantic network has a corresponding mixed traversal path in the OODB schema. Hence the introduction of the semantic OODB IS-A′ relationship improves the modeling of semantic network vocabularies by OODBs. **Copyright © 2003 John Wiley & Sons, Ltd.**

KEY WORDS: object-oriented databases; object-oriented models; object-oriented systems; knowledge representation; database models; vocabulary systems; object-oriented hierarchical relationships; object-oriented semantic relationships

*Correspondence to: James Geller, Computer Science Department, New Jersey Institute of Technology, Newark, NJ 07102, U.S.A.
†E-mail: geller@la.njit.edu

Contract/grant sponsor: HIIT; contract/grant number: #70NANB5H1011

Received 22 February 2002
Revised 30 October 2002
Accepted 2 November 2002

1. INTRODUCTION

A controlled vocabulary (CV) is a software system that unifies the terminology of large application domains [1]. With such a system in an enterprise, costly and time-consuming translation tasks can be eliminated between different organizations and software applications. The healthcare domain is a typical example where several large vocabularies have been built such as MeSH [2], CPM93 [3], CPT98 [4], SNOMED [5], ICD9-CM [6], the Medical Entities Dictionary (MED) [7] (all of which have been integrated into the Unified Medical Language System (UMLS) [8,9]) and the GALEN Core Model [10] (expressed in GRAIL [11]).

We have developed a technique for modeling a CV as an object-oriented database (OODB) [12,13], which we call an object-oriented vocabulary repository (OOVR) [14,15]. Using our methodology, we have created OOVRs based on the MED and InterMED [16], a derivative of the MED. Both OOVRs are represented in ONTOS DB/Explorer [17,18], a commercial OODB management system. The database schema of an OOVR, called the OOVR schema, gives users an abstract view of the vocabulary, compared to the tens of thousands of concepts in the vocabulary itself[‡]. An OOVR schema can be utilized by different types of vocabulary users [19]. For instance, casual users can use it to browse the contents of the vocabulary. A vocabulary administrator, on the other hand, can use this abstract view to maintain the vocabulary. In this paper, we identify several cases in which the OODB paradigm cannot fully capture the semantics of the IS-A relationships in the semantic network. As a result, the OOVR schema gives users a partial abstract view of the CV. In order to resolve this problem, we introduce a new type of relationship called IS-A′ for the OOVR schema. IS-A′ relationships capture those IS-A links of the semantic network that cannot be properly reflected by the SUBCLASS relationships of the OOVR schema. With the IS-A′ relationship, users obtain a more precise abstract view of the vocabulary.

The remainder of this paper is organized as follows. In Section 2, we describe our technique of mapping a vocabulary into an OOVR. OOVR schema browsing is discussed in Section 3. Section 4 describes the difficulties that can arise in certain special cases of schema browsing in the OOVR. In Section 5, we introduce the IS-A′ semantic relationship for an OOVR schema. Then, in Section 6, we demonstrate how IS-A′ modeling can overcome the difficulties of Section 4. Conclusions follow in Section 7.

2. OODB REPRESENTATION OF A VOCABULARY

In [14,15,20], we presented algorithms for deriving an OODB schema for a given vocabulary represented as a semantic network. A semantic network consists of concepts connected via IS-A relationships. Each concept may have properties of two kinds, attributes and relationships. Attributes have values of data types (such as an integer or text string). Relationships have values which are references to other concepts in the vocabulary. The set of properties of a concept **x** is denoted by P(**x**). We refer to the concepts in the IS-A hierarchy with family terms such as child, parent, ancestor, etc. A concept inherits the properties of its parents along the IS-A links.

[‡]For the 56 000 concepts of the MED, the MED OOVR schema contains 124 classes.

The hierarchy of an OODB schema consists of classes connected via SUBCLASS relationships. We refer to the classes in the OODB hierarchy with terms such as subclass and superclass, composed, if necessary, with family terminology. For instance, 'subclass descendants' of a class A refers to every class from which there exists a path of SUBCLASS relationships to the class A. The extent $\mathrm{E}(A)$ of a class A is the set of its objects.

In our OODB representation of a vocabulary, the extent of a class consists of objects with the same structure and similar semantics. The structure of an object is its set of properties. Hence, all the objects of a class A share the same set of properties, denoted by $\mathrm{P}(A)$. A root $\mathbf{r}_A$ of a class A is an object of the class whose parent objects are not in A. In our OODB representation [20], each class has a unique root; all the objects of a class are descendants of its root. Hence, each object in the class represents a specification of the more general object of the root. For example, if the root of a class is the object **Virus**, then all the objects of the class will represent specific viruses and specific families of viruses. Hence, all the objects of a class have semantics which are similar to the semantics of the root. To capture the semantics of a class, it is named after its root object. Hence, in our OODB representation, the objects of a class show the same structure and similar semantics as required for an OODB class.

To complete the specification of our OODB representation of a vocabulary, we need also to define the SUBCLASS relationships between classes. The SUBCLASS relationships form the hierarchy of the OODB schema and enable the inheritance of properties among classes.

A class A with a root $\mathbf{r}_A$ is a SUBCLASS of a class B if there is a concept $\mathbf{b}$ in B, such that in the semantic network $\mathbf{r}_A$ IS-A $\mathbf{b}$. In this way, the root $\mathbf{r}_A$ inherits all the properties of the class B. Those properties are further inherited to all concepts of A. Thus, we have class A inherits the properties of class B.

We have distinguished three different kinds of classes [20]:

1. property-introducing classes;
2. intersection classes; and
3. articulation classes.

These three kinds of classes are defined by the characteristics of their roots.

Definition 1. (Property-introducing class) A class A is a property-introducing class if a property which was not defined for any of its superclasses is introduced at the root object $\mathbf{r}_A$. That is, $\exists \mathbf{x} \in \mathrm{P}(\mathbf{r}_A)$ such that for all B_i where A SUBCLASS B_i, $\mathbf{x} \notin \mathrm{P}(B_i)$.

Definition 2. (Property-introducing object) The root of a property-introducing class is a property-introducing object.

Definition 3. (Intersection class) A class A is an intersection class with j superclasses $B_1, \ldots, B_j$, $j > 1$, if it is not a property-introducing class and its root $\mathbf{r}_A$ has parents $\mathbf{b}_i$ in all superclasses B_i of A, $i = 1, \ldots, j$, all of which have a different set of properties to $\mathbf{r}_A$. That is, for all i, $1 \leq i \leq j$, such that A SUBCLASS B_i, $\exists \mathbf{b}_i$ such that $\mathbf{b}_i \in B_i$ and $\mathrm{P}(\mathbf{r}_A) \neq \mathrm{P}(\mathbf{b}_i)$.

Definition 4. (Intersection object) The root of an intersection class is an intersection object.

For both the property-introducing class and the intersection class, a class A satisfies the following condition: for all B_i such that A SUBCLASS B_i, $\mathrm{P}(A) \neq \mathrm{P}(B_i)$.

However, the reasons for this are different. For a property-introducing class, this is because a new property is introduced at the root of the class. For an intersection class, no new properties are introduced

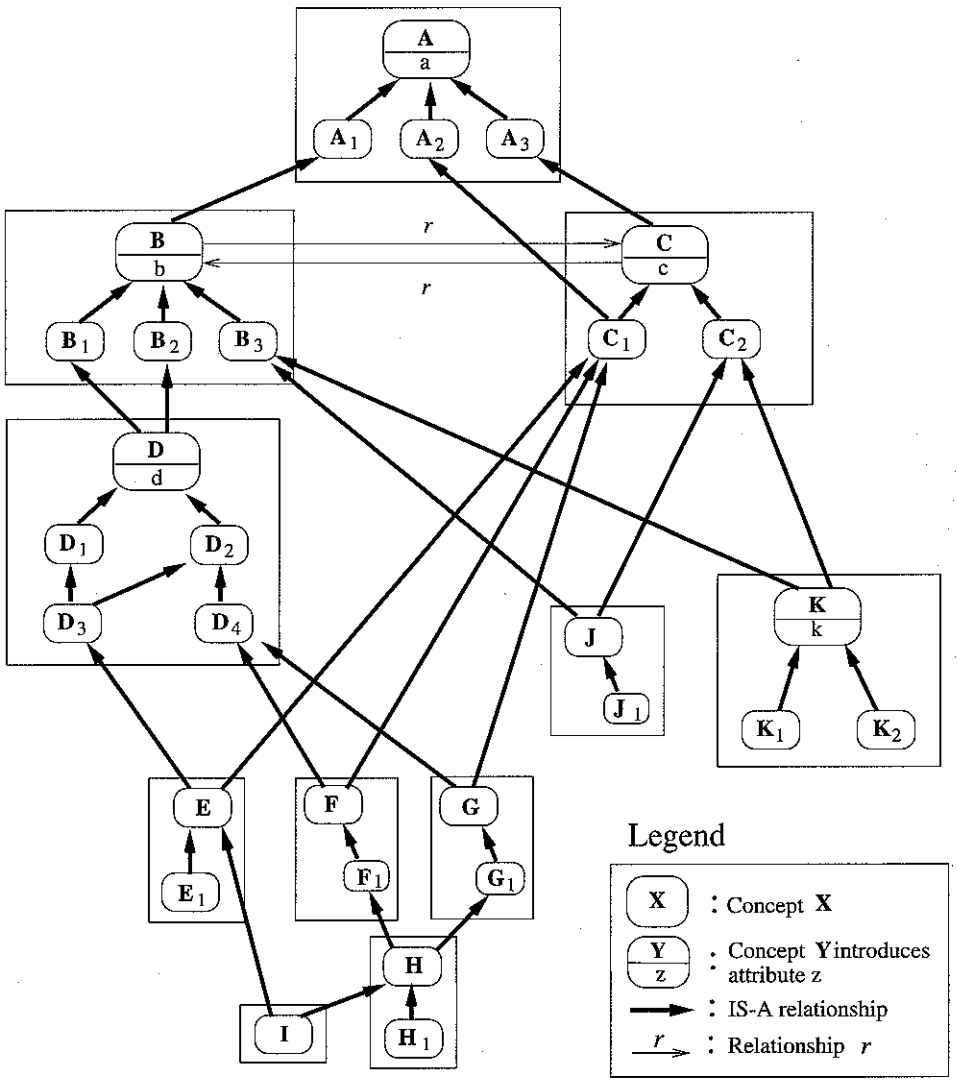


Figure 1. A small CV with 30 concepts.

at the root. However, the class inherits properties from different superclasses in such a way that it has more properties than any of the superclasses.

In Figure 1, we present a small CV which illustrates all these definitions and cases. This figure will also be used later to illustrate the definitions of the articulation object and the articulation class.

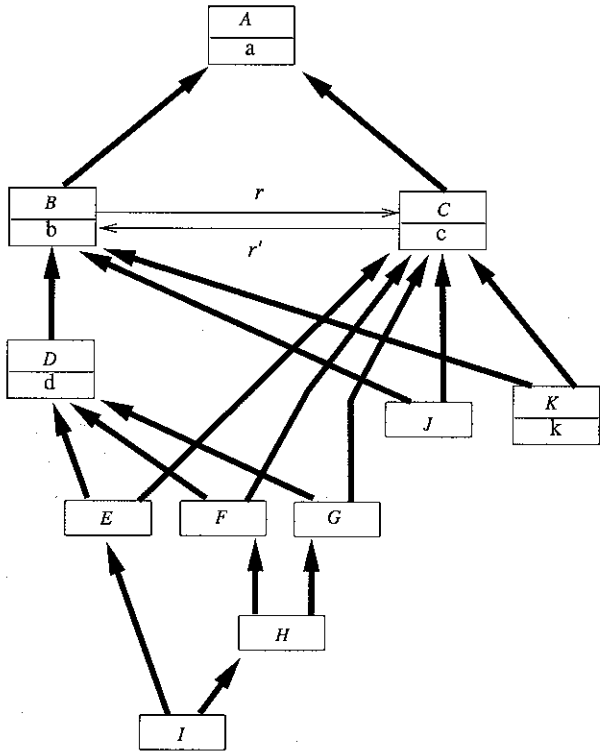


Figure 2. The OODB schema for the CV shown in Figure 1.

The objects of a class are enclosed within a box. For example, the objects **A**, $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{A}_3$, enclosed within the top block, all belong to the class A, named after its root **A**. The objects **A**, **B**, **C**, **D** and **K** are property-introducing objects. The objects **E**, **F**, **G** and **J** are intersection objects. The OODB schema for this CV is shown in Figure 2. The classes A, B, C, D and K are property-introducing classes which introduce the attributes a, b, c, d and k, respectively. The classes B and C further introduce the relationship r and its inverse r', respectively. $\mathrm{P}(A) = \{a\}$, $\mathrm{P}(B) = \{a, b, r\}$, $\mathrm{P}(C) = \{a, c, r'\}$, $\mathrm{P}(D) = \{a, b, d, r\}$, $\mathrm{P}(K) = \{a, b, c, k, r, r'\}$. Class J is an intersection class, where $\mathrm{P}(J) = \{a, b, c, r, r'\}$. The classes E, F and G are also intersection classes and $\mathrm{P}(E) = \mathrm{P}(F) = \mathrm{P}(G) = \{a, b, c, d, r, r'\}$.

We will now discuss the third kind of class, called the articulation class. An articulation class has the same set of properties as its parent classes, which may be intersection classes or other articulation classes. This implies that for each articulation class there exist several intersection classes or articulation classes with the same sets of properties. The reason why these intersection classes are separated is that their extents do not share similar semantics. Each such intersection class has a different

root, although all these roots share the same set of properties. For example, the three intersection classes E, F and G share the same set of properties, but have different roots.

Definition 5. (Articulation class) A class A is an articulation class if it is neither a property-introducing class nor an intersection class and its root is an articulation object.

The notion of an articulation object and an algorithm for determining the articulation objects and the objects of an articulation class are complex and are fully discussed in [20]. In this paper, we avoid repeating the exact definition for reasons of brevity and will provide an intuitive explanation and an example to clarify the nature of articulation classes and objects.

As mentioned before, the need for an articulation class only arises when there are several intersection classes B_i, $1 \leq i \leq j$, with identical property sets. We denote by DSN($\mathbf{x}$) the set of all objects which are descendants of object $\mathbf{x}$ in the CV.

In case $\text{DSN}(\mathbf{r}_{B_i}) \cap \text{DSN}(\mathbf{r}_{B_k}) = \emptyset$ for all pairs (i, k) such that $1 \leq i < k \leq j$ (that is, there are no objects in the CV which are descendants of more than one of the roots $\mathbf{r}_{B_1}, \ldots, \mathbf{r}_{B_j}$), there is no need to introduce an articulation class with the same set of properties. The need for a new articulation class arises when two or more such roots share common descendants. If, for example, $\mathbf{r}_{B_1}$ and $\mathbf{r}_{B_2}$ have a common descendant, then it cannot simultaneously belong to both classes B_1 and B_2. This is due to the OODB rule that an object cannot belong to more than one class. To resolve this conflict, we pick an object $\mathbf{X}$ whose parents do not have this conflict; that is, they are descendants of only one of the roots $\mathbf{r}_{B_1}$ and $\mathbf{r}_{B_2}$. Such an object $\mathbf{X}$ is called an articulation object and its own descendants which are in $\text{DSN}(\mathbf{r}_{B_1}) \cap \text{DSN}(\mathbf{r}_{B_2})$ are put in the articulation class rooted by $\mathbf{X}$. (Note that $\mathbf{X}$ is not a property-introducing object since, in such a case, it would not be a candidate object for the intersection classes B_1 and B_2, but would start its own property-introducing class.) In Figure 2, class H is an articulation class, since the object $\mathbf{H} \in \text{DSN}(\mathbf{F}) \cap \text{DSN}(\mathbf{G})$ is an articulation object in Figure 1.

More articulation classes may be necessary as two articulation objects (or one articulation object and one intersection object) $\mathbf{A}_1$ and $\mathbf{A}_2$ may have joint descendants. In such a case, a new articulation class to contain these joint descendants needs to be defined. Class I, in Figure 2, is one such articulation class since the object $\mathbf{I} \in \text{DSN}(\mathbf{E}) \cap \text{DSN}(\mathbf{H})$ is an articulation object in Figure 1. In [20], we provided a recursive definition and an algorithm to identify all articulation objects and articulation classes for a given CV.

3. VOCABULARY SCHEMA BROWSING

In order to demonstrate the browsing of the CV and its OODB schema, we introduce two notions: an object-level browsing path and a schema-level browsing path.

Definition 6. (Object-level browsing path) An object-level browsing path is a sequence of objects $\text{P}_C = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$ such that $\mathbf{c}_{i+1}$ IS-A $\mathbf{c}_i$, $1 \leq i < n$, in the semantic net.

Definition 7. (Schema-level browsing path) A schema-level browsing path is a sequence of classes $\text{P}_A = (A_1, A_2, \ldots, A_k)$ such that class A_{i+1} is a SUBCLASS of class A_i, $1 \leq i < k$.

Note that we restrict the SUBCLASS relationships to direct SUBCLASS associations for the schema-level browsing path. In Figure 2, the class D is a SUBCLASS of the class B and the class B is a SUBCLASS of the class A. Thus, there is a schema-level browsing path from A to D: (A, B, D).

However, (D, A) is not considered a valid browsing path even though the class D is a SUBCLASS of the class A by transitivity. The reason why we rule out transitively implied SUBCLASS relationships is because this may result in paths which omit certain classes which express the specific semantics of the path in the OODB schema.

Now we can define the conditions that have to hold for all the SUBCLASS relationships to properly reflect the IS-A relationships of the CV. For every browsing path $P_C = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n)$ on the object level of a CV, there must exist a parallel browsing path $P_A = (A_1, A_2, \ldots, A_k)$ on the schema level which satisfies the following conditions:

1. the object $\mathbf{c}_1$ is an instance of class A_1;
2. the object $\mathbf{c}_n$ is an instance of class A_k;
3. there exists a partition of P_C into disjoint consecutive subpaths, say $(\mathbf{c}_{j_1}, \ldots, \mathbf{c}_{j_e})$, which are paths in the induced subnetwork of class A_j, $1 \leq j \leq k$.

For example, in Figure 1, the object-level browsing path $(\mathbf{A}, \mathbf{A}_1, \mathbf{B}, \mathbf{B}_2, \mathbf{D}, \mathbf{D}_2, \mathbf{D}_4, \mathbf{F}, \mathbf{F}_1, \mathbf{H}, \mathbf{H}_1)$ has a parallel schema-level browsing path (A, B, D, F, H) in Figure 2. While browsing a concept-level path, the user needs to scan the children of each object to pick the correct concept with which to continue. A similar situation exists when scanning the subclasses of a class in schema-level browsing. The object-level path requires a scan of a total of 18 children, while the class path requires the scanning of only nine classes and one IS-A relationship from $\mathbf{H}$ to $\mathbf{H}_1$, as the browsing switches to the object level when reading the last class. This example demonstrates that, typically, a schema-level browsing path is shorter and requires less scanning than an object-level browsing path (see [20] for a detailed discussion).

4. PROBLEMS FOR UTILIZING THE OODB PARADIGM TO MODEL A CV

In the previous section, we demonstrated the acceleration and simplification of browsing a CV due to utilizing a schema-level browsing path. However, in some special cases which we shall see, the OODB schema derived for the CV does not support all possible object level browsing paths. The reason for these problems is a deficiency of SUBCLASS relationships representing all the IS-A relationships in the semantic network. In this section, examples of three different kinds of problems are presented.

4.1. Removal of 'short cut' relationships

Let us consider the first such problem in which some IS-A relationships are not represented by SUBCLASS relationships. Figure 3 shows a small CV with the extents of the classes shown as enclosing boxes. Note that objects $\mathbf{w}$ and $\mathbf{x}$ have the same sets of properties. These four objects will generate three property-introducing classes as shown in Figure 4.

Note that the root $\mathbf{z}$ of the class Z has parents in the classes W and Y, where Y is a SUBCLASS of W. Thus, we would need a SUBCLASS relationship from Z to W and to Y. In such a case, the link between the class Z and the class W is called a 'short cut' relationship. In general, a SUBCLASS relationship from a class A to a class B is called a short cut if there is a path of at least two SUBCLASS relationships from A to B. Referring to the SUBCLASS relationship, using family terminology, a short cut is a SUBCLASS relationship to an ancestor superclass. In the OODB paradigm, it does not make

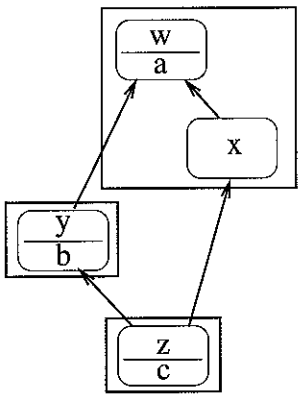


Figure 3. An example CV with a 'short cut' problem.

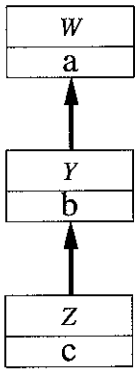


Figure 4. The schema for the CV shown in Figure 3.

any sense to define a SUBCLASS relationship from Z to W or as any short cut, since the SUBCLASS relationships are transitive by definition. Short cut SUBCLASS relationships are not contained in the schema (Figure 4). In other words, there is no SUBCLASS relationship from Z to W. This omission will not have any effect on property inheritance, since the class Z inherits the class W's properties via the class Y. From the view point of property inheritance, the class Z will have the correct property set even without the short cut relationship in the schema.

However, from the point of view of the connections between classes, the removal of such relationships results in a loss of information. The resulting problem can be viewed from two aspects. First, given a schema such as Figure 4, there is no way to tell whether there originally was a short cut SUBCLASS or not. There is no information in the schema indicating whether we have removed a short

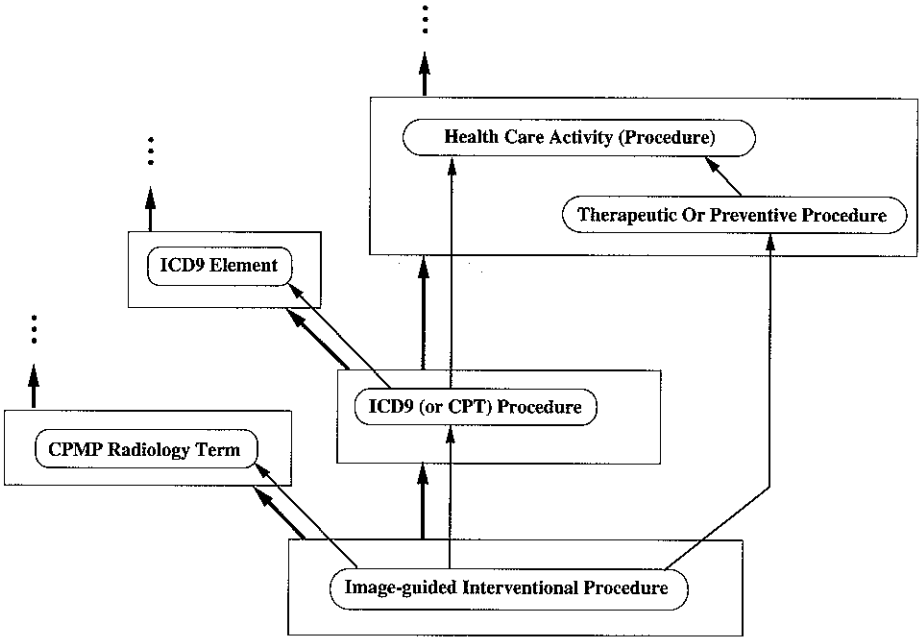


Figure 5. A 'short cut' example from the MED.

cut SUBCLASS or whether it was not there to begin with. Second, a schema such as Figure 4 has no schema-level browsing path parallel to the object-level browsing path (**w**, **x**, **z**). This is due to the lack of a representation in the schema that reflects that in the CV **w** IS-A **z**.

Figure 5 shows such an example from the MED. In this excerpt of the MED, there are five classes: *Health Care Activity Procedure*, *ICD9 Element*, *ICD9 or CPT Procedure*, *CPMC Radiology Term* and *Image-guided Interventional Procedure*. Note that only a portion of their objects are shown in this picture. The root of *Image guided Interventional Procedure*, **Image-guided Interventional Procedure**, is connected by IS-A to three objects: **ICD9 (or CPT) Procedure**, **CPMC Radiology Term** and **Therapeutic Or Preventive Procedure**. Corresponding to the first two of these IS-A relationships, there are SUBCLASS relationships to the classes rooted at the corresponding objects. The SUBCLASS relationship to the third parent is a short cut relationship and is omitted from the schema. When we browse this schema, we have no idea that objects in *Image-guided Interventional Procedure* have parents residing in *Healthcare Activity Procedure*, since the SUBCLASS relationship between these two classes was removed as a short cut. Thus there is no schema-level browsing path corresponding to the object-level browsing path (**Health Care Activity (Procedure)**, **Therapeutic Or Preventive Procedure**, **Image-guided Interventional Procedure**).

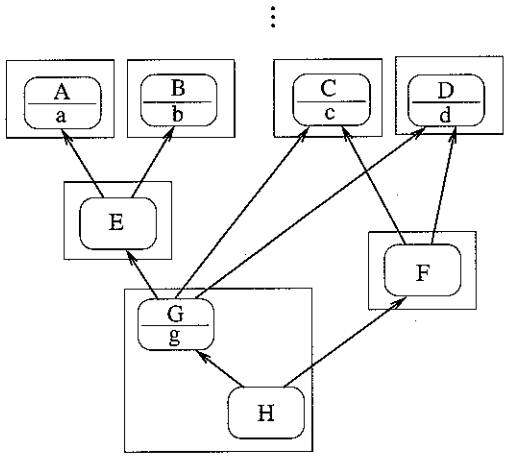


Figure 6. A CV with a missing link problem.

4.2. The problem of 'missing' SUBCLASS relationships

We have found that the instances of two unrelated classes may be connected by IS-A relationships. This kind of abstraction inaccuracy makes the use of the OODB schema less effective for browsing the CV. We call this kind of problem the 'missing SUBCLASS relationship problem' (or missing link problem for short).

An example of one such case is shown in Figure 6, which contains eight objects **A** through **H**. The objects **A**, **B**, **C**, **D** and **G** are property-introducing objects and introduce attributes a, b, c, d and g, respectively. The objects **E** and **F** are intersection objects with property sets $\{a, b\}$ and $\{c, d\}$, respectively. In Figure 6, objects within a box constitute a class and have the same properties. Objects **G** and **H** have the same property sets $\{a, b, c, d, g\}$. Therefore, they should reside in one class, named G since **G** is the root of this class. By our mapping method, the superclasses of the class G are the classes C, D and E. This results in the schema of Figure 7.

Unfortunately, Figure 7 shows no relationship between the class G and the class F. However, the object **H** in the class G is a child of the object **F** in the class F. This IS-A relationship between **H** and **F** will not be represented in the schema, since object **H** is not the root of its class. Thus, there is no schema-level browsing path in the schema of Figure 7 corresponding to the object-level browsing path (**D**, **F**, **H**).

The OODB schema in Figure 7 is not incorrect from the perspective of capturing property inheritance, since objects are instances of the classes with the correct property sets. The only problem is that when we use the OODB schema for browsing, the missing links may cause difficulties. Adding a SUBCLASS relationship between G and F would create two additional problems. First, it would create

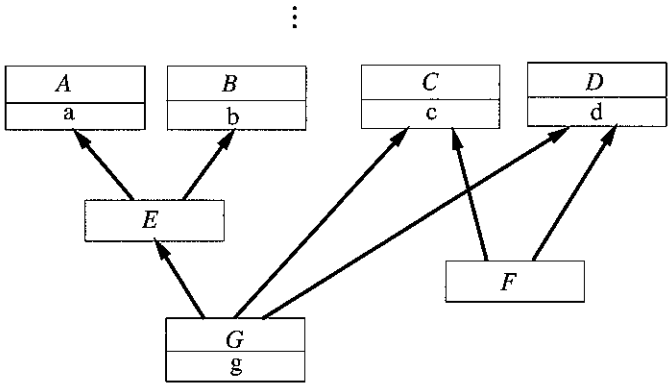


Figure 7. The singly-rooted schema of the CV shown in Figure 6.

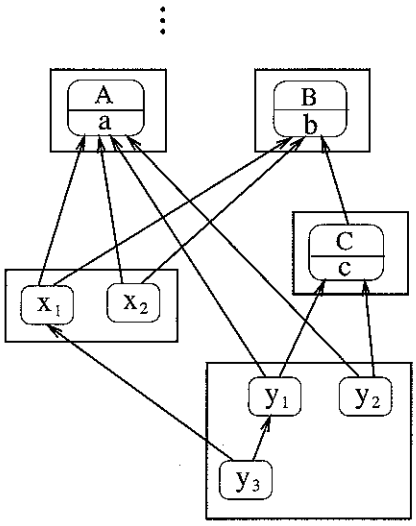


Figure 8. An example CV with eight objects, three property-introducing objects and four intersection objects.

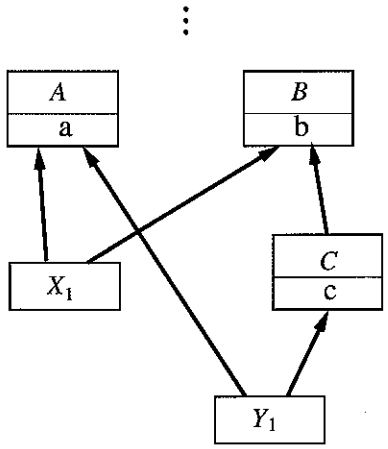


Figure 9. The OOVR schema for the CV shown in Figure 8.

two short cut relationships in the schema (from G to C and to D). Short cut SUBCLASS relationships were mentioned before as harmful. Second, if we define G to be a SUBCLASS of F, the schema may mislead users to think that **G** IS-A **F** does hold.

4.3. Boundary-crossing relationships

In one of our mapping methodologies [20], all multi-rooted intersection classes are further partitioned into singly-rooted classes. This partitioning process is executed within each multi-rooted intersection class. The missing link problem may also happen in a singly-rooted schema.

We will now develop such an example. Figure 8 shows the objects **A**, **B** and **C** which introduce the attributes a, b and c, respectively. The objects $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{y}_1$ and $\mathbf{y}_2$ are intersection objects. As before, objects with the same property sets are displayed in one box representing their class. The OOVR schema of Figure 8 is shown in Figure 9. Via the SUBCLASS relationships, each class will inherit the correct property set for its instances. For example, class Y_1 has the property set $\{a, b, c\}$ and class X_1 has the property set $\{a, b\}$. Note that since $\mathbf{y}_3$ is not a root, X_1 is not a parent class of Y_1. As we mentioned in [20], modeling a multi-rooted intersection class in such a way may pose problems when browsing the OOVR schema. For example, the object-level browsing path $(\mathbf{B}, \mathbf{x}_1, \mathbf{y}_3)$ has no parallel schema-level browsing path, since there is no link from Y_1 to X_1 to represent the link between objects $\mathbf{x}_1$ and $\mathbf{y}_3$.

Figure 10 shows the singly-rooted schema for Figure 8. The intersection class X_1 has been partitioned into two classes: X_1 and X_2. Similarly, the intersection class Y_1 has been partitioned into two classes: Y_1 and Y_2. This schema still reflects the problem of misleading users into thinking that there are no IS-A relationships between the objects of X_1 and Y_1. In spite of the singly-rooted modeling, there is still no schema-level browsing path corresponding to the object-level browsing path $(\mathbf{B}, \mathbf{x}_1, \mathbf{y}_3)$.

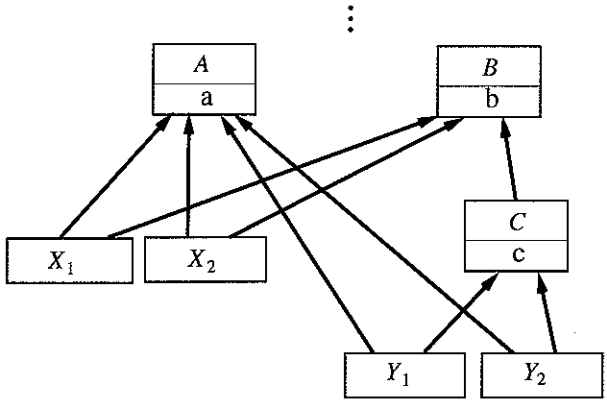


Figure 10. The singly-rooted schema for the CV shown in Figure 8.

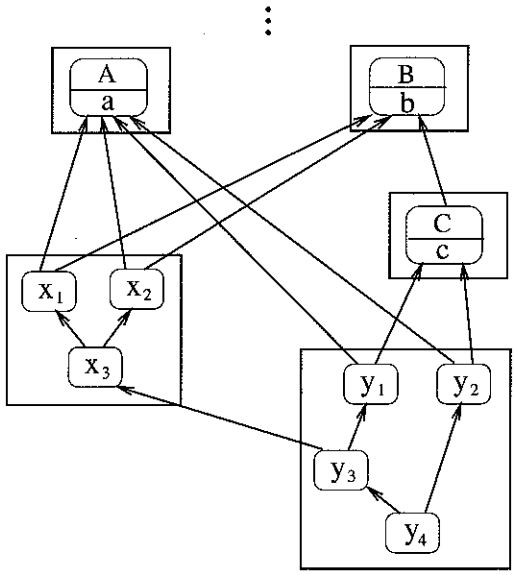


Figure 11. An example CV with ten objects, three property-introducing objects and four intersection objects.

Since $\mathbf{x}_1$ is not in the same intersection class as $\mathbf{y}_3$, the IS-A relationships between them will not be considered when the class diagram is generated. Therefore, the object $\mathbf{y}_3$ will not be considered an articulation object.

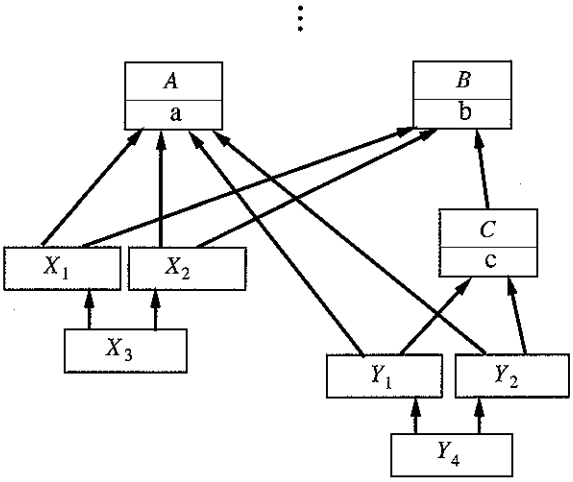


Figure 12. The singly-rooted schema for the CV shown in Figure 11.

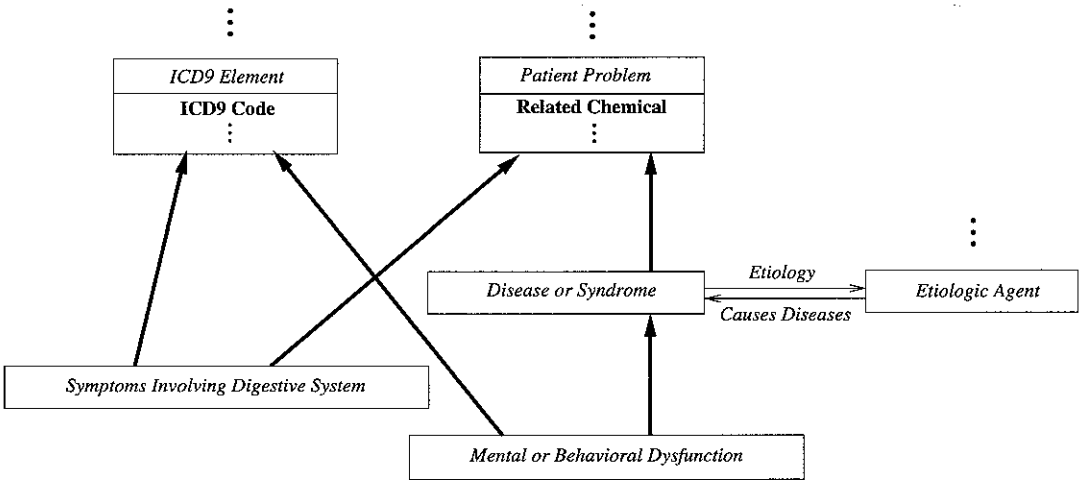


Figure 13. An excerpt of the MED OOVR schema.

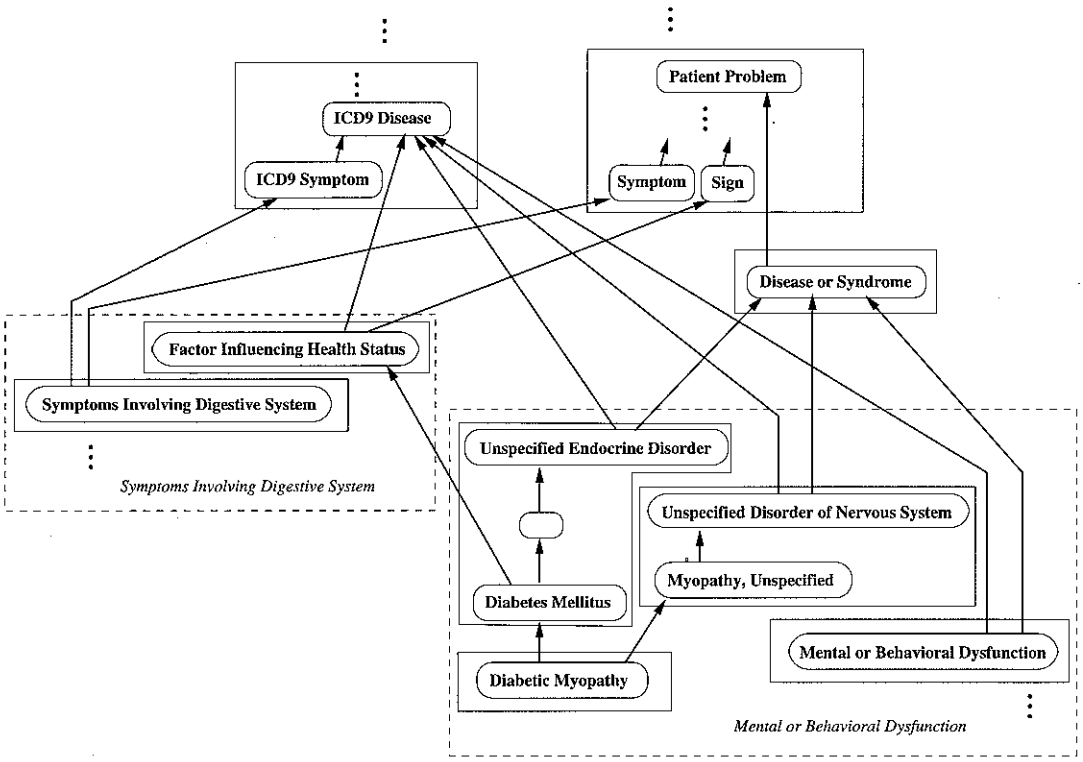


Figure 14. The excerpt CV for the schema shown in Figure 13.

Figure 11 shows a more complicated example. This example is the same as in Figure 8, except that there are two extra objects, $\mathbf{x}_3$ and $\mathbf{y}_4$. The object $\mathbf{x}_3$ has two parents: $\mathbf{x}_1$ and $\mathbf{x}_2$. The object $\mathbf{y}_4$ has two parents: $\mathbf{y}_2$ and $\mathbf{y}_3$. For Figure 11, Figure 9 shows the OOVR schema for Figure 11, which did not change. The singly-rooted schema for this example is shown in Figure 12. We can see that, compared with Figure 10, there are two extra classes, X_3 and Y_4. This is the case because, according to our definition, objects $\mathbf{x}_3$ and $\mathbf{y}_4$ are articulation objects. However, object $\mathbf{y}_3$ is not an articulation object because our methodology is executed within a given class. The partitioning of a multi-rooted intersection class has no effect on representing SUBCLASS links between its resulting classes and external classes. Consequently, this singly-rooted schema will still mislead users into thinking that the objects in x_3 and the objects in y_1 do not have IS-A relationships connecting them. That, of course, is not true in this case. Similarly to the previous case, the object-level browsing path $(\mathbf{B}, \mathbf{x}_2, \mathbf{x}_3, \mathbf{y}_3)$ has no parallel schema-level browsing path. In addition, Figure 12 shows a more serious problem which does not appear in a simple CV like that of Figure 8. In Figure 12, there is no path from Y_4 to X_3. The roots of these two classes are $\mathbf{x}_3$ and $\mathbf{y}_4$ and they do have a path connecting them in the object

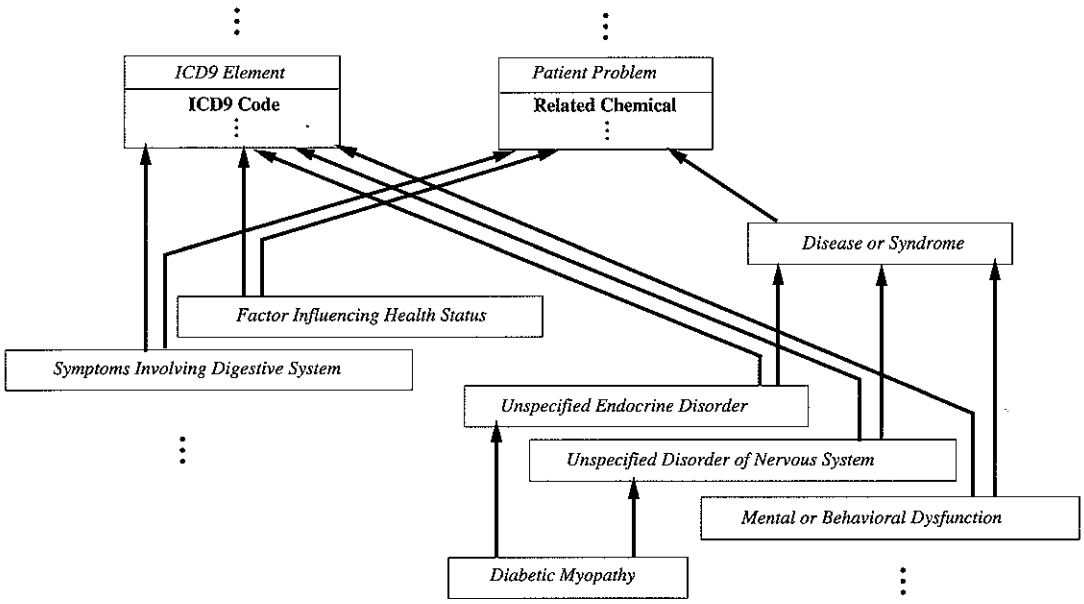


Figure 15. An excerpt of the singly-rooted schema for the CV shown in Figure 14.

network. In other words, $\mathbf{y}_4$ is a descendant of $\mathbf{x}_3$. This missing path makes the singly-rooted schema modeling unsatisfactory. While in Figure 10 we did not have a schema-level browsing path to an object which is not a root of its class, in Figure 12 we do not even have such a path to an object which is a root of its class. This is an even bigger modeling deficiency.

We find many such examples in the singly-rooted MED OOVR schema. This kind of problem usually involves four to five classes. We start our explanation with Figure 13, an excerpt from the MED OOVR schema. The bottom two classes are multi-rooted intersection classes. Note that the class *Symptoms Involving Digestive System* is not a parent of the class *Mental or Behavioral Dysfunction*. That is because the object **Mental or Behavioral Dysfunction** as well as the other roots of this class do not have parents residing in *Symptoms Involving Digestive System*.

Figure 14 shows some objects of singly-rooted classes corresponding to the two multi-rooted intersection classes shown in Figure 13. The class *Mental or Behavioral Dysfunction* has 29 roots in total, only some of which are shown, and the class *Symptoms Involving Digestive System* has 13 roots. Dashed-line boxes enclose the original multi-rooted intersection classes. One of the roots of the multi-rooted class *Mental or Behavioral Dysfunction*, the object **Unspecified Endocrine Disorder**, is an ancestor (but not a parent) of the object **Diabetes Mellitus** in this class. The object on the path connecting them is **Pancreatic Internal Secretion Disorder** that only has one parent **Unspecified Endocrine Disorder**. The object **Diabetes Mellitus** has another ancestor **Factor Influencing Health Status** which is a root of its intersection class *Symptoms Involving Digestive System*.

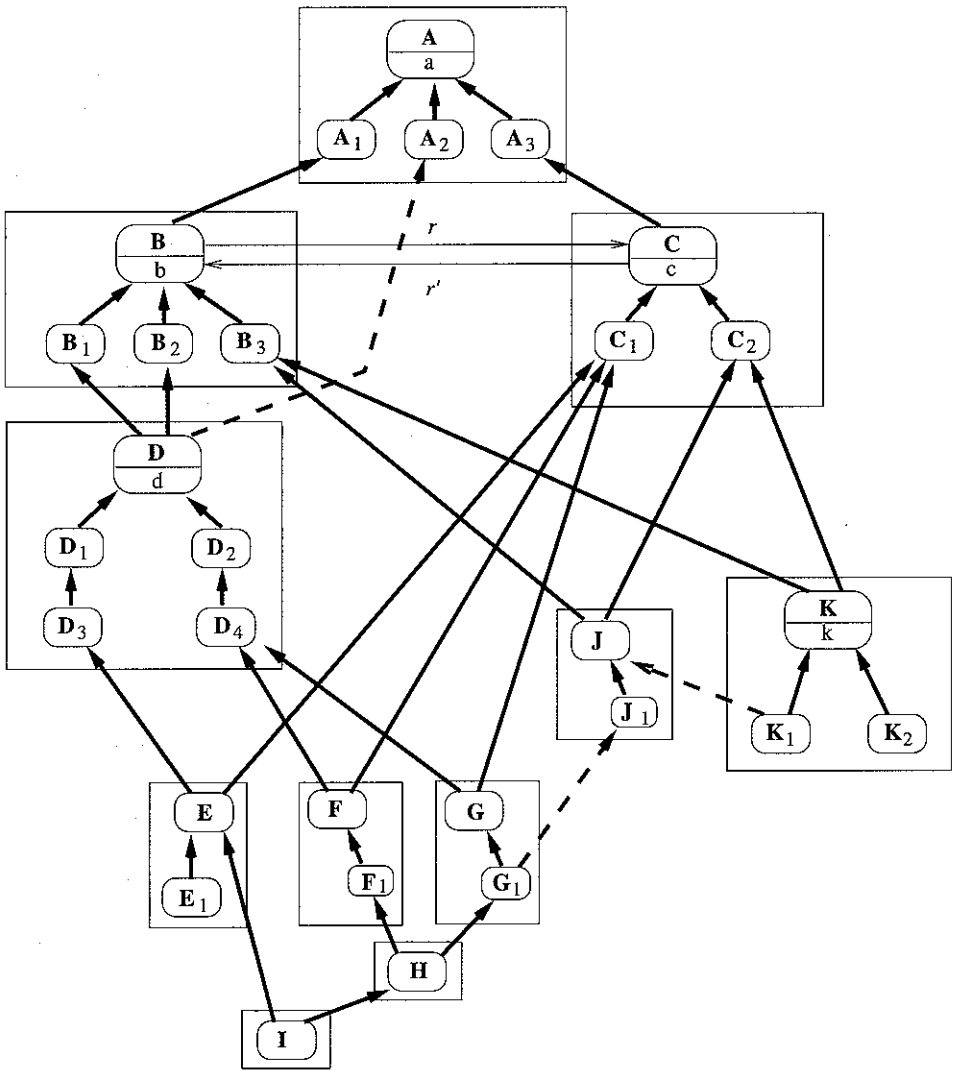


Figure 16. A CV with all three types of problem.

The object **Diabetic Myopathy** has two parents **Diabetes Mellitus** and **Myopathy, Unspecified**. The latter is a child of **Unspecified Disorder of Nervous System**. According to our singly-rooted mapping methodology, it will be considered as an articulation object.

If we apply our singly-rooted mapping methodology to the MED, the two intersection classes in Figure 14 will be partitioned into several classes. Figure 15 is an excerpt of the singly-rooted MED OOVR schema that reflects the objects shown in Figure 14. Note that the object **Diabetes Mellitus** will not be considered as an articulation object, since it has only one path to one intersection object, **Unspecified Endocrine Disorder**, in its multi-rooted intersection class. The object **Factor Influencing Health Status** is an intersection object, but this fact does not have any effect when we further partition the class *Mental or Behavioral Dysfunction* in Figure 13. If **Diabetes Mellitus** is an articulation object, then we will use its IS-A configuration to define the parent of its class. Unfortunately, because **Diabetes Mellitus** is not an articulation object, the class it belongs to will not have any information to indicate the IS-A link between the object **Diabetes Mellitus** and the object **Factor Influencing Health Status**. Note that the property sets of the object **Diabetes Mellitus** and the object **Unspecified Endocrine Disorder** are identical and are a superset of the property set of the object **Factor Influencing Health Status**. Furthermore, in Figure 15 there is no schema level browsing path between *Factor Influencing Health Status* and *Diabetic Myopathy*. That makes the singly-rooted schema unsatisfactory since the object **Factor Influencing Health Status** is an ancestor of the object **Diabetes Mellitus** and **Diabetic Myopathy**. The singly-rooted schema still cannot overcome the boundary-crossing problem as it occurs in a real medical vocabulary.

The problems we have presented in this section are summarized by Figure 16: the IS-A relationship $(\mathbf{D}, \mathbf{A}_2)$ is a short cut; the IS-A relationship $(\mathbf{K}_1, \mathbf{J})$ will generate a missing link in the schema; the IS-A relationship $(\mathbf{G}_1, \mathbf{J}_1)$ is a boundary-crossing relationship.

5. IS-A′ SEMANTIC RELATIONSHIP

The three problems that were presented in the previous section, short cut relationships, missing links and boundary-crossing relationships, arise from the fact that the SUBCLASS relationships connecting classes in a schema cannot fully reflect the IS-A relationships of all instances of a CV. Without a proper link to reflect some important IS-A relationships that are not captured by SUBCLASS relationships, the OOVR schema is not an effective abstraction of a vocabulary. As demonstrated in the previous section, this manifests itself by the existence of object-level browsing paths for which there are no corresponding schema-level browsing paths. To overcome these problems, we define a new kind of relationship, called IS-A′ for the schema. The SUBCLASS relationships in the schema remain the same.

Definition 8. (IS-A′ relationship) Let X and Y be distinct classes such that Y is not a SUBCLASS of X. If there exists an object **c** in Y and an object **d** in X such that **c** IS-A **d**, then we define link IS-A′ from Y to X.

As was mentioned before, the IS-A relationship in a CV has two functions. It is a mechanism for providing property inheritance. It also expresses specialization/generalization knowledge. The IS-A′ relationship in an OOVR schema reflects the second function of the IS-A relationship wherever or not it is expressed by a SUBCLASS relationship.

The IS-A′ relationship is a hierarchical semantic relationship. It is hierarchical in nature, since it is derived from the hierarchical IS-A relationships of the CV, as is the SUBCLASS relationship. However, in contrast, when a class A is a SUBCLASS of a class B then for every instance **a** of A, there exists an instance **b** of the class B such that there is an IS-A relationship or a chain of IS-A relationships in the CV leading from **a** to **b**. Hence, every object in the class B inherits all the properties of every object of class A. Thus we define the SUBCLASS relationship to support inheritance of properties between classes. However, when A IS-A′ B, this only guarantees the existence of some instance or instances of A with an IS-A relationship (or a chain of such relationships) to some instance(s) of B. Thus, the IS-A′ relationships do not provide property inheritance between classes, as opposed to the SUBCLASS relationships.

The IS-A′ relationships do not satisfy transitivity either. As we saw, IS-A′ models exceptional cases in the vocabulary, e.g. an IS-A relationship from a non-root instance of the class A to an object of the class B where no root of A has such an IS-A relationship. Another example is when one of the roots of a multi-rooted intersection class does not have connections to all the parent classes that the other roots do. The problem of missing relationships crossing boundaries that interrupt browsing paths is handled by adding the IS-A′ relationship after the partitioning.

Users can navigate the OOVR schema using SUBCLASS as well as IS-A′ relationships. Hence, the definition of schema-level browsing path can be extended to include the IS-A′ relationships as follows.

Definition 9. (Mixed schema-level browsing path) A mixed schema-level browsing path is a sequence of classes $P_A = (A_1, A_2, \ldots, A_k)$ such that A_{i+1} is a SUBCLASS of A_i or A_{i+1} IS-A′ A_i.

6. THE SINGLY-ROOTED SCHEMA WITH IS-A′ RELATIONSHIPS

We now apply the IS-A′ modeling to all the examples mentioned in Section 4. Figure 17(b) shows the OOVR schema with IS-A′ relationships for the CV shown in Figure 17(a). We use a dashed arrow to represent IS-A′ in a diagram. The IS-A link from **z** to **x** in Figure 17(a) induces the IS-A′ between the class Z and the class W in Figure 17(b). The schema subgraph consisting of all the SUBCLASS relationships has no short cuts. The short cut from the class Z to the class W is captured by the IS-A′ relationship. The semantics represented by this IS-A′ relationship is that there exists an object (**z**) in Z and an object (**x**) in W such that in the CV **z** IS-A **x**. The other two IS-A relationships in the CV of Figure 17(a), namely **z** IS-A **y** and **y** IS-A **w**, are represented by the two SUBCLASS relationships which also support the inheritance of the attributes a and b. Consequently, the schema of Figure 17(b) contains the mixed schema-level browsing path (W, Z) parallel to the object-level browsing path (**w**, **x**, **z**).

The missing link problem described in Section 4.2 can also be solved by adding IS-A′ relationships to a schema. Figure 18(b) shows the singly-rooted schema for the CV of Figure 18(a) with an IS-A′ relationship. Since G IS-A′ F, users are informed that some non-root objects in G have parents in F. With this IS-A′ relationship, users can traverse from G to F or *vice versa*. The schema of Figure 18(b) contains the mixed schema-level browsing path (D, F, G) parallel to the object-level browsing path (**D**, **F**, **H**).

The problem of the partitioning boundary described in Section 4.3 can also be solved by introducing IS-A′ relationships. That is because, by definition, IS-A′ relationships are not restricted to one

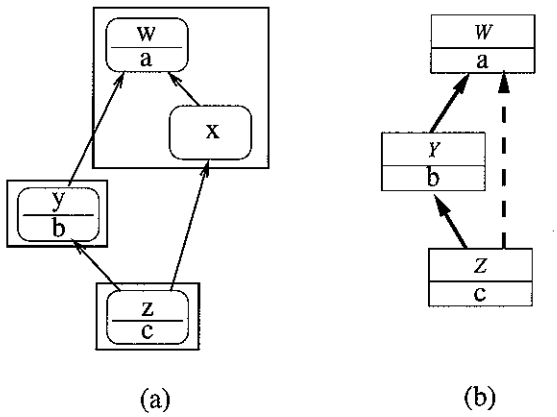


Figure 17. The singly-rooted schema for a CV (a) and its corresponding IS-A′ relationship (b).

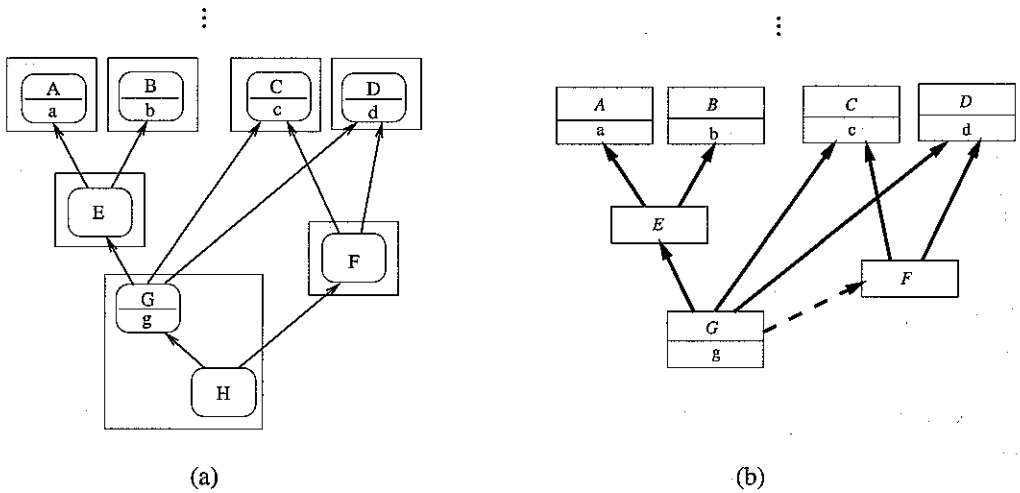
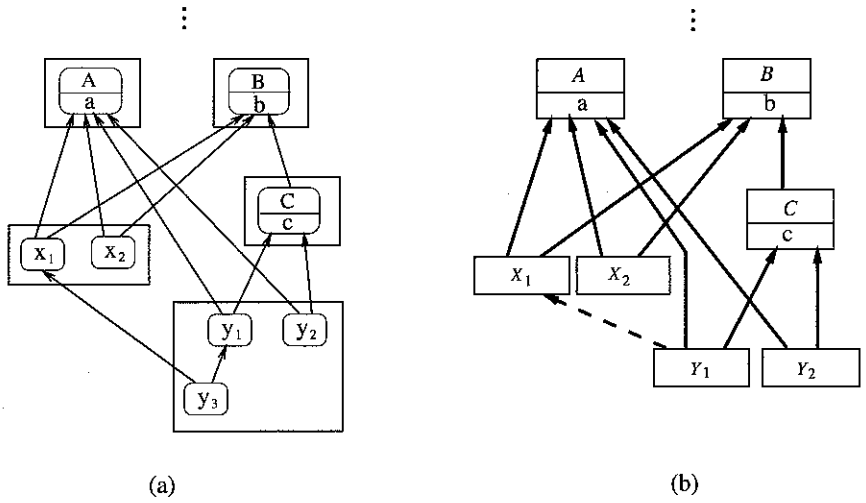
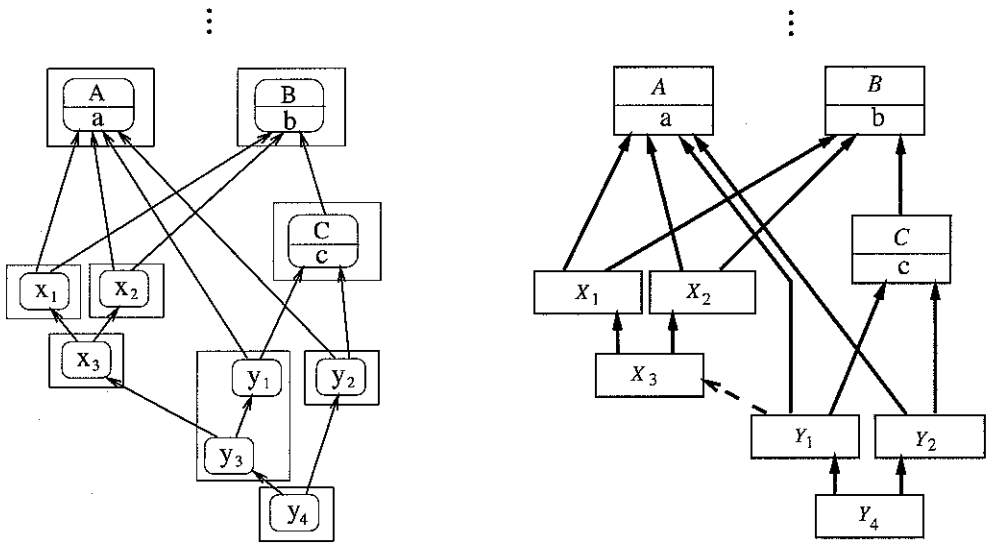


Figure 18. The singly-rooted schema for a CV (a) and its corresponding IS-A′ relationship (b).



(a) (b)

Figure 19. The singly-rooted schema for a CV (a) and its corresponding IS-A′ relationship (b).



(a) (b)

Figure 20. The singly-rooted schema for a CV (a) and its corresponding IS-A′ relationship (b).

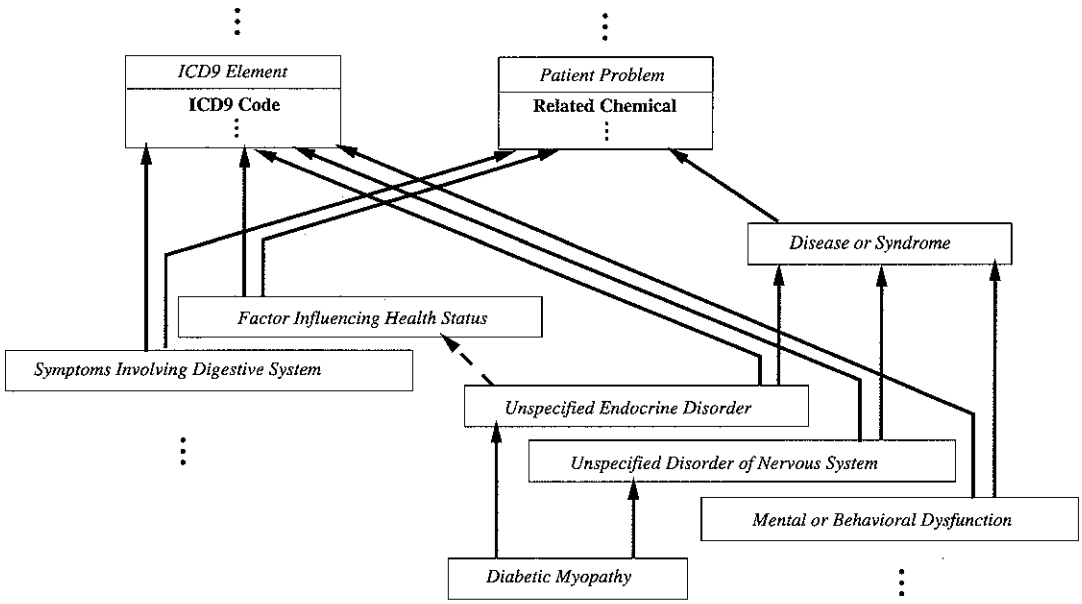


Figure 21. The singly-rooted schema with IS-A′ relationships for the CV in Figure 14.

multi-rooted intersection class. In other words, such a relationship can cross the boundary of a multi-rooted intersection class. Figure 19(b) shows the singly-rooted schema with IS-A′ relationships for the CV shown in Figure 19(a). In Figure 19(b), all the links between classes are SUBCLASS relationships, except for the IS-A′ between X_1 and Y_1. With this relationship, users are made aware that there exists at least one pair of objects in these two classes connected by an IS-A relationship. For instance, the schema of Figure 19(b) contains the mixed schema-level browsing path (B, X_1, Y_1) parallel to the object-level browsing path $(\mathbf{B}, \mathbf{x}_1, \mathbf{y}_3)$ in Figure 19(a).

Figure 20(b) shows the singly-rooted schema with IS-A′ relationships for the CV shown in Figure 20(a). In this figure, there is only one IS-A′ relationship, the one between X_3 and Y_1. From this example, we can see how important an IS-A′ relationship can be. Since the object $\mathbf{y}_1$ is not a descendant of $\mathbf{x}_3$, it may look correct to have no relationship between Y_1 and X_3. However, the object $\mathbf{y}_4$ is not only a descendant of $\mathbf{y}_1$ but also a descendant of $\mathbf{x}_3$. Without a link between Y_1 and X_3, the schema is deficient because it has no path between X_3 and Y_4. The IS-A′ relationship between Y_1 and X_3 makes up for the missing link and creates a mixed path composed of SUBCLASS and IS-A′ relationships between Y_4 and X_3. For instance, the schema of Figure 20(b) contains the mixed schema-level browsing path (B, X_1, X_3, Y_1, Y_4) parallel to the object-level browsing path $(\mathbf{B}, \mathbf{x}_1, \mathbf{x}_3, \mathbf{y}_3, \mathbf{y}_4)$.

Figure 21 shows the singly-rooted schema with IS-A′ relationships for the CV shown in Figure 14. The IS-A link between **Diabetes Mellitus** and **Factor Influencing Health Status** is represented by the IS-A′ relationship between *Unspecified_Endocrine_Disorder* and

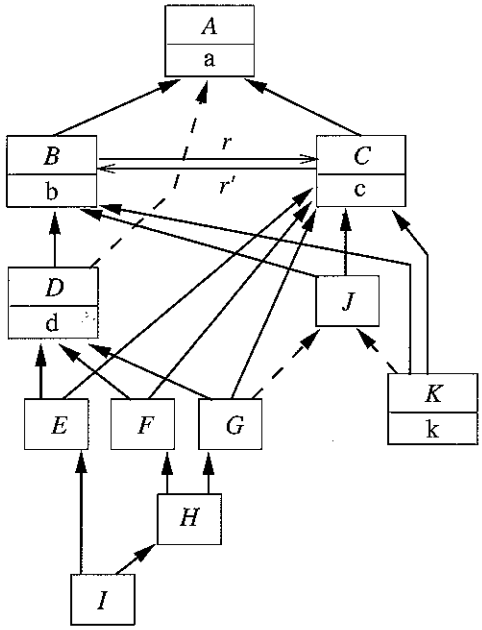


Figure 22. The singly-rooted schema with IS-A′ relationships for the CV in Figure 16.

Factor_Influencing_Health_Status. This IS-A′ relationship facilitates an important indirect connection between *Factor_Influencing_Health_Status* and *Diabetic_Myopathy*. Without it, the singly-rooted schema is incorrect since the path from **Diabetic Myopathy** to **Factor Influencing Health Status** is not reflected. In other words, the object-level browsing path (**Factor Influencing Health Status**, **Diabetes Mellitus**, **Diabetic Myopathy**) has a parallel mixed schema-level browsing path (*Factor_Influencing_Health_Status*, *Unspecified_Endocrine_Disorder*, *Diabetic_Myopathy*) in Figure 21, while it does not have a parallel schema-level browsing path in the schema of Figure 15.

By using the IS-A′ relationships, the singly-rooted schema solves the problems described in Section 4. Moreover, with the addition of an IS-A′ relationship in every place where a link was missing in the schema, there is now a mixed schema level browsing path for every object-level browsing path in the CV. Hence the singly-rooted schema with IS-A′ links completely captures the abstraction of the CV, since there is an abstraction for every possible object-level browsing path.

In Figure 22, we show the singly-rooted schema with IS-A′ relationships for the CV in Figure 16. This schema contains three IS-A′ relationships between three pairs of classes K and J, G and J, and D and A. The object-level browsing path ($\mathbf{A}, \mathbf{A}_3, \mathbf{C}, \mathbf{C}_2, \mathbf{J}, \mathbf{J}_1, \mathbf{G}_1, \mathbf{H}$) has a parallel mixed schema-level browsing path (A, C, J, G, H). Note that the IS-A relationship between the objects $\mathbf{G}_1$ and $\mathbf{J}_1$ is captured by IS-A′ between G and J in the schema. The object-level browsing path ($\mathbf{A}, \mathbf{A}_1, \mathbf{B}, \mathbf{B}_3, \mathbf{J}, \mathbf{K}_1$) has a parallel mixed schema-level browsing path (A, B, J, K). Similarly, the

object-level browsing path ($\mathbf{A}$, $\mathbf{A}_2$, $\mathbf{D}$, $\mathbf{D}_2$, $\mathbf{D}_4$, $\mathbf{F}$, $\mathbf{F}_1$) has a parallel mixed schema-level browsing path (A, D, F).

7. CONCLUSIONS

In this paper, we have solved three problems encountered in modeling a semantic network vocabulary using an OODB. The OODB schema obtained offers a compact abstract view of the vocabulary. Such a schema contributes to the comprehension of the structure and content of the vocabulary. Furthermore, it enables the user to traverse the smaller schema until the right class is found and then traverse the concepts of that class, rather than to traverse the larger vocabulary on the concept level.

We identified three problematic cases where the SUBCLASS relationships failed to model IS-A relationships in the vocabulary. As a consequence, in those cases of 'short-cut relationships', 'missing SUBCLASS relationships' and 'boundary-crossing relationships', there were no schema-level browsing paths parallel to existing object-level browsing paths.

As a solution, IS-A′ relationships were introduced in the OOVR schema to capture IS-A links between objects which were not properly reflected by the SUBCLASS relationships in the schema. More precisely, two classes without a SUBCLASS relationship between them may have instances with IS-A links between them. In such a situation, users may face difficulties when browsing the OOVR schema. By defining the IS-A′ relationship, one can use mixed schema-level browsing paths containing SUBCLASS and IS-A′ relationships to browse the OOVR schema. The problems of short cut relationships, missing SUBCLASS relationships and boundary crossing relationships disappear. Examples have been presented to show how IS-A′ relationships can help in browsing the OOVR schema. Every object-level browsing path now has a corresponding mixed schema-level browsing path.

ACKNOWLEDGEMENTS

This research was (partially) done under a cooperative agreement between the National Institute of Standards and Technology Advanced Technology Program (under the HIIT contract #70NANB5H1011) and the Healthcare Open Systems and Trials, Inc. consortium.

REFERENCES

1. Cimino JJ, Hripcsak G, Johnson SB, Clayton PD. Designing an introspective, multipurpose, controlled medical vocabulary. *Proceedings 13th Annual Symposium on Computer Applications in Medical Care*, Washington, DC, November 1989. IEEE Computer Society Press: Los Alamitos, CA, 1989; 513–517.
2. National Library of Medicine, Bethesda, MD. *Medical Subject Headings*. Updated annually.
3. Columbia Presbyterian Medical Center. *Columbia Presbyterian Medical Center Medical Entities Dictionary*. Columbia Presbyterian Medical Center: New York, 1993.
4. American Medical Association. *Physicians' Current Procedural Terminology: CPT* (4th edn). American Medical Association: Chicago, IL, 1998.
5. College of American Pathologists. *Systematized Nomenclature of Human Medicine*. College of American Pathologists: Northfield, IL, 1976.
6. United States National Center for Health Statistics. *International Classification of Diseases: Ninth Revision, with Clinical Modifications*. United States National Center for Health Statistics: Washington, DC, 1980.

7. Cimino JJ, Clayton P, Hripcsak G, Johnson S. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association* 1994; **1**(1):35–50.
8. Humphreys BL, Lindberg DAB, Schoolman HM, Barnett GO. The Unified Medical Language System: An informatics research collaboration. *Journal of the American Medical Informatics Association* 1998; **5**(1):1–11.
9. Lindberg DAB, Humphreys BL, McCray AT. The Unified Medical Language System. *Methods of Information in Medicine* 1993; **32**:281–291.
10. Rector AL, Bechhofer S, Goble CA, Horrocks I, Nowlan WA, Solomon WD. The GRAIL concept modelling language for medical terminology. *Artifical Intelligence in Medicine* 1997; **9**:139–171.
11. Rector AL, Nowlan WA, Glowinski AJ. Goals for concept representation in the GALEN project. *Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93)*, Washington, DC, 1993. McGraw-Hill: New York, 1993; 414–418.
12. Loomis MES. *Object Databases—The Essentials*. Addison-Wesley: Redwood City, CA, 1995.
13. Zdonik SB, Maier D (eds.). Fundamentals of object-oriented databases. *Readings in Object-Oriented Database Systems*. Morgan Kaufmann: San Mateo, CA, 1990; 1–32.
14. Liu L, Halper M, Geller J, Perl Y. Controlled vocabularies in OODBs: Modeling issues and implementation. *Distributed and Parallel Databases* 1999; **7**(1):37–65.
15. Liu L, Halper M, Gu H, Geller J, Perl Y. Modeling a vocabulary in an object-oriented database. *Proceedings of the 5th International Conference on Information and Knowledge Management (CIKM-96)*, Rockville, MD, November 1996. ACM Press: New York, 1996; 179–188.
16. Oliver DE, Shortliffe EH, InterMed Collaboratory. Collaborative model development for vocabulary and guidelines. Cimino JJ (ed.). *Proceedings of the 1996 AMIA Annual Fall Symposium*, Washington, DC, October 1996. Hanley and Belfus: Philadelphia, PA, 1996; 826.
17. ONTOS. *ONTOS DB/Explorer 4.0 Reference Manual*. ONTOS Inc.: Lowell, MA, 1996.
18. Soloviev V. An overview of three commercial object-oriented database management systems: ONTOS, ObjectStore, and O_2. *SIGMOD Record* 1992; **21**(1):93–104.
19. Gu H, Halper M, Geller J, Perl Y. Benefits of an OODB representation for controlled medical terminologies. *Journal of the American Medical Informatics Association* 1999; **6**:283–303.
20. Liu L, Halper M, Geller J, Perl Y. Using OODB modeling to partition a vocabulary into structurally and semantically uniform concept groups. *IEEE Transactions on Knowledge and Data Engineering* 2002; **14**(4):850–866.

