Methods of Information in Medicine

H. Gu[1], Y. Perl[2], M. Halper[3], J. Geller[2], F. Kuo[2], J. J. Cimino[4]

Partitioning an Object-Oriented Terminology Schema

[1]Dept. of Health Informatics, University of Medicine & Dentistry of NJ, Newark, USA
[2]CIS Dept., New Jersey Institute of Technology, Newark, NJ, USA
[3]Dept. of Mathematics & Computer Science, Kean University, Union, NJ, USA
[4]Dept. of Medical Informatics, Colombia University, New York, NY, USA

Abstract: Controlled medical terminologies are increasingly becoming strategic components of various healthcare enterprises. However, the typical medical terminology can be difficult to exploit due to its extensive size and high density. The schema of a medical terminology offered by an object-oriented representation is a valuable tool in providing an abstract view of the terminology, enhancing comprehensibility and making it more usable. However, schemas themselves can be large and unwieldy. We present a methodology for partitioning a medical terminology schema into manageably sized fragments that promote increased comprehension. Our methodology has a refinement process for the subclass hierarchy of the terminology schema. The methodology is carried out by a medical domain expert in conjunction with a computer. The expert is guided by a set of three modeling rules, which guarantee that the resulting partitioned schema consists of a forest of trees. This makes it easier to understand and consequently use the medical terminology. The application of our methodology to the schema of the Medical Entities Dictionary (MED) is presented.

Keywords: Medical Terminology, Partitioning, Object-oriented Schema, Forest Hierarchy, Comprehension

1. Introduction

Controlled medical terminologies should play a major role in overcoming terminological differences in the design and integration of computerized healthcare information systems. However, the size and complexity of a typical terminology – usually tens of thousands of concepts and a proportionate number of properties – can cause serious problems of comprehension for its users. This can greatly limit the effective utilization of terminologies in overcoming the above – mentioned communication and integration problems.

In previous work, we have developed techniques for representing a controlled medical terminology as an object-oriented database (OODB), a form we call an Object-Oriented Vocabulary Repository (OOVR) [1]. A major advantage of an OOVR is its schema, which provides an important abstract view of the terminology. Using this view, one is able to obtain an understanding of the terminology's overarching structure. The schema can also be used as a means for effectively browsing and traversing the terminology [2] and gaining an orientation to its contents. Moreover, the schema can be an important tool in improving the terminology. It can be used to uncover and correct errors in the system [2, 3].

We have applied our techniques to a number of terminologies, including the Medical Entities Dictionary (MED) [4] of New York Presbyterian Medical Center. The MED schema was derived by partitioning the concepts into classes such that all the concepts of a class share the same properties, i. e., attributes and relationships. Furthermore, in [2] we have developed an object-oriented schema which extends the Semantic Network of the Unified Medical Language System (UMLS) of the National Library of Medicine [5]. The object-oriented UMLS schema was derived by partitioning all concepts of the UMLS into classes such that all the concepts in a class belong to the same set of semantic types of the UMLS Semantic Network. The object-oriented schemas for such terminologies are compact compared to the sizes of the original terminologies. An object-oriented schema of 124 classes captures the MED's approximately 56,000 concepts (1998 version of the MED) – approximately a 450-to-1 reduction [3]. In a recent paper [6], we provided a more refined object-oriented schema for the MED. A schema of 1,296 classes represents the over 500,000 concepts of the UMLS – a 385-to-1 reduction [2].

Even though the object-oriented schema is an important abstraction that promotes enhanced understanding of terminologies, it can still be too large and difficult to comprehend. In this paper, we focus on the issue of reducing the complexity of a schema. We present both a theoretical paradigm and a methodology to aid in the comprehension of existing large terminology schemas. Our approach employs a combination

of the notions of *informational thinning* (i.e., displaying only high priority elements of the schema) and *partitioning.*

Our methodology is based on the partitioning of a large terminology schema into parts. Based on the rules of *disciplined modeling,* a new refined modeling technique [7], we develop a theoretical paradigm that guarantees the identification of a meaningful forest hierarchy within the terminology schema's specialization hierarchy. The methodology for finding such a forest relies on an interaction between an expert and a computer. An expert refines the specialization hierarchy of a terminology schema based on his understanding of the application domain, according to the rules of disciplined modeling. The computer provides support by performing heavily computational procedures. We will apply our methodology to the MED's schema [1]. The resulting forest represents a partition of the specialization hierarchy into disjoint, meaningful, manageably sized trees. Such a hierarchy functions as a skeleton of the schema and supports comprehension efforts.

Some users, such as terminology designers and maintainers need a higher level of comprehension of the terminology. For this purpose they need first to comprehend a schema of the terminology to acquire an abstract overarching view of the terminology. Although such users are in the minority, it is essential to provide them with the necessary tools and support to perform their task, since all users are dependent on the design and maintenance work of the terminology. Thus, to comprehend the schema, such a user can begin with the study of small logical fragments and progress from there to study the relationships between pairs of such fragments. Importantly, each of the fragments is a tree structure and can typically fit on a single computer screen.

A preliminary presentation of the theoretical paradigm only appeared in [7]. In [8], we presented a technique for directly partitioning a terminology's knowledge content (modeled as a semantic network). It was initially not clear if these methods can also be applied to a schema. In this paper, we establish how to adapt our paradigm to a schema and show that the desired result is indeed sound.

2. Informational Thinning and Partitioning

2.1 Schema Complexity

We measure the size of a schema by the number of its classes. Our experience is that comprehension difficulties for a large and complex schema stem more from the density of the relationships than from the size. We define the *complexity* c of a schema to be the ratio of the number of the relationships (between classes) to the number of classes. For two schemas of equal size, we conjecture, based on our experience, that a more complex schema is more difficult to comprehend.

In [1, 3], we developed an object-oriented schema, containing 124 classes, that captures the knowledge content of the MED [4]. We will be applying our methodology to this schema. In addition to its classes, the MED schema contains 262 relationships, and thus has a complexity c = 262/124 =2.11. A substantial effort is required to understand the contents of this schema, which is too large to be displayed on one computer screen.

2.2 Informational Thinning

The specialization hierarchy of an object-oriented schema serves as the platform for property inheritance. It is the backbone of an object-oriented schema. Informational thinning lets us concentrate on the specialization hierarchy of a schema by removing all other properties except for the *subclass* relationships. We call it the *hierarchical (sub)schema of a schema.* The hierarchical subschema has the same size as the original schema but a lower complexity, and it is easier to understand. It is furthermore easier to comprehend due to the uniform nature of the hierarchical relationships, in contrast to the varied semantics of the rest of the relationships which are user-defined. Due to this difference, the designer does not have to label the subclass relationships, which are identified by a special graphical icon, while for the user-defined relationships, labels are necessary to denote their semantics.

Since a class in an object-oriented schema can be specialized into a number of subclasses and can also be generalized into a number of superclasses, the hierarchical subschema of an object-oriented schema will typically be a directed acyclic graph (DAG). Thus, for a large schema, even the hierarchical subschema may be difficult to comprehend due to the existence of multiple superclasses for many classes. A hierarchical schema has a forest structure if no class has more than one superclass. A connected hierarchical schema is a spanning tree of the DAG. It is generally easier to comprehend a forest than a DAG of the same size, due to the fact that upward paths are not branching in a forest.

2.3 Schema Partitioning

A second approach that simplifies the comprehension of a complex large schema is partitioning it into smaller subschemas. This applies to DAG and tree schemas alike. From the technical side, only a limited size subschema can be displayed on a computer screen. From the conceptual side, human comprehension capacity is limited and functions better when concentrating on a small subschema at a time. Hence, we will partition a large schema into smaller subschemas. In the partitioning process, we typically have two goals: First, to identify small subschemas which form logical units. Second, to generate a small number of subschemas which fit on a computer screen, and which together make up the complete schema.

The need to achieve a logical partitioning of an existing schema introduces a vicious cycle, as one needs to comprehend the schema in order to partition it logically. A possible line of action is to combine informational thinning and partitioning, by trying to partition the hierarchical subschema and use this partition to impose a partition on the original schema. Obviously, this partitioning problem is much simpler than the original partitioning problem since the subschema has a lower complexity. However, unless the subschema is a forest, the partitioning problem in general is still NP-complete [9]. On the other

hand, if the hierarchical subschema has a forest structure, then there exist efficient algorithms for optimal partitioning according to various criteria [10-13]. In this paper, we will describe an approach to show that in a hierarchical subschema of a general schema, there exists such a forest hierarchical schema, which helps to support comprehension.

3. Rules of Disciplined Modeling

In [7], we presented the framework of disciplined modeling for partitioning an object-oriented schema and a proof that if its three rules were followed, the schema would contain a forest of contexts.

However, we cannot use that framework in a wholesale manner for our purposes here. The object slicing [14] model of the object-oriented schema used in [7] assumes that in a hierarchy of the schema we may refer to the same "'real-world object'' in several levels. That is, the information regarding a given real-world object is distributed among several instances of classes in the hierarchy. To extract the full information pertaining to a single real-world object, one has to collect it from the various instances.

However, this is not the model of object-oriented representations we used in the definition of the OOVRs. In the OOVRs, the concepts of the terminology are partitioned between the various classes, such that the sets of concepts of various classes are disjoint. The knowledge pertaining to a concept is stored in a single instance of one class only. Thus, we use a different model [15] of OODB which does not permit the distribution of information of a single real-world object among various classes. The definitions of the attributes defined in the higher level classes in the hierarchy are inherited at the lowest level. The values for the attributes are assigned only at the instance at the lowest level. Due to this difference, we need to define one of the rules in a different way from that in [7]. Furthermore, the proof in this paper does not use the notion of "real-world object''.

3.1 Category-of and Role-of Specialization Relationships

To identify a meaningful forest subschema of a schema, we look into the nature of the specialization relationship. Previous research has identified two kinds of specialization relationships, namely, *category-of* and *role-of.* According to our definition [16], *category-of* is a specialization relationship used where both the superclass and the subclass are in the same context. *Role-of* is the specialization relationship used where the superclass and the subclass are in different contexts.

How does a designer of an object-oriented schema determine whether a given specialization relationship is *category-of* or *role-of*? That depends on whether the two classes connected by the relationship are in the same context or not. However, this determination is not always easy. In spite of extensive research, e. g., [17-21], there is still no definition of "context'' which is widely accepted. One line of research on context comes out of the CYC project [22]. There, contexts were introduced for structuring purposes. Work following this line [17, 20] assumes that a context is a first-class object used in axiom schemata. As a workshop on the use of "context'' in natural language processing showed [19], researchers agree to disagree on what contexts are.

We, however, are not trying to define the notion of context. Rather we are making the *a priori* assumption that contexts exist, and we are trying to find them. We accept that for some designers two classes are in the same context while for others they are in different contexts, due to different views of the application and levels of refinement. From our standpoint, the designer of a schema should determine the context for each class.

We believe that organizing a complex schema into *reasonable* contexts is preferable to leaving the schema without such an organization. We provide in this paper a theoretical paradigm for the existence of such assignments of classes to contexts that results in a forest subschema of the DAG hierarchical schema. Also, we introduce a methodology for finding such a forest subschema to support comprehension of the schema.

In order to ensure that a forest hierarchical subschema can be identified, the assignment of classes to contexts must follow *disciplined modeling* which satisfies three rules which will be reviewed below. As we shall see, every situation, which can be modeled when the rules are not adhered to, can be captured with the rules and a few modifications in the modeling to satisfy the rules.

3.2 Rules for Contexts

First, we define a mathematical relation *equicontext* between classes. A pair of classes belongs to the equicontext relation if both classes belong to the same context.

Rule 1: The equicontext relation between classes is an equivalence relation.

An equivalence relation satisfies reflexivity, symmetry, and transitivity and partitions the elements of a set into disjoint subsets, such that only every two elements of the same subset are related. Hence, **Rule 1** implies **Rule 1'.**

Rule 1': The equicontext relation partitions the classes of a schema into disjoint contexts.

Rule 1' forces the designer into the explicit specification of the contexts in the schema and the resolution of ambiguous situations in a systematic way. There is no unique way of assigning classes to contexts. As we are dealing with data modeling, there are usually different ways to model the real-world environment. We further do not claim that contexts in an application are naturally disjoint. To the contrary, in many complex applications, contexts overlap. However, to achieve our goals, disciplined modeling requires the modeler to enforce disjoint contexts. As will be seen, the partitioning of classes into disjoint contexts is a difficult task involving subtle analysis.

Rule 2: Two *category-of* specialization classes of the same superclass cannot be *category-of* descendants of one another and cannot have a common *category-of* descendant class.

To guarantee **Rule** 2 in disciplined modeling, consider the case of a class *A* which is a subclass of two classes *B* and *C*; by **Rule** 2, it cannot be that both these subclass relationships are *category-of.* Thus, we need to give the

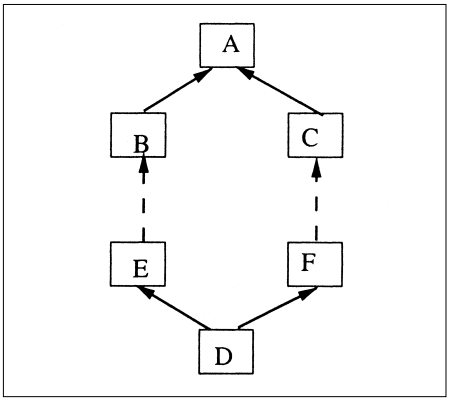

Fig. 1 A schema demonstrating our proof.

modeler guidelines about how to deal with the modeling of such a situation. Such guidelines will be discussed in Section 4.

Rule 3: For each context, there exists one class *R* which is the *major* (or defining) class for the context such that every class in the context is a descendant of *R*.

In other words, each context has one class which is its "root," i.e., there is a directed path of *category-of* relationships from each class of the context to this class.

Rule 3 does not limit the modeling of the application. If a context has several root classes, a new unique root class *R* can be created with all the root classes as its subclasses.

Theorem: Using disciplined modeling, a class has at most one *category-of* superclass.

Proof: Assume to the contrary that there exists a class *D* which has two *category-of* superclasses *E* and *F* (see Fig. 1). According to the definition of *category-of, D* and *E* are in the same context. Similarly, *E* and *F* are in the same context. By the transitivity of the equicontext relation **(Rule 1)**, *E* and *F* are in the same context.

By **Rule 3**, there is a major class *A* for this context such that the classes *E* and *F* are *category-of* descendants of *A*. Hence, there is a sequence of *category-of* relationships from *E(F)* up to *A*. If the paths of *category-of* relationships from *E* to *A* and from *F* to *A* are not disjoint (i.e. the class *A* is not the first class which appears in both paths), then denote now by *A* the first such joint class on these two paths. Let *B(C)* be a subclass of the class A on a path of *category-of* relationships from *E(F)* to *A*. Hence, class *D* is a *category-of* descendant of class *B(C)*. Thus, both the *category-of* subclasses *B* and *C* of the class *A* have a common *category-of* descendant class *D*. A contradiction to **Rule** 2.

The theorem implies that the *category-of* hierarchy has a forest structure of one or more trees which serve as the backbones of the schema.

4. Methodology for Finding a Forest Hierarchy

We have described a conceptual partitioning framework which guarantees that for the price of following the rules of disciplined modeling, there can be found a *forest structure subschema* of a schema. This forest structure subschema serves as a skeleton supporting the comprehension of the terminology schema. Furthermore, the trees of the forest represent contexts which are each a logical subschema approximating all knowledge relevant to a specific subject area, further supporting the comprehension of the original schema.

In this section, we will describe a methodology that identifies a forest structure subschema of a given schema. The methodology involves human-computer cooperation. The human domain expert makes some judgment decisions based on an understanding of the medical knowledge, while the computer provides results of algorithmic procedures for tasks which do not involve complex intuitive decisions but might require many computational steps.

We will specify which steps are performed by a computer and by the human domain expert. The result of our methodology is a refinement of the specialization hierarchy of the terminology schema. Every subclass relationship becomes either a *category-of* or a *role-of*. We will differentiate between three kinds of *role-of* relationships. They are *regular role-of, role-of/intersection*, and *role-of/category-of*.

However, for partitioning purposes, they will all be treated in the same way. The *category-of* relationships will form a forest. We will also show an example of applying our methodology to the large and complex MED object-oriented schema. The MED's 56,000 concepts are captured by an object-oriented schema consisting of 124 classes and 190 subclass relationships [1, 3]. We selected a subschema which contains 34 classes and 52 subclass relationships to demonstrate our methodology.

Step 1 Informational thinning *(Computer)*: All attributes and relationships other than subclass relationships are removed from the object-oriented schema.

Step 2 Topological sort *(Computer)*: Arrange the subschema in topological sort order [23].

Step 3 Identify roots of contexts *(Human)*: The subschema is scanned top-down. Defining classes (roots) of contexts are identified. The decision should be made by the meaning and importance of the class in the terminology compared to its superclasses' meanings. These chosen classes start new contexts rather than refining the contexts of their superclasses.

The subclass relationships from the root classes to their superclasses are changed to *role-of* relationships. This relationship is a *regular role-of*, where the relationship models a switch of context.

For example, the MED subschema is scanned top-down following the topological sort ordering 1 to 34 (in Fig. 2). The class *Diagnostic Procedure* (14) which has one superclass *Health Care Activity (Procedure)* (5) starts a new context since *Health Care Activity (Procedure)* describes all the activities of a healthcare plan and *Diagnostic Procedure* specifically focuses on the procedures for diagnostic purposes. Thus, *Diagnostic Procedure* is a *role-of Health Care Activity (Procedure)*.

The class *Blood Gas Panel* (22) has two superclasses, *ICD9 Diagnostic Procedure* (19) and *Laboratory Diagnostic Batteries* (23). *Blood Gas Panel* refers to tests of concentrations of gases. However, *ICD9 Diagnostic Procedure* and *Laboratory Diagnostic Batteries* describe general diagnostic procedures. Thus, the class *Blood Gas Panel* defines a new context and is a *role-of* both *ICD9 Diagnostic Procedure* and *Laboratory Diagnostic Batteries*.

In this step, 12 classes are chosen to define new contexts. Figure 2 shows all

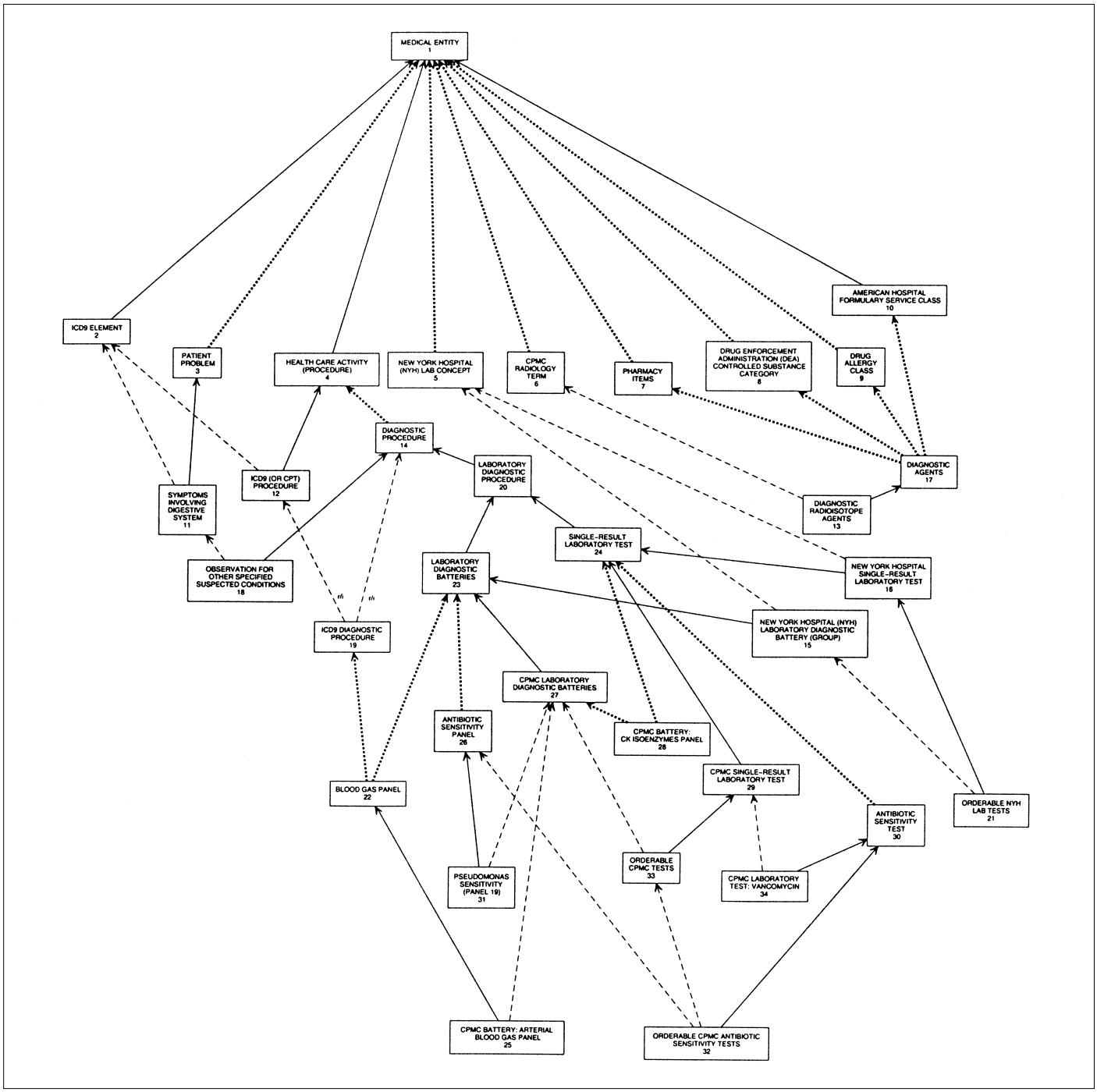

Fig. 2 The MED subschema after Step 5.

these classes, except the root class, as *role-of* (dotted lines) their superclasses.

Step 4 Multiple superclasses *(Computer)*: All classes with multiple non-*role-of* relationships to superclasses are listed in bottom-up order. (We will explain later why we are using bottom-up processing at this point.)

In Fig. 2, there are 13 classes with multiple non-*role-of* relationships to their superclasses. They are (34), (33), (32), (31), (25), (21), (19), (18), (16), (15), (13), (12), and (11).

Step 5 Identify major superclass *(Human)*: For each class identified in **Step 4**, the expert identifies at most one superclass in the same context as the class in order to conform to **Rule** 2. The relationship to this superclass is defined as a *category-of* relationship while other relationships to parents of the class are defined as *role-of*.

In our experience, for most of the classes with multiple superclasses, an expert can easily determine which of the superclasses should have a *category-of* relationship directed to it. There is a minority of cases where the decision about a major superclass is not easy. In such cases, we try to distinguish which of the several superclasses, if any, should have a *category-of* relationship pointing to it, based on the partial con-

text information we have already accumulated in our bottom-up processing. We provide the following guidelines.

Case 1: One of the superclasses is definitional, describing the essence or the definition of the subclass, while the other superclasses describe the functionality or usage of the subclass. Then we look at the partial context to which the class and its descendants belong. (This is the reason for the bottom-up processing.) We try to determine whether the nature of the *category-of* relationships in this partial context is functional or definitional. If it is definitional, the definitional superclass is chosen as the major superclass. If it is functional, then we will prefer the functional superclass. If there are several functional superclasses, we will prefer the one which matches the function appearing in the partial context of the class. If the class is currently the only class in its context, we will choose the definitional superclass. The class is made *category-of* this major superclass and *role-of* the other superclasses. This kind of a *regular role-of* relationship is a switch of context from the class to the superclass.

An example of **Case 1** is the class *CPMC Battery: Arterial Blood Gas Panel* (25) with two superclasses, *Blood Gas Panel* (22) and *CPMC Laboratory Diagnostic Batteries* (27). Since (22) defines various kinds of blood gas tests, it is the definitional superclass of (25) while (27) defines CPMC laboratory procedures for diagnostic purposes of batteries of tests and is a functional superclass of (25). Thus, (25) is a *category-of* (22) and a *role-of* (27). (We use dashed lines in Fig. 2 to represent *role-of* relationships identified in **Step** 5.)

Case 2: Both superclasses are definitional, however it is possible to distinguish the major by linguistic analysis of the name of the subclass. When the concept of one superclass is expressed in the subclass name as a noun while the concept of another superclass is expressed in the subclass name as an adjective, then the noun defines the major superclass. If both concepts are expressed grammatically as nouns, then the second noun is considered the major concept. There are well known exceptions to this rule. A toy gun is a toy and not a gun.

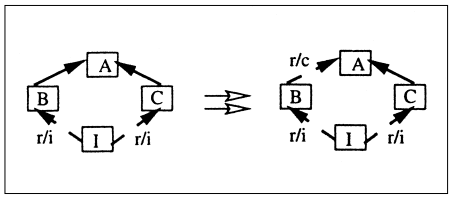

Fig. 3 A diamond structure.

Case 3: All superclasses are definitional, with the same importance or indistinguishable importance, as each of them contributes to the definition of the class in an equal or indistinguishable way. In this case, the semantics of the class is a combination of the semantics of all its superclasses. In such a situation, the class with multiple superclasses could belong to the context of any of its superclasses.

However, by **Rule 1**, it cannot belong to more than one context. Also, we have no reason to prefer one over the others. Each choice of context will disassociate the class from the other contexts. This conflict is resolved by requiring that such a class start a new context which represents the class as an intersection of its superclasses. Thus, this class is *role-of* all its superclasses. We call this type of *role-of* "`role-of/ intersection'' represented as r/i in the figures.

One example of **Case 3** is the class *ICD9 Diagnostic Procedure* (19) which has two definitional superclasses, *Diagnostic Procedure* (14) and *ICD9 (or CPT) Procedure* (12). Since both superclasses contribute with equal importance to the class (19), we cannot prefer one over the other. (For different viewpoints, each one is playing a major role.) Hence, the class (19) is a *role-of* both its superclasses. Fig. 2 shows the subschema after identifying the major superclass for each class listed in **Step 4**.

As we mentioned before, we realize that sometimes different experts will make different choices for the major superclass due to their perspectives. For example, if the choice is made by a radiology expert, he may choose the class *CPMC Radiology Term* (6) as a major superclass for the class *Diagnostic Radioisotope Agents* (13) rather than the class *Diagnostic Agents* (17) as marked in Fig. 2. As a result, we will get a *CPMC Radiology Term* context of two classes. On the other hand, *Diagnostic Agents* (17) will be an isolated class. For a radiology expert, such a partition is more meaningful. Our techniques enable each kind of expert to obtain a partitioning fitting their interest.

Step 6 Identify diamond structures *(Computer)*: For each class *I* in the resulting list of **Step 4** and each pair of superclasses S_1 and S_2 of *I*, find a lowest common ancestor *A* of both S_1 and S_2. For each pair of such classes *I* and *A*, output the structure (represented by <*I,A*>) containing *I, A*, and all the classes which are both descendants of *A* and ancestors of *I*. This is called a diamond or extended diamond structure.

Step 7 Resolve contradictions in the diamond structures *(Computer)*: In order to fulfill **Rule** 2 of disciplined modeling, each diamond or extended diamond structure must contain classes from more than one context. After executing the above steps, all the diamond structures already satisfy **Rule** 2. However, there is one case where we must artificially change additional *category-of* relationships to *role-of* relationships, in order to resolve a contradiction. In such a case, which we call a *contradictory diamond case*, the class *I* of the diamond structure <*I, A*> is a *role-of/ intersection* of its superclasses. All other classes in the diamond structure belong to one context (see Fig. 3). Since the class *I* is the intersection of two superclasses *B* and *C*, they cannot both belong to the same context of their superclass *A*. Otherwise, since the intersection of a context with itself will result in the original context, the intersection class must belong to this common context. Thus, the classes *B* and *C* should belong to different contexts. The *category-of* relationship from *B* to *A* is changed to *role-of*.

However, we want to maintain the distinction between this *role-of* and the two other kinds. Therefore, we denote this kind of *role-of* as "*role-of/category-of.*" It is represented by r/c in the figures.

For example, in Fig. 2 the diamond structure <*ICD9 Diagnostic Procedure* (19), *Health Care Activity* (5)> is the only one which contains *role-of/intersection.* However, it is not a contradictory diamond structure.

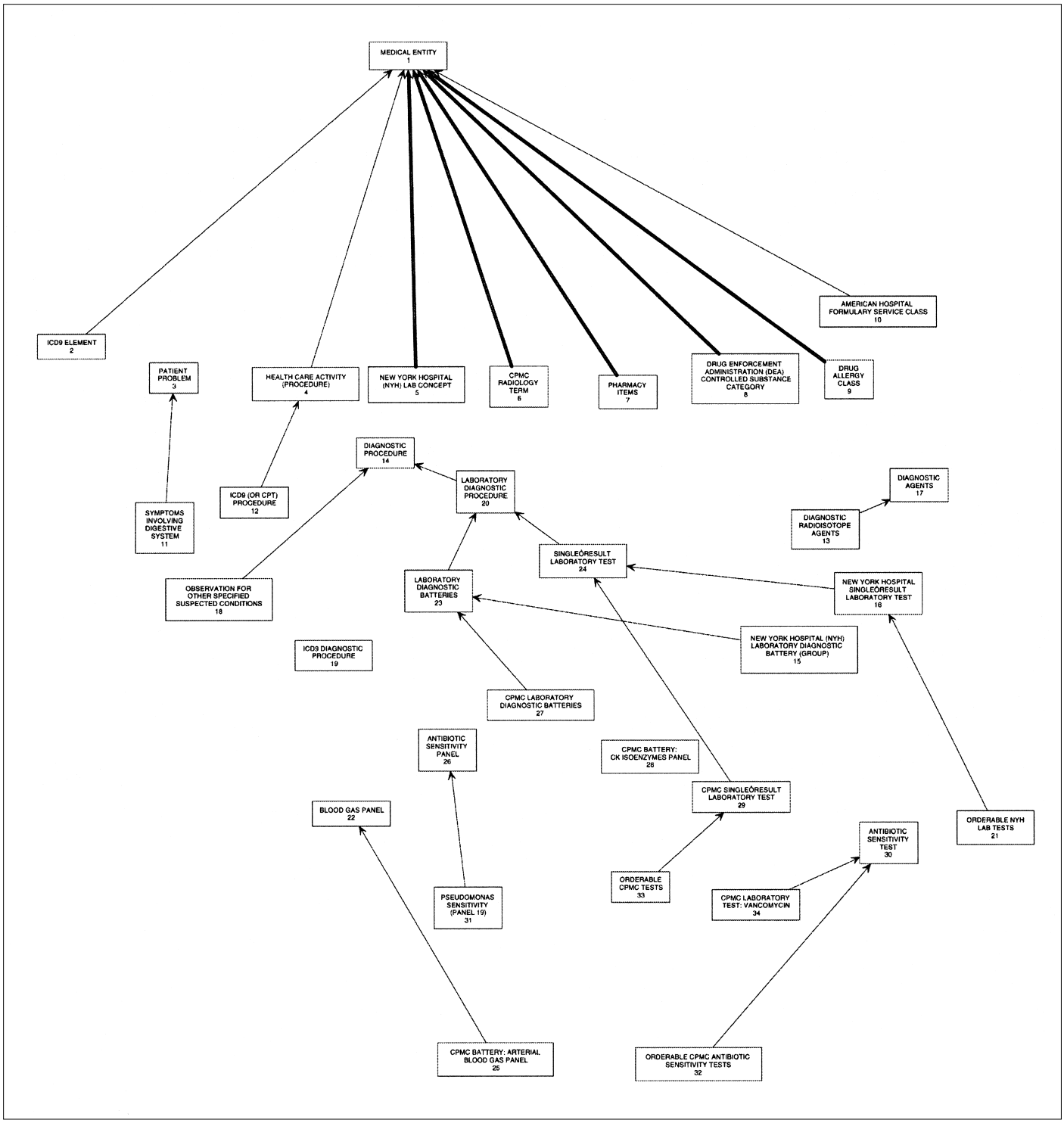

Fig. 4 A forest structure of the subschema after applying the methodology.

Step 8 Get a forest hierarchy *(Computer)*: After all subclass relationships are refined as either *category-of* or *role-of* relationships, a forest hierarchy of the *category-of* relationship is obtained by deleting all three kinds of *role-of* relationships.

The forest hierarchy in Fig. 4 is obtained by removing all *role-of* relationships from Fig. 2. It shows the resulting partition of the MED subschema into 14 trees. Seven of the trees are single nodes not matching other nodes. The other seven trees consist of logical units such as "Diagnostic Procedure" (11 nodes), "Antibiotic Sensitivity Test" (3 nodes), and "Blood Gas Panel" (2 nodes). The partition of the subschema helps in its comprehension.

Step 9 Reconnect single nodes *(Computer)*: Reconnect each single node,

which has either originally only one parent, or only one of its original parents is itself a single node, to this parent. Such a single node context was created in **Step 3** where it was judged both important and different enough from its parents to warrant starting a separate context. However, as such a context was not further developed, it is proven unjustified and is reconnected to its parent.

In our example, 5 single nodes are reconnected to their parent *Medical Entity* resulting in a ten node context (see dual-lines in Fig. 4). The resulting partition has 9 trees, 2 of which are single nodes. An average tree contains 4 nodes.

The methodology used both top-down processing and bottom-up processing. The determination of the context of classes is performed top-down, since the context of the root class defines the context of its descendants. When scanning the schema top-down, an expert can identify which class defines a new context rather than continuing a context of one of its superclasses. On the other hand, when determining bottom-up to which context a class belongs, choosing from among its superclasses, it is important to know the descendants of the class which belong to the same context. This knowledge will help to determine which of the superclasses fits best to the already constructed partial context.

We need to emphasize that the purpose of the partitioning is not just to help comprehension of the forest resulting by deleting many subclass relationships. The purpose is to support comprehension of the whole original schema including the non-hierarchical relationships eliminated in the informational thinning of **Step 1**. We propose to divide the process of obtaining comprehension of the schema into many small tasks. This process is supported by the resulting forest hierarchy as follows. First the user studies each of the tree hierarchies of the emerging contexts. Then, the user approaches the challenge of studying the omitted relationships by concentrating each time on one pair of contexts and the inter-context relationships connecting their classes. For example, the user may choose the context rooted by *Diagnostic Procedure* and the context rooted by *Antibiotic Sensitivity Panel* (see Fig. 2). The user reviews the relationships connecting a class of one context with a class of another context. In the case of these two contexts, the subclass relationships are from *Antibiotic Sensitivity Panel to Laboratory Diagnostic Batteries*, and from *Pseudomonas Sensitivity (Panel 19)* to *CPMC Laboratory Diagnostic Batteries* (see Fig. 4). Furthermore, the user reviews other relationships existing between these two contexts. For example, the relationship *has-parts* is from *Pseudomonas Sensitivity (Panel 19)* to *CPMC Single-Result Laboratory Test.*

The user repeats this process for every pair of contexts which interest him. For each such pair, the number of relationships is small. Hence, we divide the large complex task of review of all the relationships in a schema into many small tasks which are much easier.

5. Conclusions

In this paper, we presented both a theoretical paradigm and a methodology to identify a meaningful forest subschema of a given object-oriented medical terminology schema. The extraction of the forest subschema employs two approaches, *informational thinning* and *partitioning*. We reviewed three rules which express limitations and refinements to the modeling of the terminology schema. Based on these three rules, a technique for medical terminology modeling called *disciplined modeling* was presented. A theorem guaranteeing the existence of a forest subschema was given. A human-computer interactive methodology was developed for finding the forest subschema. Such a forest subschema functions as a skeleton of the original medical terminology schema and supports comprehension efforts with respect to it. The methodology was applied to the MED object-oriented schema. The MED schema, containing 124 classes and 190 subclass relationships, was divided into 30 trees, 2 of which are single nodes. The 30 trees consist of logical unites averaging about 4 classes.

Acknowledgments

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, and the Center for Manufacturing Systems.

REFERENCES

1. 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.
2. Gu H, Perl Y, Geller J, Halper M, Liu L, Cimino JJ. Repesenting the UMLS as an OODB: Modeling issues and advantages. JAMIA 2000; 7 (1): 66-80.
3. Gu H, Halper M, Geller J, Perl Y. Benefits of an OODB representation for controlled medical terminologies. JAMIA 1999; 6 (4): 283-303.
4. Camion JJ, Clayton PD, Hripcsak G, Johnson S. Knowledge-based approaches to the maintenance of a large controlled medical terminology. JAMIA 1994; 1 (1): 35-50.
5. US Dept. of Health and Human Services, NIH, National Library of Medicine. Unified Medical Language System 1998.
6. Liu L, Halper M, Geller J, Perl Y. Using OODB modeling to partition a vocabulary into structurally and semantically uniform concept groups. To appear in: Transactions on Knowledge and Data Engineering.
7. Perl Y, Geller J, Gu H. Identifying a forest hierarchy in an OODB specialization hierarchy satisfying disciplined modeling. In: Proc. CoopIS'96. Brussels, Belgium 1996; 182-95.
8. Gu H, Perl Y, Geller J, Halper M, Singh M. A methodology for partitioning a vocabulary hierarchy into trees. Artificial Intelligence in Medicine 1999; 15 (1): 77-98.
9. Gary MR, Johnson DS. Computers and Intractability. New York: Freeman 1979.
10. Becker RI, Perl Y. Shifting algorithms for tree partitioning with general weighting functions. J Algorithms 1983; 4: 101-20.
11. Becker RI, Perl Y. The shifting algorithm technique for the partitioning of trees. Discrete Applied Mathematics 1995; 62: 15-34.
12. Becker RI, Perl Y, Schach S. A shifting algorithm for min-max tree-partitioning. J ACM 1982; 29: 56-67.
13. Perl Y, Schach S. Max-min tree-partitioning. J ACM 1981; 28: 5-15.
14. Kuno HK, Ra YG, Rundensteiner EA. The object-slicing technique: A flexible object representation and its evaluation. Technical Report CSE-TR-241-95, Univ. of Michigan 1995.
15. Bertino E, Martino L. Object-Oriented Database Systems: Concepts and Architectures. New York: Addison-Wesley Publishing Company 1993.
16. Geller J, Perl Y, Neuhold E. Structure and semantics in OODB class specifications. SIGMOD Record 1991; 20 (4): 40-3.
17. Buvac S, Fikes R. A declarative formalization of knowledge translation. In CIKM-95, Proc. 4th Int'l Conference on Information and Knowledge Management. Baltimore, MD 1995; 340-7.

18. Buvac S, Mason IM. Propositional logic of context. In: Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93), Washington, DC 1993; 412-9.
19. Iwanska L. Context in natural language processing. In: Working Notes of Workshop W13, IJCAI. Montreal, Canada 1995.
20. McCarthy J. Notes on formalizing context. In: 13th International Joint Conference on Artificial Intelligence, Chambery, France 1993; 555-60.
21. Miller GA. Wordnet: A lexical database for English. Communications of the ACM 1995; 38 (11): 39-41.
22. Lenat DB, Guha RV. Building Large Knowledge-Based Systems: Representation and Inference in the CYC project. Reading, MA: Addison-Wesley 1990.
23. Ringold EM, Nievergelt J, Deo N. Combinational Algorithms: Theory and Practice. Prentice Hall 1997.

Address of the authors:
Huanying (Helen) Gu,
Dept. of Health Informatics,
University of Medicine,
Dentistry of NJ, Newark, NJ 07103,
Tel.: (973) 972-0995,
Fax: (973) 972-1054,
E-mail: guhy@umdnj.edu