

## Controlled Vocabularies in OODBs: Modeling Issues and Implementation\*

LI-MIN LIU  
*CIS Dept., NJIT, Newark, NJ 07102*

limin@homer.njit.edu

MICHAEL HALPER  
*Dept. of Mathematics and Computer Science, Kean University, Union, NJ 07083*

mhalper@turbo.kean.edu

JAMES GELLER  
YEHOASHUA PERL  
*CIS Dept., NJIT, Newark, NJ 07102*

geller@homer.njit.edu

perl@homer.njit.edu

### Editor:

**Abstract.** A major problem that arises in many large application domains is the discrepancy among terminologies of different information systems. The terms used by the information systems of one organization may not agree with the terms used by another organization even when they are in the same domain. Such a situation clearly impedes communication and the sharing of information, and decreases the efficiency of doing business. Problems of this nature can be overcome using a controlled vocabulary (CV), a system of concepts that consolidates and unifies the terminologies of a domain. However, CVs are large and complex and difficult to comprehend. This paper presents a methodology for representing a semantic network-based CV as an object-oriented database (OODB). We call such a representation an Object-Oriented Vocabulary Repository (OOVR). The methodology is based on a structural analysis and partitioning of the source CV. The representation of a CV as an OOVR offers both the level of support typical of database management systems and an abstract view which promotes comprehension of the CV's structure and content. After discussing the theoretical aspects of the methodology, we apply it to the MED and InterMED, two existing CVs from the medical field. A program, called the OOVR Generator, for automatically carrying out our methodology is described. Both the MED-OOVR and the InterMED-OOVR have been created using the OOVR Generator, and each exists on top of ONTOS, a commercial OODBMS. The OOVR derived from the InterMED is presently available on the Web.

**Keywords:** Controlled Vocabulary, Object-Oriented Database, Object-Oriented Modeling, Terminology, Ontology, Ontology Modeling

### 1. Introduction

As the norm, different enterprises, whether they be in healthcare, manufacturing, financial services, or some other business, employ their own *ad hoc* terminologies that hinder communication and sharing of information. In the healthcare field, cases have been reported of differences between the terminologies used by different laboratories within the same hospital [7]! Such industries and areas of commerce

---

\* 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.

need controlled vocabularies (CVs)—systems of concepts that consolidate and unify the terminology of large application domains [9]. By maintaining a common, centralized CV, costly and time-consuming translation tasks can be eliminated from the lines of communication existing between different organizations and between the information systems each employs. A CV can also help standardize common information processing chores and thus reduce the overall cost of doing business.

The semantic network [27, 48] has proven to be an excellent modeling tool for CVs [8, 9]. However, in the face of a very large CV—encountered in many domains—a potential user or application developer might shy away from employing it due to the overwhelming complexity. It is well known that the components of the human cognitive apparatus have numeric limitations. For instance, it is accepted that human short-term memory is limited to “seven plus or minus two” chunks [32]. In a similar vein, there are well known limitations of the visual system [3]. Semantic networks and database design methodologies (like the ER model [6]) are popular because of their graphical representations, among other reasons. According to common experience, a graphical network display with around twenty concepts appears to be within human cognitive limitations, while for a graphical structure with thousands or tens of thousands of concepts, this is certainly not the case.

The reliance on a semantic network model for CVs can also be problematic from a practical standpoint. Although many useful semantic network processing systems exist (see, e.g., [2, 31, 45]), for unclear reasons, semantic networks have never taken off as commercial products. Because of this, technical support, multi-user access, documentation, and even adequate editing tools are often not available. This makes the use of a CV implemented as a semantic network a risky proposition for application developers whose environments often need exactly these missing features.

In this paper, we address these problems by introducing a methodology for representing a semantic network-based CV as an object-oriented database (OODB) [25, 54], a representation we call an Object-Oriented Vocabulary Repository (OOVR, pronounced “over”). Our methodology is based on a structural analysis and partitioning of the source CV. The schema yielded by this process is an important new layer of abstraction on top of the semantic network. The schema may, in fact, be several orders of magnitude smaller than the actual content of a large CV. Its size is more in line with the limitations of the human cognitive system than the potentially overwhelming concept network. A schema that correctly abstracts the major features of the network structures will be effective as a vehicle for exploring, studying, and ultimately comprehending the CV’s subject-matter. As stated by James Cimino, the principal designer of an extensive medical CV called the Medical Entities Dictionary (MED): “The schema highlights the essence of the vocabulary while hiding minutiae” [7]. Indeed, in [16], we showed how the schematic representation of the MED helped its designers readily uncover and correct several inconsistencies and errors in the MED’s original modeling. We have built an *analogical*, Web-based interface that exploits the two different view levels offered by the OODB representation (i.e., the schema-level view and the concept-level view) to facilitate interactive access to a CV [18]. By providing access to both the original

CV and the schema layer, the interface allows a user to choose a verbose or compact representation for the display of the vocabulary knowledge.

Of course, utilizing an OODB also allows us to leverage all the features offered by top-of-the-line OODB management systems, which are commercially available (see, e.g., [14, 39, 41, 44, 52]). In its OOV form, the CV can be accessed declaratively using the OQL standard [5] and other SQL extensions like ONTOS's OSQL [42], or using a "path" language such as XQL [24]. The OOV also offers a "low impedance" pathway [54] to the CV's knowledge for application programs like intelligent information-locators, decision-support systems, and end-user browsers which are being built using object-oriented programming languages and technology. Concepts represented as objects in the OOV are represented the same way within those applications. Therefore, the movement of and access to the concepts is greatly facilitated.

We have built a program called the *OOV Generator* which automatically carries out all phases of our methodology. It takes as its input a CV in a "flat" semantic-network format and produces an equivalent, fully populated OOV as its output. Both the formal aspects of the methodology and the details of this accompanying software will be described.

In order to demonstrate our methodology, we will be applying it to two existing CVs, the MED and the InterMED, both from the medical domain. The MED is a large CV containing about 48,000 concepts. It was developed and is currently in use at Columbia-Presbyterian Medical Center (CPMC) [8]. The InterMED was created as an offshoot of the MED [40, 46] for a more general healthcare setting. It comprises about 2,500 concepts. The representations of these two CVs as OODBs will be called the MED-OOV and the InterMED-OOV, respectively. Both are currently up and running on top of the ONTOS DB/Explorer, a commercial OODB management system [41, 42, 47]. The InterMED-OOV is accessible via the Web [43] using the browser described in [18].

Let us point out that even though both of the demonstration CVs are from the medical field, the methodology is applicable to CVs from any application domain. All we assume is that the vocabulary is—or can be—represented as a semantic network [53] of concepts containing a concept-subsumption (IS-A) hierarchy. We will discuss more of the details of the assumed representation in the next section.

The remainder of this paper is organized as follows. In Section 2, we present background material, including an overview of the general structure of CVs that are amenable to our methodology (including the MED and InterMED), and a discussion of related research work. Section 3 presents the details of the OODB representation of the CV. Software which automatically creates and populates an OOV with respect to a given CV is presented in Section 4. Section 5 contains the conclusions.

## 2. Background

### 2.1. Structural Form of a CV

A common formalism used in the construction of CVs, including both the MED and the InterMED, is the semantic network [27, 53]. A CV is a collection of nodes, each of which represents a single concept. A concept can have two kinds of properties, attributes and relationships. An attribute is a property whose value is from some data type (such as an integer or text string). A relationship, on the other hand, has as its value a reference to another concept in the CV. Each concept has a unique name which is often called its *term* [13]. A term is stored as the attribute *name* of its concept. An example of a relationship found in the MED, as well as in the InterMED, is *part-of* which links a source concept to the concept of which it is definitionally a part.

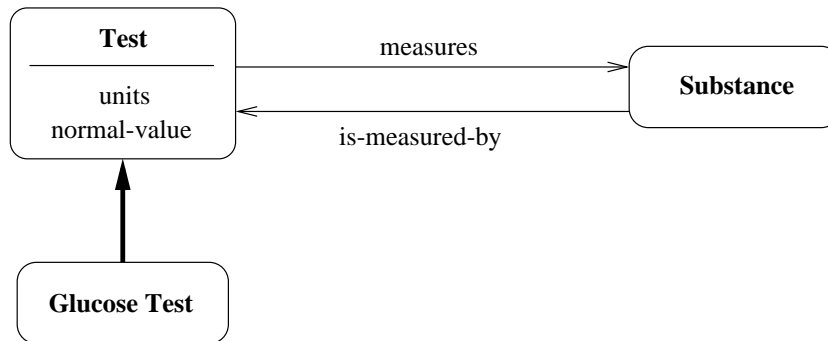


Figure 1. Concepts **Test**, **Substance**, and **Glucose Test**

We will be using the following graphical conventions when drawing the elements of a CV. A concept is a rectangle having rounded edges with its name (term) written inside. The names of any attributes introduced by the concept (when shown) are written below the concept's name and are separated from it by a line. Note that the values of such attributes will not be included in any diagrams. A relationship is a labeled arrow directed from the source concept to the target concept. As an example, we show the concepts **Test**<sup>1</sup> and **Substance** in Figure 1. The concept **Test** introduces the two attributes *units* and *normal-value* and the relationship *measures* directed to **Substance**. The concept **Substance** introduces the relationship *is-measured-by* (the converse of *measures*) but no attributes.

Following [8, 9], CVs are assumed to adhere to the design criteria of *nonredundancy* and *synonymy*, among others. The nonredundancy criterion states that a concept must be represented by a single node in the network. Of course, a con-

cept may be known by several names. This is addressed by the synonymy criterion which states that the CV should maintain these alternative names (i.e., synonyms) with the single node. To accommodate this, all concepts have the attribute *synonyms*. Thus, a concept's primary designation (i.e., its term) is stored in the attribute *name*, while any secondary designations (i.e., its synonyms) are stored in the attribute *synonyms*. Determining which name is primary and which others are secondary is strictly a design decision. The point is that all acceptable names are stored directly with the concept, and the concept is accessible via any of them.

An important feature of a CV is the concept-subsumption hierarchy, a singly-rooted, directed acyclic graph (DAG) of concepts connected via IS-A links. This hierarchy serves two main purposes. First, it supports the inheritance of properties among concepts within the CV. A subconcept is defined to inherit all the properties of its superconcept(s). For example, **Glucose Test** IS-A **Test** and therefore inherits all of **Test**'s properties. In other words, the set of properties of **Glucose Test** is a superset of the properties of **Test**. An IS-A link is drawn as a bold, unlabeled arrow directed from the subconcept to the superconcept as shown in Figure 1. Because the IS-A hierarchy is a DAG, a concept may have more than one superconcept (or parent). In such a case, the subconcept inherits from all its parents. As we will discuss further below, most CVs exhibit what we call *sparse inheritance* which influenced our design decisions.

The second purpose of the hierarchy is to support reasoning in the form of subsumption-based inferences. For example, using the fact that **Tetracycline** IS-A **Antibiotic**, a decision-support system can infer that a patient is on antibiotics from an entry in a clinical database stating that the patient is taking tetracycline.

As noted, the IS-A hierarchy is singly-rooted. We refer to the root concept as **Entity**. Note that this requirement does not affect generality because such a concept can be artificially introduced, if necessary.

We will be using the MED and the InterMED as source vocabularies to demonstrate our methodology. The MED comprises about 48,000 concepts which are connected by more than 61,000 IS-A links and 71,000 non-hierarchical (i.e., non-IS-A) relationships. The figures for the InterMED are as follows: approximately 2,500 concepts, 3,400 IS-A links, and 3,500 non-hierarchical relationships.

Both the MED and InterMED obey the following rule pertaining to the introduction of properties into the vocabulary.

**Rule (Uniqueness of Property Introduction):** A given property  $x$  can be introduced at only one concept in the CV.

If other concepts also need  $x$ , then they must be defined as descendants of the concept at which  $x$  is introduced and obtain it by inheritance. We will be assuming that any CV to which our methodology is to be applied satisfies the above rule. Note that this is not overly restrictive because if there is a need to introduce a property  $x$  at several independent concepts, then an "artificial" superconcept of these can be created for the purpose of defining  $x$  [7].

## 2.2. *Related Research Work*

Computerizing natural language concepts has long been a major goal of computer scientists. Various forms of semantic networks [2, 27, 48, 53], knowledge representation languages [23, 31, 35], ontologies [38], and semantic data models [19, 20] have been recruited to tackle this task. While most attempts have been limited to small domains or “toy” applications, there have been a number of notable exceptions such as CYC [29] and WordNet [33].

Aside from the MED and the InterMED, the medical field has seen the introduction of a number of CVs. These include UMLS [49], SNOMED [10], and ICD9-CM [50]. A descriptive semantic network called Structured Meta Knowledge (SMK), employing a terminological knowledge-base, has been used to capture the semantics of patients’ medical records [15].

An object-oriented framework has previously been employed as a modeling platform for thesauri used in (natural) language-to-language translation [12, 13]. TEDI, a terminology editor, was built in the same context as a tool for extracting relevant information from hypermedia documents [34]. The O<sub>2</sub> OODB system [11, 47] has been used to store portions of a general English dictionary based on a “feature structure” description of its entries [21]. In a similar effort, [55] presents a technique for storing a dictionary in an ObjectStore database [26, 47].

Database technology has been used as a means for bringing persistence to knowledge-based systems. In [22], the EXODUS object manager [4] is used as a subsystem of a frame representation system [23]. A storage model, based on techniques previously proposed for OODBs [51], has been employed as the basis for storing Telos knowledge-bases on disk [36]. Both these efforts sought to incorporate their database subsystems transparently. In contrast, we are directly utilizing a commercial OODB system for the representation of our CV. After the conversion of the CV into the form of an OODB, the original semantic network version of the CV is no longer needed. Users of the vocabulary, whether they be programmers or casual browsers, can directly access the vocabulary through the various mechanisms provided by the OODB management system.

A preliminary version of our methodology appeared previously in [30]. In that paper, we applied it to the InterMED. In this paper, a substantially cleaner way of capturing our methodology is presented, and some gaps in the original treatment in [30] are filled in. We also show the application of the methodology to the much larger MED. In addition, we describe the algorithms which automatically carry out the conversion of a CV into its OOV<sub>R</sub> representation.

## 3. **Representing a CV as an OODB**

In this section, we present our methodology for representing a CV as an OODB. The methodology attacks this modeling problem by extracting a schematic representation from the source CV and then assigning the CV’s concepts to appropriate object classes. In the source CV, there are concepts; in the OOV<sub>R</sub>, there will be objects which denote those concepts. Our methodology derives a schema with far

fewer classes than the number of concepts in the source CV. The tasks are to determine how many classes are necessary, what the classes will look like, what their relationships will be, and to which classes the various concepts will belong.

In the following, we first present the version of our methodology which is applicable to CVs that do not contain *intersection concepts*, a notion which will be further discussed and formally defined below. The extended methodology that encompasses CVs exhibiting intersection concepts is described in Section 3.2.

### 3.1. OODB Schema for CVs without Intersection Concepts

The OODB schema produced by our approach is derived automatically from an overall structural analysis of the CV. It is based on the partitioning of the CV into groups of concepts that exhibit the exact same set of properties. To be more precise, we will need the following definition, where we use  $P(x)$  to denote the entire set of properties of the concept  $x$ .

**Definition 1 (Area):** Let  $A$  be a non-empty set of concepts such that  $\forall x, y \in A, P(x) = P(y)$ . For such a set, let  $P(A) = P(x)$  for some  $x \in A$ . That is,  $P(A)$  is the set of properties exhibited by each of  $A$ 's members.  $A$  is called an *area* if there does not exist a set of concepts  $B$  such that  $\forall v, w \in B, P(v) = P(w), P(B) = P(A)$ , and  $A \subset B$ . In other words,  $A$  is an area if it is the maximal set of concepts which exhibit the set of properties  $P(A)$ .  $\square$

By definition, if  $A_1$  and  $A_2$  are areas, and  $A_1 \neq A_2$ , then  $A_1 \cap A_2 = \emptyset$ . That is, distinct areas are always disjoint. Given this fact, it is relatively straightforward to define the OODB schema of a CV if one knows all the CV's areas. For each area  $A$ , a class having the properties  $P(A)$  is created. While this description leaves out some important details, it does capture the essence of the approach. The problem, of course, lies in the identification of the areas.

In the remainder of this section, we will be describing the process of identifying all areas in the CV—and, hence, partitioning the CV into mutually exclusive sets—under the assumption that the CV does not contain *intersection concepts*, which will be formally defined in the next subsection. (Informally, an intersection concept is one that does not introduce any new properties but still has a different set of properties from all its parents due to inheritance from several of them.) After discussing the area-partitioning, we will present the details of the OODB schema derived directly from it and describe how the CV is stored in the database.

Let us point out that the methodology as presented in this section is sufficient in itself for a CV whose IS-A hierarchy is a tree. Examples of such CVs include ICD-9 [50] and AHFS [1]. Even if the hierarchy is a DAG, the methodology will still be suitable if the CV is devoid of intersection concepts, a condition that can be detected algorithmically. An example of such a CV is NDC [37]. The main reason for delaying consideration of intersection concepts to the next section is that it greatly simplifies the presentation.

The identification of areas follows the pattern in which the concepts' properties are introduced into the CV. In this regard, we will need the following two definitions.

In the first, we use  $\Pi(x)$  to denote the set of properties intrinsically introduced or defined by the concept  $x$  (as opposed to those that are inherited by it).

**Definition 2 (Property-Introducing Concept):** A concept  $x$  is called a property-introducing concept if  $\Pi(x) \neq \emptyset$ , i.e., if it intrinsically defines one or more properties.  $\square$

**Definition 3 (Direct Property-Introducing Descendant [DPID]):** Let  $v$  and  $w$  be property-introducing concepts, and let  $w$  be a descendant of  $v$  with respect to the IS-A hierarchy. The concept  $w$  is called a direct property-introducing descendant (DPID) of  $v$  if there exists an upwardly directed IS-A path (there can be more than one) from  $w$  to  $v$  that does not contain another property-introducing concept.  $\square$

The property-introducing concepts form the basis for the areas of the CV. In fact, in a CV devoid of intersection concepts, we can equivalently state the definition of area in terms of property-introducing concept as follows:

**Definition 4 (Area [equivalent redefinition]):** An area is a set of concepts containing a property-introducing concept  $v$  and all of  $v$ 's descendants excluding its DPIDs and their respective descendants.  $\square$

Clearly, an area can contain only a single property-introducing concept. Any descendants that are also property-introducing concepts will define new areas of their own. From a top-down vantage point, the property-introducing concept is the highest node in an area, and in this sense it “starts” the area. For this reason, we refer to that concept as the *root* of the area and use it when we need to assign a name to the area.

To illustrate the partitioning of a CV, we show three areas  $A$ ,  $B$ , and  $C$  in Figure 2. The concepts are drawn as rectangles with rounded edges, while the areas are shown as large rectangles. Note that the root of area  $A$  (i.e., the concept **A**) introduces the single attribute  $x$ . Area  $A$  extends down to, but excludes, node **B** which is a DPID of **A**. **B** defines the attribute  $y$  as well as the relationship  $r$  and serves as the root of area  $B$ . Finally, area  $C$  has the root **C** which introduces attribute  $z$  and the relationship  $r'$ , the converse of  $r$ . The ellipses in the figure indicate the omission of additional concepts above the areas  $A$  and  $C$ .

As concrete examples, we show portions of three areas from the MED in Figure 3. The concept **Event Component** introduces the new attribute *event-component-display-name* and is thus the root of a new area, “Event Component” area. The concept **Pharmacy Order Component** also resides in that area. **Drug Order** introduces two properties: the attribute *drug-role-code*, and the relationship *ordered-drug* pointing at **Pharmacy Item**. **Drug Order** therefore roots the “Drug Order” area. Finally, **Pharmacy Item** defines the attribute *dose-strength-units* (among others) and the relationship *ordered-in*, and serves as the root of “Pharmacy Item” area.

Another example is the area rooted at the concept **Entity**, which, as we noted above, is assumed to be the top concept of all CVs. **Entity** introduces a number of properties (including *name*) and is therefore the root of “Entity” area.

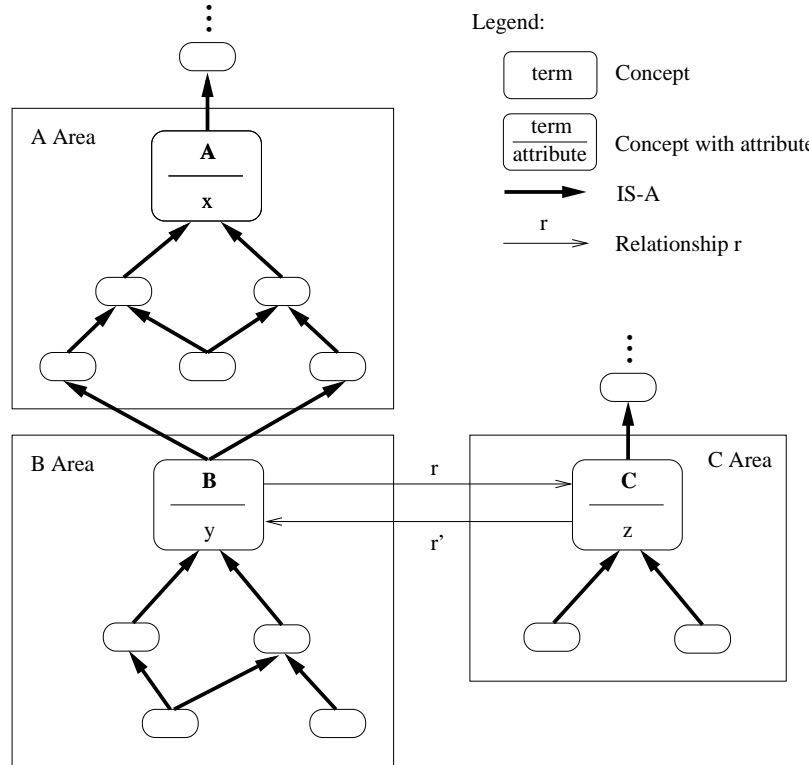


Figure 2. Three areas of a CV

Once all the areas of the CV have been identified, the OODB schema can be created as follows. For each area  $A$ , define a class (called  $A\_Area$ ) whose instances will be exactly the concepts in  $A$ , including  $A$ 's root (call it  $r_A$ ). As all concepts in  $A$  possess the same set of properties as  $r_A$  [namely,  $P(r_A)$ ], it seems natural to define the properties of  $A\_Area$  to be  $P(r_A)$ . However, this ignores the fact that  $r_A$  may have inherited some of its properties rather than defined them all intrinsically. In fact, it is only necessary to endow  $A\_Area$  with the set of intrinsic properties of  $r_A$ ,  $\Pi(r_A)$ . The rest of the properties can be obtained via OO subclass inheritance. The root  $r_A$  inherits from its parents—which reside in areas distinct from  $A$ —those properties that it does not itself introduce. From this, it can be seen that  $A\_Area$  should inherit from all the classes which represent areas that contain a parent of  $r_A$ . That is,  $A\_Area$  should be a subclass of all those classes. To characterize this precisely, let us define the following.

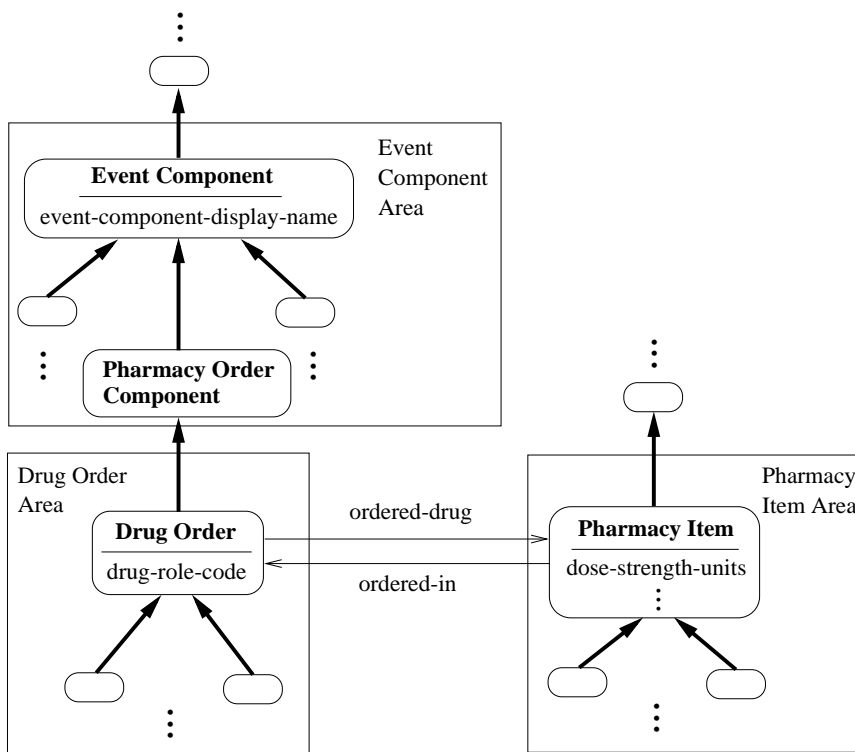


Figure 3. Three areas from the MED

**Definition 5 (Parent [Area]):** Let  $B$  and  $C$  be areas. If a parent of  $r_C$ , the root of  $C$ , resides in  $B$ , then  $B$  is called a parent (area) of  $C$ .  $\square$

Continuing, let the areas  $T_1, T_2, \dots, T_n$  ( $n \geq 1$ ) be the parents of area  $A$ . In other words,  $T_1, T_2, \dots, T_n$  each contain at least one parent concept of  $r_A$ . It will be noted that  $P(A) = \Pi(r_A) \cup P(T_1) \cup P(T_2) \cup \dots \cup P(T_n)$ . That is, the properties of the area  $A$  are gathered from its root and all its parents. To capture this in the schema,  $A\_Area$  is defined with the set of intrinsic properties  $\Pi(r_A)$ , and as a subclass of all  $n$  classes  $T_1\_Area, T_2\_Area, \dots, T_n\_Area$  representing, respectively, the areas  $T_1, T_2, \dots, T_n$ . Let us note that it is possible that one of these classes, say,  $T_i\_Area$  is a parent or an ancestor of another, say,  $T_j\_Area$ . In such a case, defining subclass relationships between  $A\_Area$  and both  $T_i\_Area$  and  $T_j\_Area$  would lead to a “short circuit” (i.e., a materialization of a transitive subclass connection) in the OODB schema. The relationship between  $A\_Area$  and  $T_i\_Area$  is clearly redundant because, in such a situation,  $P(T_i) \subset P(T_j)$ . Therefore, the subclass relationship

to  $T_j\text{-Area}$  gives  $A\text{-Area}$  all the properties that the relationship with  $T_i\text{-Area}$  would provide. Due to this, the subclass link between  $A\text{-Area}$  and  $T_i\text{-Area}$  is omitted.

Since all concepts (except for **Entity**) have superconcepts,  $r_A$ 's parents will probably have their own parents, and so the subclass relationships of the schema will branch upward in a DAG structure until they reach the class  $Entity\text{-Area}$  representing the top area (i.e., "Entity" area) of the CV.

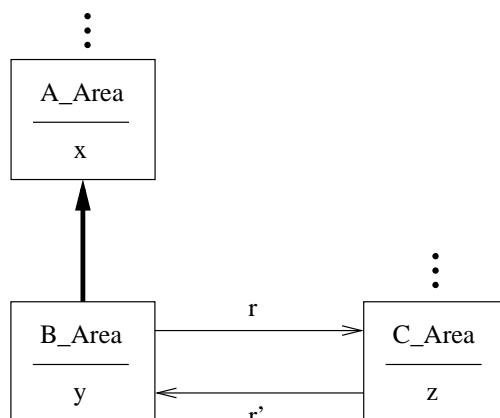


Figure 4. Area classes corresponding to the three areas in Figure 2

In Figure 4, we show the three classes that represent the areas from Figure 2. A class is drawn as a rectangle; its intrinsic attributes are listed inside beneath its name. A subclass relationship is denoted as a bold arrow directed upward from the subclass to its superclass, and a relationship is represented by a labeled thin arrow. As we see,  $A\text{-Area}$  has the attribute  $x$  defined by the concept **A**, the root of the area. Likewise,  $B\text{-Area}$  has the attribute  $y$  and the relationship  $r$ , and  $C\text{-Area}$  has the attribute  $z$  and the relationship  $r'$ . Note that  $B\text{-Area}$  is a subclass of  $A\text{-Area}$  because area  $A$  contains the parents of concept **B**, the root of area  $B$ .

Referring back to the sample areas from the MED in Figure 3, "Event Component" area would have the corresponding class  $Event\_Component\_Area$ , which defines the attribute *event-component-display-name*. The "Drug Order" area would have the class  $Drug\_Order\_Area$  with the attribute *drug-role-code* and the relationship *ordered-drug*. The class  $Pharmacy\_Item\_Area$  with the appropriate properties would denote the "Pharmacy Item" area. Lastly, "Entity" area would be associated with the class  $Entity\_Area$  possessing the property *name* and so forth. It should be noted that  $Entity\_Area$  is the root class of the schema.

One final aspect of the OODB that warrants special consideration is the representation of the CV's IS-A hierarchy. We have already used this feature and, more specifically, its inheritance mechanism to derive the subclass relationships of the

schema. However, it is still required that the individual concepts themselves (at the instance-level of the OODB) be connected to their parents (and vice versa). This can be done by once again noting that all concepts in the CV, except for the root, have parents. Thus, we equip all concepts with two additional generic relationships, “has-superconcept” and “has-subconcept,” which connect a concept to its parents and children, respectively. Within the semantic network, these relationships can be seen as being defined by **Entity** and inherited by every other concept. Following that, they are defined reflexively at the root class *Entity\_Area* of the OODB schema. If, in the original CV, concept  $x$  IS-A  $y$ , then, in the OODB, the object representing  $y$  is a referent of  $x$  with respect to the *has-superconcept* relationship; conversely,  $x$  is a referent of  $y$  via *has-subconcept*.

Because the direct extension of a class in the OODB schema is precisely one area, we refer to it as an *area class*. Overall, the schema comprises a collection of area classes. Since it is an abstraction of the property definitions and accompanying inheritance that occur within a CV as modeled by a semantic network, we call this kind of schema a *network abstraction schema*.

It is important to point out that the area-partitioning described above does not lead to a proliferation of classes in the OODB schema. There are two major reasons for this: (a) typically, there is a small number of distinct properties in a CV compared to the total number of concepts; and (b) the “uniqueness of property introduction” rule. Due to the latter, one does not find redundant property introductions strewn throughout the CV. Since property-introducing concepts always start new areas, this helps keep their numbers down.

Point (a) is, fortunately, a general characteristic of CVs which stems from the fact that they are definitional structures rather than dynamic data stores. In the InterMED, there are only 51 distinct properties for a total of about 2,500 concepts. For the MED, the number is 150 properties for approximately 48,000 concepts. As a consequence of this, very few concepts intrinsically introduce properties; most properties are inherited. There are just 26 property-introducing concepts in the InterMED and 57 in the MED. It is interesting to contrast this sparseness of property introduction in a CV with the denseness of the same in a typical OODB schema where at (almost) every class we expect to find the definitions of new properties.

In Figure 5, we show the OODB schema for the InterMED in the case where its intersection concepts (and their descendants) are omitted. It should be noted that the subclass hierarchy of such a schema does not necessarily have a tree structure because a property-introducing concept can have parents in many different areas. The concept **Chemical** is an example. See the class *Chemical\_Area* in the figure.

### 3.2. CVs with Intersection Concepts

The problem of identifying areas is made more difficult when intersection concepts are present in the CV. Before formally defining what we mean by this notion, let us give an illustration. In Figure 6, we show a more complex version of the CV excerpt

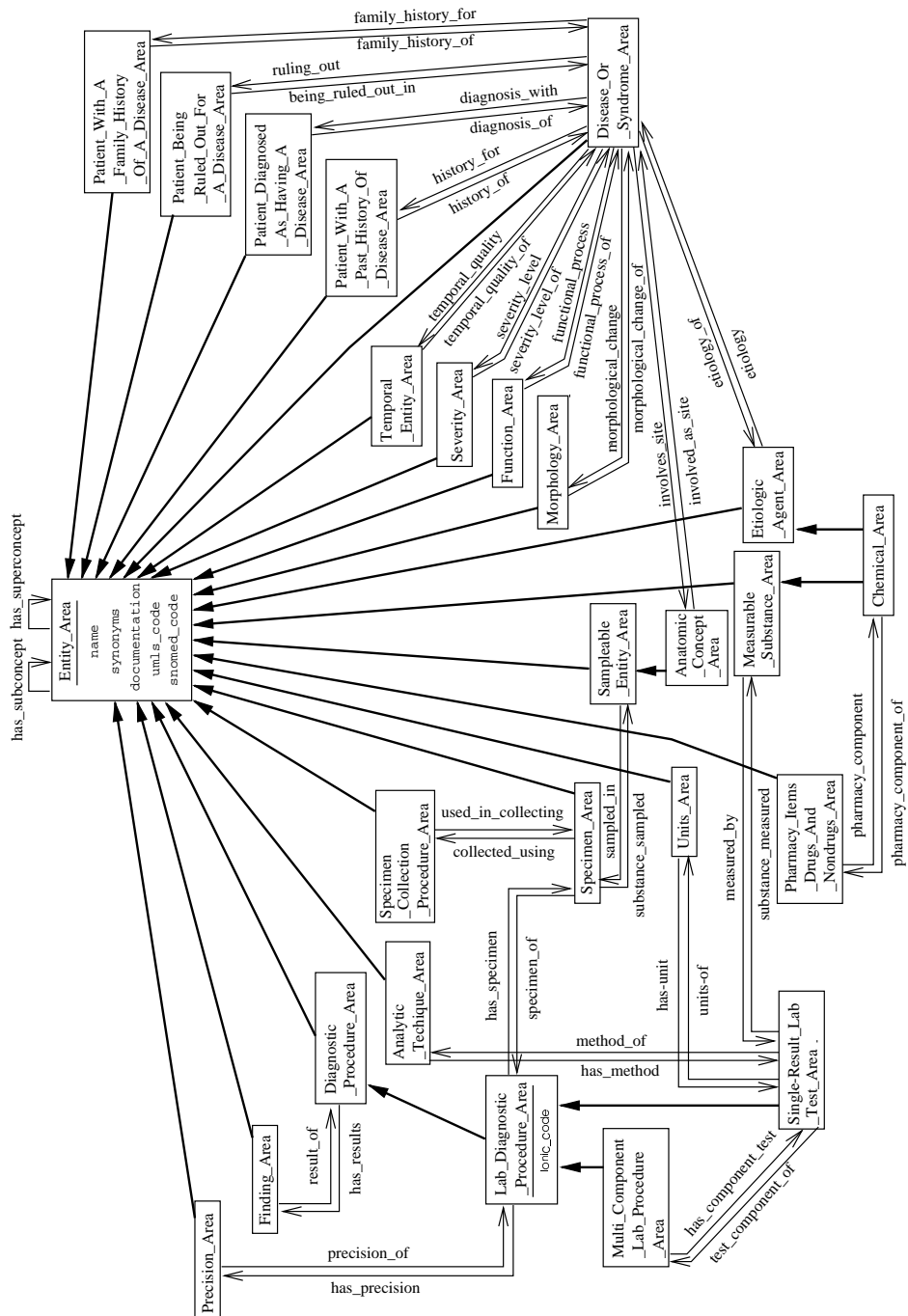


Figure 5. Schema for the InterMED excluding intersection concepts

appearing in Figure 2. In the following, we will call two concepts *ancestrally related* if there exists an ancestor/descendant relationship between them.

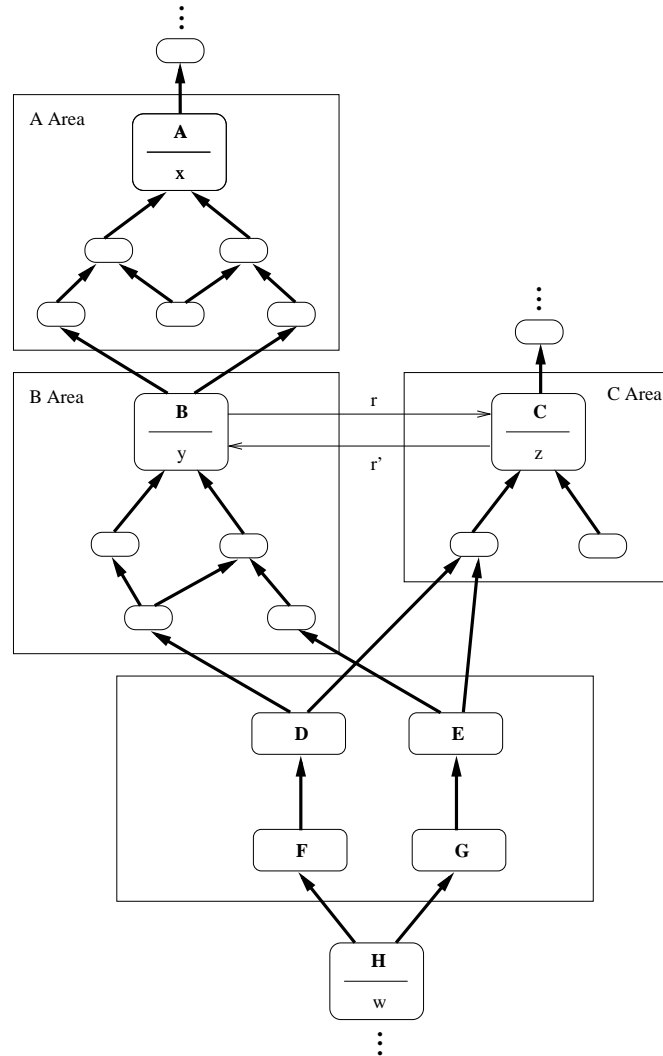


Figure 6. A more complex version of Figure 2 containing intersection concepts **D** and **E**

According to our specification of an area in terms of a property-introducing concept given in the previous section, the concepts **D**, **E**, **F**, and **G** (enclosed in a large box) should belong to the area rooted at **B** (i.e., area *B*) since they are “between”

it and one of its DPIDs, namely, **H**. However, on closer inspection, they similarly belong to the area  $C$ . On the other hand, those concepts cannot belong to area  $B$  (area  $C$ ) since they have extra properties not in  $P(B)$  ( $P(C)$ ) which they inherit from  $C$  ( $B$ ). In fact, they make up a new area of their own, even though none is a property-introducing concept. They obtain their properties via inheritance from two other areas that are, in a sense, independent. Each of the concepts **D** and **E** can be seen to lie at the juncture of some inheritance paths emanating downward from the ancestrally unrelated property-introducing concepts **B** and **C**. For this reason, we call **D** and **E** *intersection concepts*. While we could formalize this notion in terms of IS-A paths, it is simpler to do it as follows.

**Definition 6 (Intersection Concept):** Let  $x$  be a concept which is not a property-introducing concept and which has multiple superconcepts  $t_1, t_2, \dots, t_n$  ( $n > 1$ ). Then  $x$  is called an intersection concept if it satisfies the following:  $\forall i: 1 \leq i \leq n, P(x) \neq P(t_i)$ . In other words, the set of properties of  $x$  differs from the set of properties of each of  $x$ 's parents.  $\square$

Note that a concept having a single parent cannot be an intersection concept: Its properties could only differ from its parent's if it intrinsically introduced some, in which case it would be a property-introducing concept. Furthermore, at least two of an intersection concept's parents must be from different areas. Another characteristic of intersection concepts, which has bearing on the area-partitioning of the CV, is that two such concepts having identical sets of properties (e.g., **D** and **E**) cannot be ancestrally related.

For CVs containing intersection concepts—e.g., the MED and the InterMED—there are two different kinds of areas. The first, discussed in the previous subsection, starts at a single property-introducing concept and extends downward until other property-introducing concepts or intersection concepts are reached. The second kind, defined below, is rooted in one or more intersection concepts and branches down in an identical manner to that of the first kind. We will call the first kind of area a *property-introducing area*; the second will be referred to as an *intersection area*. To define these more precisely, we will need the following.

**Definition 7 (Direct Intersection Descendant [DID]):** Let  $v$  be a property-introducing concept,  $w$  be an intersection concept, and let  $w$  be a descendant of  $v$  with respect to the IS-A hierarchy. The concept  $w$  is called a direct intersection descendant (DID) of  $v$  if there exists an upwardly directed IS-A path (there can be more than one) from  $w$  to  $v$  that does not contain another property-introducing concept or intersection concept.  $\square$

In Definitions 3 and 7, we use a property-introducing concept  $v$  as the ancestor with respect to which DPID and DID are defined. It is a straightforward matter to define both DPID and DID with respect to an intersection concept ancestor, as well. We omit these two additional definitions for the sake of brevity. We will, however, be utilizing them and referring to an intersection concept's DPIDs and DIDs below.

**Definition 8 (Property-introducing Area):** A property-introducing area is a set of concepts containing a property-introducing concept  $v$  and all of  $v$ 's descendants excluding its DPIDs and DIDs and their respective descendants.  $\square$

**Definition 9 (Intersection Area):** Let  $E = \{e_1, e_2, \dots, e_n\}$  be a set of intersection concepts (as defined in Definition 6) such that  $\forall i, j : 1 \leq i, j \leq n, P(e_i) = P(e_j)$ . Furthermore, let  $E$  be maximal, i.e., there does not exist an intersection concept  $s \notin E$  such that  $P(s) = P(e_k)$  for some  $k$ . Now,  $\forall i : 1 \leq i \leq n$ , let  $E_i$  be the set containing  $e_i$  and all of  $e_i$ 's descendants excluding its DPIDs and DIDs and their respective descendants. The set  $I = E_1 \cup E_2 \cup \dots \cup E_n$  is called an intersection area.  $\square$

The concepts  $e_1, e_2, \dots, e_n$  will be called the roots of the intersection area because each starts a portion of it. These portions may overlap. The name of the area can be chosen arbitrarily (perhaps by the vocabulary administrator) from among the  $n$  concepts.

Referring back to Figure 6, we see that the four concepts, **D**, **E**, **F**, and **G**, constitute an intersection area. Both **D** and **E** are intersection concepts and serve as the roots of the area. The concepts **F** and **G** are not roots. Indeed, they are not intersection concepts but happen to reside in an intersection area by dint of their IS-A connections to intersection concepts.

In the InterMED, only 2 of its 2,500 concepts are intersection concepts. For the MED, it is 1,332 out of 48,000. An example of an intersection concept in the InterMED is **Water** whose parents reside in two areas: “Sampleable Entity” area and “Chemical” area. An intersection concept from the MED is **Chloramphenicol Preparations** whose parents belong to three areas, “Antihistamine Drugs,” “Drug Allergy Class,” and “DEA Controlled Substance Category.”

In the OODB schema, property-introducing areas are treated in the same manner described previously. One *property-introducing (area) class* is created for each property-introducing area. The properties and subclass relationships of the class are determined by the area's root and its parents, respectively.

For an intersection area, a class [called an *intersection (area) class*] is defined as we previously defined a property-introducing class for each property-introducing area. However, this class contains *no* intrinsic properties. Instead, it gets all its properties via inheritance—just as its roots do.

The subclass relationships originating with an intersection class are once again determined by the parents of a root. A subtlety that arises here comes from the fact that the parents of one root may reside in areas different from those of the parents of another root. Even so, the union of the parents' sets of properties with respect to one root is always the same as the union with respect to any other. If not, the roots would belong to different areas.

To illustrate this point, consider the six concepts,  $Q_1$  through  $Q_6$ , shown in Figure 7. Concepts  $Q_1$ ,  $Q_2$ , and  $Q_3$  introduce the attributes  $a$ ,  $b$ , and  $c$ , respectively, and therefore serve as the roots of three different property-introducing areas. (In this simplified configuration, they are the only concepts in their respective areas.)

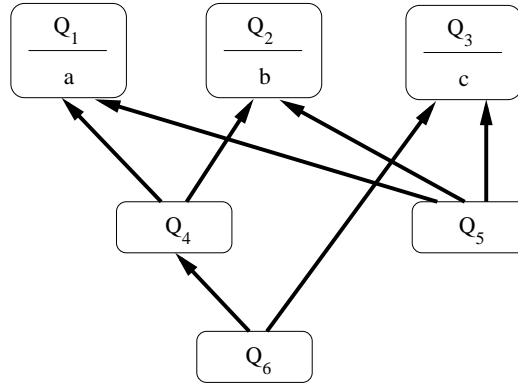


Figure 7. Parents of the roots of an intersection area residing in different areas

$Q_4$  is an intersection concept possessing the properties  $a$  and  $b$  obtained via inheritance from its parents. The interesting aspect of the figure involves the concepts  $Q_5$  and  $Q_6$ . Both are intersection concepts and possess all three attributes,  $a$ ,  $b$ , and  $c$ . Therefore, they are roots of the same intersection area. However,  $Q_5$  has the three parents  $Q_1$ ,  $Q_2$ , and  $Q_3$  residing in their own property-introducing areas. Concept  $Q_6$  shares the parent  $Q_3$  with  $Q_5$ , but has only one other parent  $Q_4$ , which, as noted, is an intersection concept. To summarize, we see that the sets of parent areas of an intersection area are not unique. They can differ with respect to its various roots.

Because of this, there are potentially many equivalent subclass configurations (and, hence, OODB schemas) that can be used to represent such an intersection area. The question is: Which of these should be chosen? Our answer is to select the root whose parents collectively reside in the fewest areas and define the subclass relationships with respect to those area classes. This minimizes the required number of subclass relationships.

To demonstrate this, let us refer back to Figure 7. We would select the root  $Q_6$  for the intersection area rooted at both  $Q_5$  and  $Q_6$  because  $Q_6$ 's parents reside in two areas while  $Q_5$ 's reside in three. The subclass relationships for this intersection area class would be directed to two classes, one representing the property-introducing area of concept  $Q_3$  and the other representing the intersection area rooted at  $Q_4$ .

In general, the process of determining the subclass relationships for an intersection class is as follows. Let  $I$  be an intersection area and let  $r_i$  be one of its roots whose parents reside in the fewest different areas.<sup>2</sup> Moreover, let  $T_1, T_2, \dots, T_n$  be all the areas containing at least one of  $r_i$ 's parents. Then the class  $I\_Area$ , the intersection class for  $I$ , is defined as a subclass of  $T_1\_Area$ ,  $T_2\_Area$ , through  $T_n\_Area$ , the respective area classes of  $T_1, T_2, \dots, T_n$ .

As pointed out above, an intersection concept's parents must reside in at least two different areas, so an intersection class will have at least two superclasses. This demonstrates that the OODB schema, for this type of CV, will exhibit multiple inheritance.

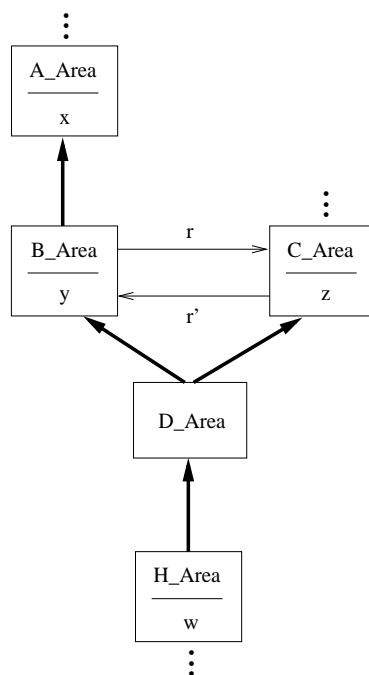


Figure 8. Area classes for the areas in Figure 6

To illustrate the schema construction, we show the area classes for the areas from Figure 6 in Figure 8. The ellipses indicate the omission of subclass relationships and additional classes that would appear in an expanded drawing. The three classes, *A\_Area*, *B\_Area*, and *C\_Area*, are defined as discussed previously. Each is a property-introducing class. The class *D\_Area* is an intersection class representing the intersection area containing the four concepts, **D**, **E**, **F**, and **G**. Both **D** and **E** are roots of the area and are thus viable designations for it. The name *D\_Area* was chosen because **D** appears first in a scan of the area. The class does not have any intrinsic properties. It is a subclass of *B\_Area* and *C\_Area* because **D**'s parents (as well as **E**'s) belong to the areas *B* and *C*. *H\_Area* is a property-introducing class which introduces the attribute *w* and is a subclass of *D\_Area*.

In Figure 9, we show the entire InterMED-OOVR schema comprising a total of 28 area classes and 30 subclass relationships. Of the 28 classes, 26 are property-

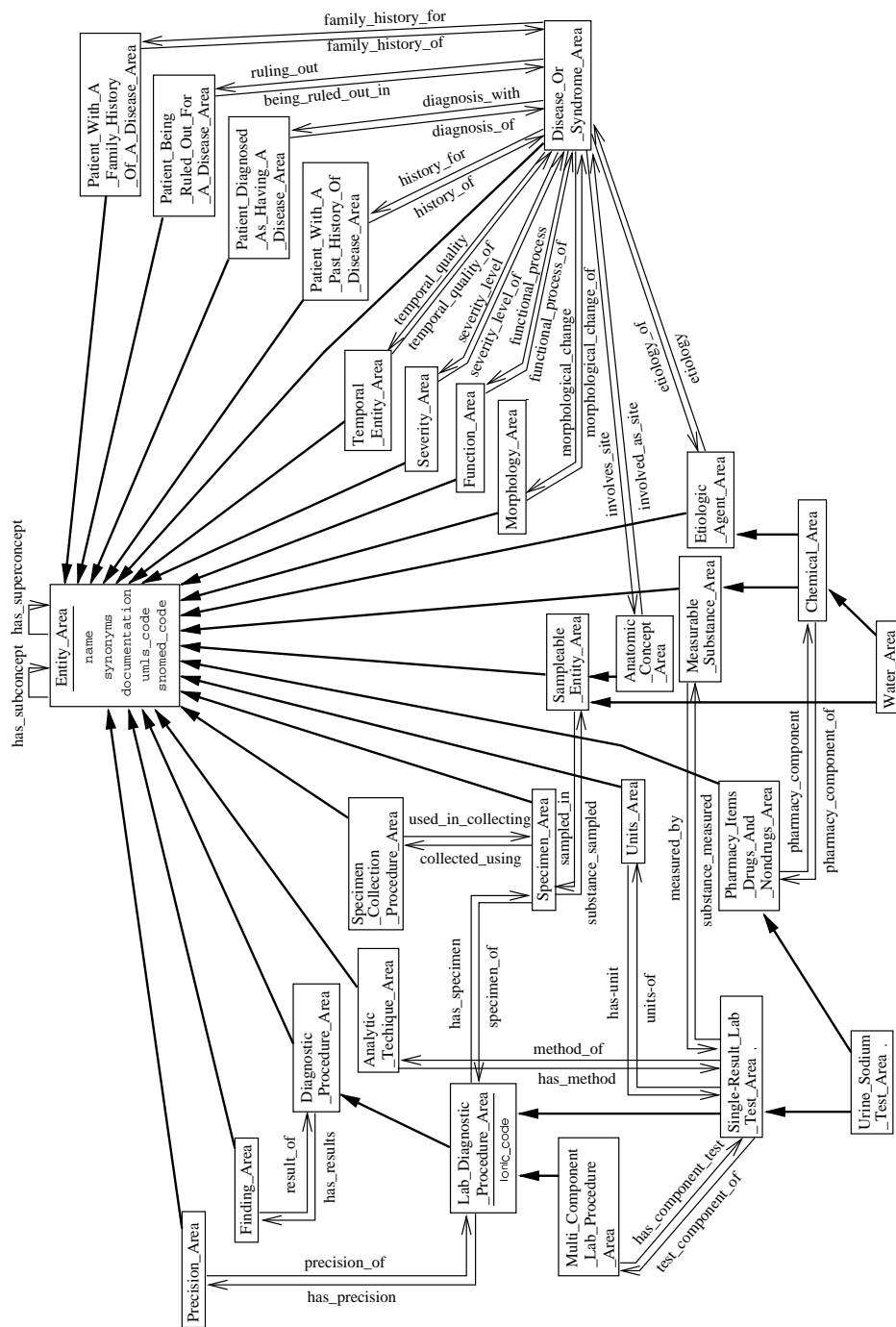


Figure 9. InterMED-OOVR schema

introducing classes and 2 are intersection classes. One of the intersection classes is *Water\_Area* which is a subclass of *Sampleable\_Entity\_Area* and *Chemical\_Area*. The other is *Urine\_Sodium\_Test\_Area* which has the parents *Single\_Result\_Lab\_Test\_Area* and *Pharmacy\_Items\_Drugs\_and\_Nondrugs\_Area*.

The schema for the MED-OOVR contains 90 classes (57 property-introducing classes, 33 intersection classes) and 134 subclass relationships. Due to its large size, it is not convenient to show the entire schema graphically on one page. In Figure 10, we show only the 57 property-introducing classes. In order to save space, we have drawn the properties as numbers. The corresponding property names can be found in Figure 11. Figure 12 contains all the property-introducing classes (with all properties omitted) as well as the following six intersection classes: *Antihistamine\_Drugs\_Area*, *Chloramphenicol\_Preparations\_Area*, *Organism\_Area*, *Wucheria\_Bancrofti\_Area*, *Black\_Piedra\_Area*, and *Abnormal\_Finding\_in\_Body\_Substance\_Area*. It should be noted that it is possible for one intersection class to be a subclass of another intersection class. This is demonstrated by three of the intersection classes, *Chloramphenicol\_Preparations\_Area*, *Wucheria\_Bancrofti\_Area*, and *Black\_Piedra\_Area*. Moreover, *Black\_Piedra\_Area* is two levels below the intersection class *Organism\_Area*. Note also that *Chloramphenicol\_Preparations\_Area* has three parents.

An important aspect of our methodology is the compactness of the resultant OODB schema. For the InterMED, which contains about 2,500 concepts, the schema has merely 28 area classes—about an 80-to-1 reduction. The MED contains approximately 48,000 concepts and has a schema of around 90 classes—about a 500-to-1 ratio! Additionally, we find a slow growth rate for the schemas with respect to the size of the source CVs. The content of the MED is nineteen times larger than that of the InterMED, yet its schema is only about three times the size.

In [16], we showed how the compactness of the schema helped a vocabulary administrator uncover mistakes that had been introduced into the MED. We also discussed how this representation can be used as a tool for comprehending the content of a CV. In fact, deriving an OODB schema for a CV was helpful for us both by the process and the result. The process of finding a schema gave us a source for asking intelligent questions about the vocabulary, the answers to which added insights to our comprehension of its knowledge content. The result, the schema itself, is a knowledge-rich abstraction that allows us to split the comprehension process into two steps. In the first step, a person studying the schema gets a good understanding of the CV's overall structure. In the second step, a person using the schema as a road map can then advance to studying selected areas of the CV in detail. In summary, a schema adds a valuable layer of abstraction on top of the large and complex content of a CV. We have built a program that utilizes this separation of CV and schema along with what we call "analogical forms" to provide an enhanced interface to CVs [18]. Using this program, a traversal of the CV can begin at the schema level and continue until the proper class is identified; at that point, the traversal can proceed at the concept level. Further abstraction of a CV can be achieved by partitioning the set of concepts of an area class into smaller units, as suggested in [17].

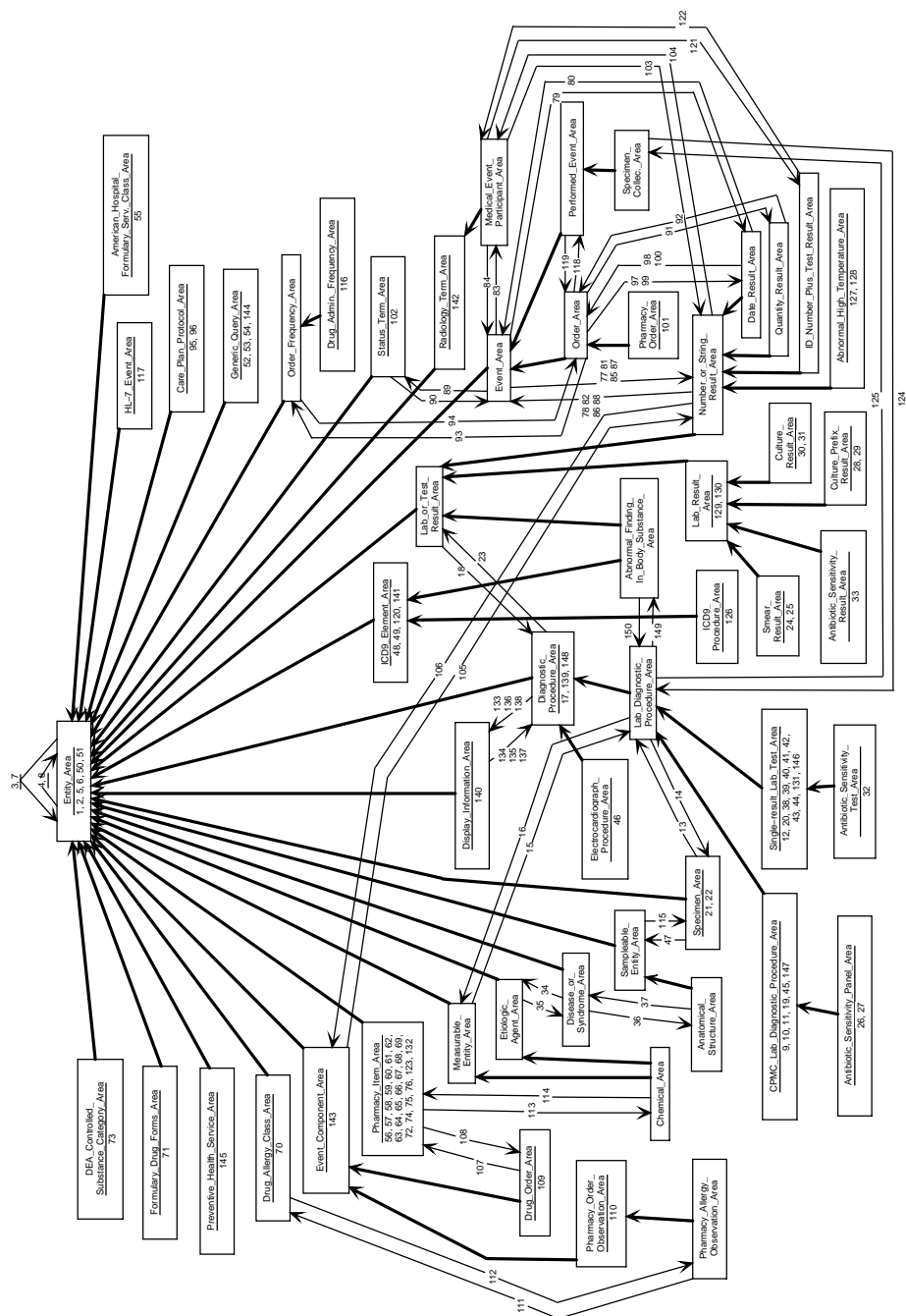


Figure 10. Property-introducing classes of MED-OOVR schema

|                                  |                                   |                                  |
|----------------------------------|-----------------------------------|----------------------------------|
| 1: umls_code                     | 2: name                           | 3: has_subconcept                |
| 4: has_superconcept              | 5: synonyms                       | 6: print_name                    |
| 7: has_part                      | 8: part_of                        | 9: cpmc_lab_proc_code            |
| 10: service_code                 | 11: cpmc_unit_names               | 12: cpmc_lab_test_names          |
| 13: specimen_of                  | 14: specimen                      | 15: measured_by                  |
| 16: substance_measured           | 17: units                         | 18: result_of_tests              |
| 19: cpmc_lab_proc_name           | 20: cpmc_lab_test_code            | 21: cpmc_lab_spec_code           |
| 22: cpmc_lab_spec_name           | 23: result_type                   | 24: cpmc_smear_code              |
| 25: cpmc_smear_name              | 26: cpmc_panel_code               | 27: cpmc_panel_name              |
| 28: cpmc_prefix_code             | 29: cpmc_prefix_name              | 30: cpmc_result_code             |
| 31: cpmc_result_name             | 32: cpmc_sensitivity_name         | 33: cpmc_sensitivity_result_name |
| 34: etiology                     | 35: causes_diseases               | 36: site                         |
| 37: site_of_diseases             | 38: normal_value                  | 39: low_normal_value             |
| 40: high_normal_value            | 41: male_low_normal_value         | 42: male_high_normal_value       |
| 43: female_low_normal_value      | 44: female_high_normal_value      | 45: normal_ranges_text           |
| 46: cpmc_ecg_name                | 47: substance_sampled             | 48: icd9_code                    |
| 49: icd9_entry_code              | 50: main_mesh                     | 51: supplementary_mesh           |
| 52: question_type                | 53: english_question              | 54: brs_question                 |
| 55: ahfs_class_code              | 56: dose_strength_units           | 57: dose_strength_number         |
| 58: formulary_name               | 59: short_formulary_name          | 60: formulary_code               |
| 61: drug_trade_name              | 62: drug_generic_name             | 63: drug_manufacturer            |
| 64: drug_rx_vs_otc               | 65: drug_form_code                | 66: drug_floor_stock             |
| 67: drug_route                   | 68: drug_in_formulary             | 69: drug_volume                  |
| 70: allergy_class_code           | 71: drug_description              | 72: drug_category                |
| 73: dea_code                     | 74: drug_specifier                | 75: drug_generic_code            |
| 76: drug_interaction_codes       | 77: event_id                      | 78: event_id_of                  |
| 79: event_date                   | 80: event_date_of                 | 81: event_patient_id             |
| 82: event_patient_id_of          | 83: event_participant             | 84: participant_of               |
| 85: event_organization           | 86: event_organization_of         | 87: event_location               |
| 88: event_location_of            | 89: event_status                  | 90: status_of                    |
| 91: order_quantity               | 92: order_quantity_of             | 93: order_frequency              |
| 94: order_frequency_of           | 95: protocol_name                 | 96: protocol_short_name          |
| 97: order_start_date             | 98: order_start_date_of           | 99: order_stop_date              |
| 100: order_stop_date_of          | 101: pharmacy_order_code          | 102: status_code                 |
| 103: participant_id              | 104: participant_id_of            | 105: order_value                 |
| 106: order_value_of              | 107: ordered_drug                 | 108: ordered_in                  |
| 109: drug_role_code              | 110: pharmacy_observation_code    | 111: observed_allergy            |
| 112: allergy_observed_in         | 113: pharmaceutical_component     | 114: pharmaceutical_component_of |
| 115: sampled_by                  | 116: admin_frequency_abbrev       | 117: hl7_event_code              |
| 118: event_object                | 119: object_of_event              | 120: old_icd9_code               |
| 121: participant_name            | 122: participant_name_of          | 123: drug_id                     |
| 124: collected_for               | 125: collected_by                 | 126: cpt4_code                   |
| 127: lower_limit_for_input       | 128: upper_limit_for_input        | 129: lab_message_code            |
| 130: lab_message_text            | 131: cpmc_long_test_name          | 132: drug_alert_code             |
| 133: has_default_displays        | 134: default_display_for          | 135: displays_elements_of        |
| 136: elements_displayed_by       | 137: has_display_parameters       | 138: is_display_parameter_of     |
| 139: has_test_display_class_name | 140: display_parameter_order      | 141: icd9_name                   |
| 142: cpmc_radiology_code         | 143: event_component_display_name | 144: query_filters               |
| 145: preventive_health_name      | 146: lab_alt_test_name            | 147: lab_alt_proc_name           |
| 148: has_proc_display_class_name | 149: defined_by_test              | 150: defines_abnormal_finding    |

Figure 11. Legend for properties of MED-OOVR schema in Figure 10

#### 4. Program for Generating the OODB Representation of a CV

We have used our methodology to transform two existing medical CVs, the InterMED and the MED, into object-oriented representations. The methodology can be applied not only to medical CVs but to any semantic network-based vocabulary, as long as it satisfies the “uniqueness of property introduction” rule discussed earlier. Both OODB representations, called, respectively, the InterMED-OOVR and the MED-OOVR, are currently up and running on top of the ONTOS DB/Explorer OODB management system. The creation of each was done automatically by a program called the *OOVR Generator*, which can be used to convert any source CV into its equivalent OODB form.

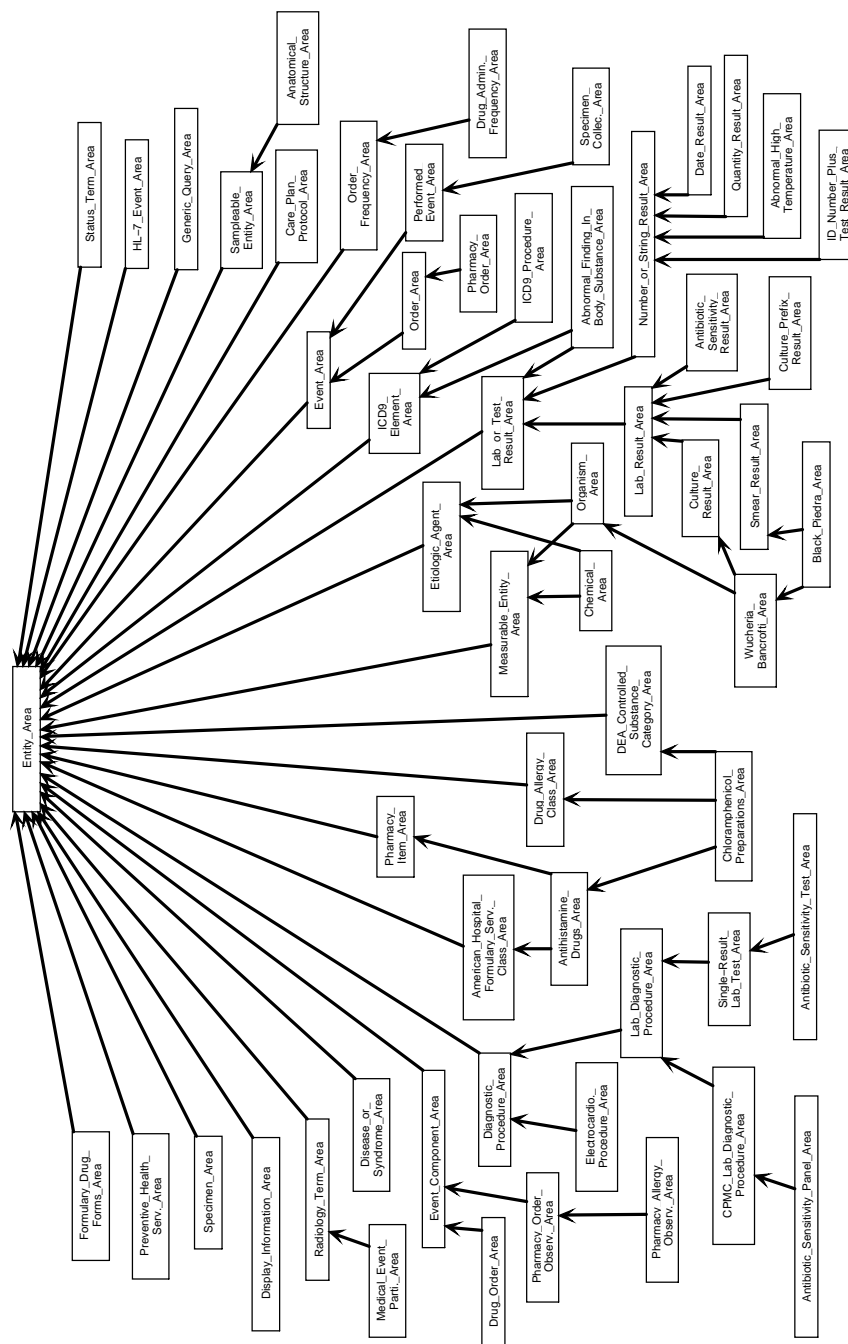


Figure 12. Property-introducing and six intersection classes of MED-OOVR schema

In this section, we describe the overall architecture of the OOV Generator. We will first present the assumed format of the source CV. We will then go on to discuss the components of the OOV Generator's functionality.

#### 4.1. Format of the Source CV

Various CVs can be assumed to be stored in different formats on disk. However, we will expect that a CV that is to be processed by our technique has a representation as a pair of text files, each having a specific syntax. If the desired source CV does not conform to this requirement, then it will first need to be converted. For this purpose, we include a *Preprocessor* module in the architecture of the OOV Generator (see Figure 13). This portion may need to be modified for different CVs. To illustrate the necessary format, we will be referring to the InterMED, from which it was originally gleaned. The MED also employs this representation.

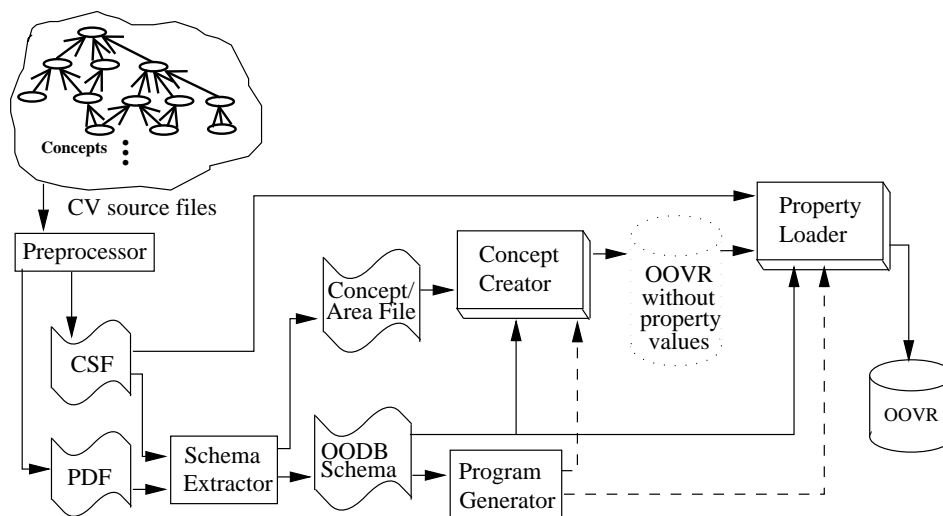


Figure 13. Architecture of the OOV Generator

The two text files making up the disk-resident format of a CV are referred to as the *Property Definition File (PDF)* and the *Concept Specification File (CSF)*. The PDF describes all the attributes and relationship types of the CV. Every attribute (or relationship type) is described by one line in the PDF. Each line is a triple whose first component is the property's number, which is assigned to a property in order to simplify references to it. The second component is the property's name; the third component is the number of the concept which introduces the property.

|                           |                                       |
|---------------------------|---------------------------------------|
| 1, umls-code, 1           | 1,1,“T071”                            |
| 2, name, 1                | 1,2,“Entity”                          |
| 3, descendant-of, 1       | 3,2,“Diagnostic Procedure”            |
| 4, is-a, 1                | 4,2,“Laboratory Diagnostic Procedure” |
| 5, synonyms, 1            | 28,2,“Finding”                        |
| 6, print-name, 1          | 35,2,“Chemical Viewed Structurally”   |
| 7, documentation, 1       | 37,2,“Inorganic Chemical”             |
| 8, snomed-code, 1         | 37,4,35                               |
| 9, has-result, 3          | 37,8,“C-10090”                        |
| 10, result-of, 28         | 37,14,7                               |
| 11, has-specimen, 4       | 38,1,“T106”                           |
| 12, specimen-of, 29       | 38,2,“Element or Elemental Ion”       |
| 13, substance-measured, 5 | 38,4,37                               |
| 14, measured-by, 30       | 39,1,“C0037473”                       |
| 15, has-precision, 5      | 39,2,“Sodium Ion”                     |
| (a) InterMED PDF          | (b) InterMED CSF                      |

Figure 14. Excerpts of InterMED source files

There are 51 lines in the InterMED’s PDF. Figure 14 (a) shows an excerpt of the file.<sup>3</sup> Note that the fields of a line are separated by commas. The MED’s PDF contains 150 lines.

The second file, the CSF, describes all the details of the concepts’ properties. Each line denotes the value for one property of some concept. A line in this file is also a triple. The first element is a concept number, uniquely identifying one of the concepts in the CV. The second number is a property number which stands for one of the relationship types or attributes and is therefore an index into the PDF. The third element may be another number (for a different concept) denoting the referent of a relationship. For an attribute, the third element is a primitive value, represented as a string type.

The InterMED’s CSF contains over 32,000 lines of text. Figure 14 (b) shows some of its entries. The line 37,2,“Inorganic Chemical” means that the concept 37 has the value “Inorganic Chemical” for the attribute *name* [property number 2 from Figure 14 (a)]. The entry 37,8,“C-10090” indicates that the SNOMED code of **Inorganic Chemical** is “C-10090.” The line 37,4,35 means that the concept 37, **Inorganic Chemical**, has the “IS-A” relationship (property number 4) to concept 35, **Chemical Viewed Structurally**. The MED’s CSF is quite a bit larger than that of the InterMED because the MED contains about nineteen times as many concepts having many more properties. In total, the MED’s CSF has around 1,000,000 lines.

#### 4.2. Architecture of the OOVR Generator

Figure 13 shows the overall architecture of the OOVR Generator. The OOVR Generator is a “second-order” process, with some of its modules being constructed by other modules during run-time. In the figure, we are using the following graphical conventions. A box represents a program module. A box with depth indicates that the module is generated by another module. The creation of such a module *A* by another module *B* is depicted by a dashed arrow from *B* to *A*. Ordinary arrows indicate the flow of data between modules either as files (wavy boxes) or databases (cylinders).

As we see from the figure, the OOVR Generator consists of five modules: *Preprocessor*, *Schema Extractor*, *Program Generator*, *Concept Creator*, and *Property Loader*. The Preprocessor is the only module which, as noted above, is CV-dependent. Hence, for a CV with a different file format from the InterMED’s, it would need to be modified.

The Schema Extractor is the first module to process the source CV. Its task is to carry out the area-partitioning and produce the appropriate OODB schema (as described in the previous section). It takes as its input both the PDF and the CSF. The output consists of two items, a “Concept/Area File” and the actual OODB schema for the OOVR. The Concept/Area File simply holds the mapping between the concepts of the CV and the areas derived from the partitioning process. Effectively, it is a two-column table, where the first column contains the concept names, and the second holds the associated area (class) names.

Presently, the OODB schema created by the Schema Extractor is specified in the DDL of the ONTOS system. Therefore, an OOVR will be an ONTOS database. We are in the process of updating the Schema Extractor such that it will build a schema specification in the portable Object Definition Language (ODL) proposed by the Object Database Management Group (ODMG) [5]. This would make our software independent of the back-end OODB system, which then could be any one that is ODMG-compliant.

The Program Generator, as its name suggests, just generates two architectural modules (as marked by dashed arrows in Figure 13): the Concept Creator and the Property Loader (drawn as boxes with depth). As we see, to do its work, the Program Generator requires the OODB schema produced by the Schema Extractor. Before describing why the Program Generator module is needed, let us discuss the details of the two modules that it generates.

The Concept Creator and the Property Loader together populate the OOVR. The Concept Creator first instantiates all concepts. That is, it creates one object in the OOVR for each concept in the source CV. The class of each object is determined by the Concept/Area File, which contains the concept-to-area mapping that we have described in the previous section. Note that the OOVR has all its concepts when the Concept Creator finishes, but none of those concepts has any property values (as indicated by the “phantom” OOVR having the dashed cylinder in the picture). This situation is rectified by the Property Loader which provides the concepts with

the values of all their attributes and relationships. It obtains these from the CSF that comes directly from the Preprocessor stage.

The reason that the process of populating the OOV is divided into two steps is because relationships are concept-to-concept (or object-to-object) references. In order to establish a relationship at a given object, the referenced objects must already exist. However, this may not be the case while the Concept Creator is carrying out its task. Therefore, it is necessary to defer the establishment of relationships until after all concepts have been created. So, in our architecture, Concept Creator first creates all the objects, and then Property Loader connects them via the appropriate relationships (and assigns their attribute values, as well).

The need for the Program Generator is dictated by the differences in structure that one finds among CVs. While we have set forth general vocabulary characteristics in Section 2, different CVs will certainly exhibit diverse properties and property introduction patterns. In other words, different CVs will have different OODB schemas! Both the Concept Creator and the Property Loader utilize the class definitions contained in the schema to perform their functions in the task of populating the OOV. The distinctions in the class definitions (e.g., the diversity of class names, numbers of properties, and so on) from CV to CV require that the declaration sections of both these modules be created anew for each source CV. Fortunately, the modules' overall forms have been captured as templates, and the process of generating them is automated. As mentioned above, this leaves only the Preprocessor open to changes for different CVs.

The complexity of the algorithm carried out by the OOV Generator is on the order of the total number of properties occurring within the CV. Specifically, let  $Attr$  be the set of all occurrences of attributes in the CV. Likewise, let  $Rel$  be the set of all occurrences of relationships. Then, the complexity is  $O(|Attr| + |Rel|)$ . We omit the details of the analysis for the sake of brevity. It will be noted that:

$$|Attr| + |Rel| = \sum_{x \in CV} |P(x)| = \text{Number of lines in the CSF.}$$

where  $|P(x)|$  is the number of properties of the concept  $x$ . Thus, the complexity is on the order of the CSF's size.

## 5. Conclusions

Controlled vocabularies (CVs) organize large groups of concepts into cogent bodies of knowledge which can be used to unify and integrate different, often independently created, applications with idiosyncratic terminologies. The semantic network has proven to be a viable modeling vehicle for CVs. However, designers, administrators, and users of CVs can be overwhelmed by the inherent complexity and size of typical vocabulary networks. In addition, semantic-network processing tools have not enjoyed significant commercial acceptance. This lack of availability is an obstacle to the wide-spread utilization of CVs.

In this paper, we have addressed these problems by presenting a methodology for converting a semantic network-based CV into the form of an object-oriented

database (OODB), called an Object-Oriented Vocabulary Repository (OOVR). The OODB schema resulting from this process is a valuable layer of abstraction. It offers a high-level view of the structure of the knowledge which can help the various classes of users in designing, managing, and utilizing a CV more effectively. In fact, in previous work we have described how the schema of the MED, a large CV from the medical domain, uncovered problems in the MED's original design. This eventually led to improvements in the MED's knowledge content. We have also constructed a Web-based, CV interface centered around the two view-levels of an OOVR (i.e., the schema level and the concept level) in order to provide enhanced access. At the moment, we are investigating additional, useful levels of abstraction that would enable a user to choose different amounts of detail.

In formally defining our methodology, we have utilized a variety of notions including that of property-introducing concept and the completely new notions of intersection concept, direct property-introducing descendant, and direct intersection descendant. Using these, it was possible to precisely describe how the CV is partitioned into areas, how object classes corresponding to these areas are defined, and which properties need to be assigned to these classes to capture the contents of the semantic-network input.

To complement the formal aspects of our methodology, we have built a program called the OOVR Generator which automatically carries out the methodology. We discussed the architectural details of this program. Interestingly, the OOVR Generator is a "second-order" process, with some of its modules being constructed by other modules during execution. Overall, the OOVR takes as its input a CV contained in two text files, called the Property Description File and the Concept Specification File, and produces a fully functioning OOVR. No human intervention in the form of database modeling or populating is required.

To demonstrate our methodology, we have applied it to the mid-sized InterMED and the very large MED, two CVs from the medical domain. Indeed, we have used the OOVR Generator to produce the OOVRs for both of these. They are currently up and running on top of ONTOS. The InterMED-OOVR is on the Web [43].

While our demonstrations focused on medical CVs, let us emphasize that our methodology is completely general. It works for any semantic-network CV in any subject area, as long as it is possible to map the CV into the PDF and CSF formats which we have described, and as long as the "uniqueness of property introduction" rule is satisfied. The OOVR Generator contains a module, called the Preprocessor, which performs any necessary mapping into the required file formats and which has to be rewritten for different semantic-network formalisms. Other than that, the OOVR Generator is completely general. In summary, our approach will help future CV builders to implement large, understandable, maintainable vocabularies, running on top of well supported database systems.

### Acknowledgments

We would like to thank Huanying Gu and Jim Cimino for their helpful discussions and insights. We also thank Boris Harmeyer for creating the MED-OOVR figures.

## Notes

1. Some typographical conventions: A bold face font will be used for concepts' terms. Properties of concepts will appear in italics and will be written strictly in lowercase letters. Object classes will also be written in italics. These class names, however, will start with uppercase letters.
2. There may be more than one such root. In that case, the choice is made arbitrarily.
3. The actual InterMED PDF contains some additional fields that are not relevant to the conversion process. These are removed by the Preprocessor.

## References

1. American Society of Hospital Pharmacists, Bethesda, MD. *American Hospital Formulary Service Drug Information*, 1997. Updated annually.
2. R. J. Brachman. On the epistemological status of semantic networks. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, Inc., New York, NY, 1979.
3. S. K. Card. User limits and the VDT computer interface (excerpt). In R. M. Baecker and W. A. S. Buxton, editors, *Readings in Human-Computer Interaction*, pages 180–191. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.
4. M. J. Carey, D. J. DeWitt, J. E. Richardson, and E. J. Shekita. Storage management for objects in EXODUS. In W. Kim and F. H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, pages 341–369. ACM Press, New York, NY, 1989.
5. R. G. G. Cattell and D. K. Barry, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.
6. P. P.-S. Chen. The Entity-Relationship Model: Toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.
7. J. J. Cimino. Personal communication, 1997.
8. J. J. Cimino, P. D. Clayton, G. Hripsak, and S. B. Johnson. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *JAMIA*, 1(1):35–50, 1994.
9. J. J. Cimino, G. Hripsak, S. B. Johnson, and P. D. Clayton. Designing an introspective, multipurpose, controlled medical vocabulary. In *Proc. Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 513–517, Washington, DC, Nov. 1989.
10. College of American Pathologists, Skokie, IL. *Systematized Nomenclature of Medicine*, second edition, 1982.
11. O. Deux et al. The O<sub>2</sub> system. *Commun. ACM*, 34(10):34–48, Oct. 1991.
12. D. H. Fischer. Consistency rules and triggers for thesauri. *Int. Classif.*, 18(4):212–225, 1991.
13. D. H. Fischer. Consistency rules and triggers for multilingual terminology. In *Proc. TKE'93, Terminology and Knowledge Engineering*, pages 333–342, 1993.
14. GemStone Systems, Inc. URL: <http://www.gemstone.com>.
15. C. A. Goble, A. J. Glowinski, W. A. Nolan, and A. L. Rector. A descriptive semantic formalism for medicine. In *Proc. 9th ICDE*, pages 624–631, Vienna, Austria, 1993.
16. H. Gu, J. Cimino, M. Halper, J. Geller, and Y. Perl. Utilizing OODB schema modeling for vocabulary management. In J. Cimino, editor, *Proc. 1996 AMIA Annual Fall Symposium*, pages 274–278, Washington, DC, Oct. 1996.
17. H. Gu, Y. Perl, J. Geller, M. Halper, J. Cimino, and M. Singh. Partitioning a vocabulary's IS-A hierarchy into trees. In D. R. Masys, editor, *Proc. 1997 AMIA Annual Fall Symposium*, pages 630–634, Nashville, TN, Oct. 1997.
18. M. Halper, R. Galnares, J. Geller, and Y. Perl. An analogical, Web-based interface to an OODB medical vocabulary. Submitted for journal publication, 1997.
19. M. Hammer and D. McLeod. Database description with SDM: A semantic database model. *ACM Trans. Database Syst.*, 6(3):351–386, 1981.
20. R. Hull and R. King. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv.*, 19(3):201–260, Sept. 1987.
21. N. Ide, J. L. Maitre, and J. Véronis. Outline of a model for lexical databases. *Information Processing and Management*, 29(2):159–186, 1993.

22. P. D. Karp, K. Myers, and T. Gruber. The generic frame protocol. In *Proc. IJCAI-95*, pages 768–774, Montreal, Canada, 1995.
23. P. D. Karp and S. M. Paley. Knowledge representation in the large. In *Proc. IJCAI-95*, pages 751–758, Montreal, Canada, 1995.
24. M. Kifer, W. Kim, and Y. Sagiv. Querying object-oriented databases. In *Proc. 1992 ACM SIGMOD Conference on Management of Data*, San Diego, CA, June 1992.
25. W. Kim and F. H. Lochovsky, editors. *Object-Oriented Concepts, Databases, and Applications*. ACM Press, New York, NY, 1989.
26. C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore database system. *Commun. ACM*, 34(10):50–63, Oct. 1991.
27. F. Lehmann. Semantic networks. In [28], pages 1–50.
28. F. Lehmann, editor. *Semantic Networks in Artificial Intelligence*. Pergamon Press, Tarrytown, NY, 1992.
29. D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley Publishing Co., Inc., Reading, MA, 1990.
30. L. Liu, M. Halper, H. Gu, J. Geller, and Y. Perl. Modeling a vocabulary in an object-oriented database. In K. Barker and M. T. Özsu, editors, *CIKM-96, Proc. 5th Int'l Conference on Information and Knowledge Management*, pages 179–188, Rockville, MD, Nov. 1996.
31. E. Mays, C. Apte, J. Griesmer, and J. Kastner. Experience with K-Rep: An object-centered knowledge representation language. In *Proc. IEEE AI Application Conference*, San Diego, CA, Mar. 1988.
32. G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psych. Rev.*, 63, 1956.
33. G. A. Miller. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41, 1995.
34. W. Möhr and L. Rostek. TEDI: An object-oriented terminology editor. In *Proc. TKE'93, Terminology and Knowledge Engineering*, pages 363–374, 1993.
35. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *TOIS*, 8(4):325–362, 1990.
36. J. Mylopoulos, V. Chaudhri, D. Plexousakis, A. Shrufi, and T. Topaloglou. Building knowledge base management systems: A progress report. Technical Report DKBS-TR-94-4, Department of Computer Science, University of Toronto, 1994.
37. The National Drug Code Directory. URL: <http://www.fda.gov/cder/ndc/index.htm>.
38. N. F. Noy and C. D. Hafner. The state of the art in ontology design: A survey and comparative review. *AI Magazine*, 18(3):53–74, Fall 1997.
39. Welcome to ODI. URL: <http://www.odi.com>.
40. D. E. Oliver, E. H. Shortliffe, and InterMed Collaboratory. Collaborative model development for vocabulary and guidelines. In J. Cimino, editor, *Proc. 1996 AMIA Annual Fall Symposium*, page 826, Washington, DC, Oct. 1996.
41. ONTOS Home Page. URL: <http://www.ontos.com>.
42. ONTOS, Inc. Lowell, MA. *ONTOS DB 3.1 Reference Manual*, 1995.
43. The OOVr Browser.  
URL: <http://object.njit.edu:2000/~newoohvr/JBI/INTERMED/InterTerms.html>.
44. Object Technology by Ardent Software (O<sub>2</sub> System).  
URL: <http://www.ardentsoftware.com/object/index.html>.
45. S. C. Shapiro and W. J. Rapaport. The SNePS family. In [28], pages 243–275.
46. E. H. Shortliffe, G. O. Barnett, J. Cimino, R. A. Greenes, S. M. Huff, and V. L. Patel. Collaborative medical informatics research using the Internet and the World Wide Web. In J. Cimino, editor, *Proc. 1996 AMIA Annual Fall Symposium*, pages 125–129, Washington, DC, Oct. 1996.
47. V. Soloviev. An overview of three commercial object-oriented database management systems: ONTOS, ObjectStore, and O<sub>2</sub>. *SIGMOD Record*, 21(1):93–104, Mar. 1992.
48. J. F. Sowa. *Principles of Semantic Networks, Explorations in the Representation of Knowledge*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991.
49. U. S. Department of Health and Human Services, National Institutes of Health, National Library of Medicine. *Unified Medical Language System*, 1996.

50. United States National Center for Health Statistics, Washington, DC. *International Classification of Diseases: Ninth Revision, with Clinical Modifications*, 1980.
51. P. Valduriez, S. Khoshafian, and G. Copeland. Implementation techniques of complex objects. In *Proc. VLDB '86*, pages 101–109, Kyoto, Japan, Aug. 1986.
52. Versant. URL: <http://www.versant.com>.
53. W. A. Woods. What's in a link: Foundations for semantic networks. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 218–241. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1985.
54. S. B. Zdonik and D. Maier, editors. *Readings in Object-Oriented Database Systems*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.
55. J. Zhang. Application of OODB and SGML techniques in text database: An electronic dictionary system. *SIGMOD Record*, 24(1):3–8, Mar. 1995.