Integrated Access to a Large Medical Literature Database

By Edward A. Fox, M. Prabhakar Koushik, Qi Fan Chen and Robert K. France

TR 91-15

Integrated Access to a Large Medical Literature Database

Edward A. Fox M. Prabhakar Koushik Qi Fan Chen
Robert K. France

May 12, 1991

Abstract

Project INCARD (INtegrated CARdiology Database) has adapted the CODER (COmposite Document Expert/effective/extended Retrieval) system and LEND (Large External Network object oriented Database) to provide integrated access to a large collection of bibliographic citations, a full text document in cardiology, and a large thesaurus of medical terms. CODER is a distributed expert-based information system that incorporates techniques from artificial intelligence, information retrieval, and human computer interaction to support effective access to information and knowledge bases. LEND is an object-oriented database which incorporates techniques from information retrieval and database systems to support complex objects, hypertext/hypermedia and semantic network operations efficiently with very large sets of data. LEND stores the CED lexicon, MeSH thesaurus, MEDLARS bibliographic records on cardiology, and the syllabus for the topic Abnormal Human Biology (Cardiology Section) taught at Columbia University. Together, CODER/LEND allow efficient and flexible access to all of this information while supporting rapid "intelligent" searching and hypertext-style browsing by both novice and expert users. This report gives statistics on the collections, illustrations of the system's use, and details on the overall architecture and design for Project INCARD.

0.1 CR Categories and Subject Descriptors:

C.2.4 [**Computer Systems Organization**]: Computer Communication Networks - Distributed Systems D.1.5 [**Software**]: Programming Techniques - Object-Oriented Programming E.2 [**Data**]: Data Storage Representations H.3.1 [**Information Systems**]: Information Storage and Retrieval - Content Analysis and Indexing H.3.2 [**Information Systems**]: Information Storage and Retrieval - Information Storage H.3.3 [**Information Systems**]: Information Storage and Retrieval - Information Search and Retrieval H.3.4 [**Information Systems**]: Information Storage and Retrieval - Systems and Software H.3.6 [**Information Systems**]: Information Storage and Retrieval - Library Automation - Large text archives H.5.1 [**Information Systems**]: Information Interfaces and Presentation - Multimedia Information Systems H.5.2 [**Information Systems**]: Information Interfaces and Presentation - User Interfaces I.2.1 [**Computing Methodologies**]: Artificial Intelligence - Applications and Expert Systems I.2.4 [**Computing Methodologies**]: Artificial Intelligence - Knowledge Representation Formalism and Methods I.7.2 [**Computing Methodologies**]: Text Processing - Document Preparation I.7.3 [**Computing Methodologies**]: Text Processing - Index Generation J.3 [**Computer Applications**]: Life and Medical Sciences

0.2 General Terms:

Design

0.3 Additional Keywords and Phrases:

blackboard, composite documents, frames, hypertext/hypermedia, interpretations, knowledge base, lexicon, Prolog, relations, thesaurus.

1 Introduction

The Integrated Academic Information Management System (IAIMS) initiative of the National Library of Medicine (NLM) was launched to integrate scholarly biomedical, clinical, and administrative information that exists in a variety of formats and access mechanisms [HAL86]. As part of this effort, we have focused on providing efficient, effective, and integrated access to scholarly information for users who may be geographically distributed. This paper describes the approach we have adopted for providing "seamless" integration of bibliographic, full-text, and thesaurus data. Building on our previous experience in applying techniques from artificial intelligence, information retrieval, and human computer interaction to the task of analyzing moderate sized collections of electronic mail digests [Fox87], and navy messages [Bar90], work has proceeded on adapting the CODER system to large collections of bibliographic citations on cardiology in MEDLARS, the MeSH thesaurus, and the course syllabus for Abnormal Human Biology (Cardiology Section) as taught at Columbia University. The distributed CODER system is ideally suited for our purpose, since the chief aim of the IAIMS effort is to make information available at multiple sites.

1.1 MEDLARS and MeSH

MEDLINE (MEDlars onLINE) provides access to a bibliographic database indexed and searchable by descriptors assigned by human indexers from MeSH (Medical Subject Headings). With more than eleven million indexed citations in the MEDLARS file, it is one of the most widely used databases in the world, second only to LEXIS [Hun86]. MeSH is the National Library of Medicine's controlled vocabulary used for indexing, cataloging, and retrieving information from the MEDLARS database. It is organized as a hierarchical subject thesaurus with approximately 15,000 main headings, 78,000 entry terms (references to main headings), and 50,000 names of chemical substances mapped to main headings. Humphrey [HM86] provides a more detailed discussion of MeSH and MEDLINE. There is approximately 300 MB of bibliographic information on cardiology in recent years of MEDLARS. The entire MeSH thesaurus is roughly 50 MB in size and contains a large number of headings and subheadings related to cardiology.

1.2 Physicians' Needs to Access Medical Literature

Medical professionals require ready access to the over 2 gigabytes of information published monthly in some 4500 journals [Hun86]. Covell et al. [CUM85] have reported that physicians require convenient access to knowledge while discharging their professional duties. It is thus desirable to build systems capable of analyzing a physician's information need and providing the desired responses without the need for a trained intermediary. Humphreys et al. [HL89], have also stated the need for direct user involvement in the development of the Unified Medical Language System (UMLS). However, owing to a very complex interface, the large volume of MEDLINE usage is the result of about 4000 professional medical librarians serving as trained intermediaries [Hun86]. While the manually indexed MEDLINE database can be accurately searched by a person well versed with the MeSH vocabulary, search uncertainty [CD87] is still very much a problem for the inexperienced user. It would thus seem that being able to exploit the precision of the MeSH thesaurus in conjunction with a vastly improved interface that can be quickly mastered by most physicians would be very valuable.

1.3 Our Approach

Our approach to serve the need for access to the medical literature is to provide and integrated environment with connections between the types of information physicians are familiar with and need. Thus, in addition to (a large number of) MEDLARS records and MeSH, we also obtained the course syllabus for Abnormal Human Biology, which has about 200 pages of information including a rich set of diagrams, references that relate to entries in the MEDLARS database, and descriptors and entry terms from the MeSH thesaurus. There is a strong interconnection among these diverse sources of information, so that various approaches to information access (e.g., starting with the syllabus, MeSH descriptors, a known citation, or just a natural language query – and then browsing) can all work in concert.

In this paper, we describe the approach we have taken to address the above issues with the INCARD system. First, we introduce the relevant literature. Second, we discuss the architecture of CODER and LEND. Next, we discuss document analysis, the advantages of adopting a structural description of documents and the work that has been done with the databases described above. Lastly, we give a detailed description of the INCARD system, describe its functions and show that the CODER/LEND approach is capable of working with diverse types of large information and knowledge bases.

2 Related Work

2.1 Document Analysis

While some have argued that full document processing is not feasible because of technology and economic considerations [JT84], nevertheless several research efforts have addressed this issue. The FRUMP system could skim newspaper stories by filling in sketchy scripts [DeJ82]. The TOPIC system builds a hierarchical structure of the document with the aid of a word expert parser [HR84]. In the IOTA system, the hierarchical structure of the text is exploited to provide for full-text indexing and retrieval [CD87]. The more recent SCISOR system also performs natural language analysis of documents [RJZ89]. However, parsing unrestrained text is difficult and most of these systems, but for SCISOR, work in constrained domains with relatively small collections of text. Gonnet et al. identified the advantages of processing documents with a well-defined structure in their work on the Oxford English Dictionary project [GW87]. The Maestro system supports structured documents, defined with SGML and addresses the issue of providing support tools [Mac90].

2.2 The Role of Thesauri

The international standard for monolingual thesaurus defines a thesaurus as "the vocabulary of a controlled indexing language, formally organized so that the a priori relationships between concepts are made explicit" [ISO86a]. The standard identifies three types of relationships between terms, namely, equivalence (synonymous), hierarchical, and associative. A term and its preferred term usually have an equivalence relation. Typically, this relationship is identified by the designation Use/Use For, while the corresponding MeSH designation is "See". Hierarchical relationships between terms capture the generality/specificity of terms. The common designation for this relationship is Broader Term/Narrower Term. The MeSH thesaurus only identifies the Broader Term relationship explicitly and this is designated as "See Under". In an associative relationship, terms are not equivalent and are not hierarchically related. The relationship includes but is not limited to, entities and their properties and processes, operations and their agents or instruments, actions and the product of the actions etc. [AG87]. Typically, this relationship is identified by the designation Related Term, while the corresponding MeSH designation is "See Related".

Thesauri are used by large online bibliographic databases to alleviate the term matching problem [CP88], [CD87]. Fidel [Fid87] has identified the disadvantages of using a printed thesaurus. Bertrand-Gastaldy et al. [BGD86] have noted the importance of being able to interactively navigate a thesaurus structure. Several researchers have attempted to provide access to a online thesaurus in order to integrate the thesaurus with the online searcher's tasks. An early version of CODER that operated with a collection of AIList Digest messages made use of a thesaurus generated from the Handbook of Artificial Intelligence to guide searchers [WF88]. The CANSEARCH system [Pol87] presented relevant portions of a thesaurus in the form of menus to aid in the process of query formulation. The EP-X system [SSGC89] makes use of hierarchically organized domain knowledge to assist users in defining their topics of interest. None of these systems have addressed the task of providing a user-friendly interface to thesauri.

2.3 Interface to Thesauri

A variety of interface styles have been in use for thesauri, the more common of which are, the alphabetical, the hierarchical, and the graphic [BGD86]. The most commonly used alphabetical display which lists all preferred and non-preferred terms in a single alphabetical sequence, with term-term relationships listed under each preferred term does not make the classificatory structure apparent. Graphic displays that portray the terms and their relationships provide 'contextual completeness' for each concept. Bertrand-Gastaldy [BGD86] cites a number of advantages to using a graphic display. However, as Lancaster [Lan72] has observed, representing the relationships between a large number of terms intelligibly is far from simple. The hierarchical display shows the full hierarchical structure at the entry point for a term, but does not display the complete definitional information as does the alphabetical display, nor does it show the complete hierarchy like a graphic display. Rada et al. [RL89] have noted that the large number of highly interconnected terms present in a thesaurus can benefit from a hypertext-style access mechanism, whose nodes and links can mirror the terms and their relationships to other terms. McMath et al. [MTR89] have reported an interesting hypertext style interface to ACM's "Computing Reviews Classification Structure" (CRCS) thesaurus. Their TraverseNet system displays the currently selected node surrounded by its offspring nodes in the center of a window. Users navigate by clicking on the offspring node of interest, which then becomes the central node surrounded by its offsprings. The system displays only hierarchical position for a term but no definitional information. While this interface may be suitable for a small thesaurus such as CRCS with about 1000 terms occurring in a strict hierarchy, we feel that it will present many of the same problems as the graphic display for a large thesaurus such as MeSH with over 100,000 terms occurring in a polyhierarchy.

2.4 Access to Full Text and Hypertext

The experience of researchers in the design of the Grolier's Electronic Encyclopedia [Ore87], the Symbolics Document Examiner [Wal87], the Hypertext version of the Oxford English Dictionary [RT88], and SuperBook [RGL87] has provided useful insight in our endeavor. Dillon et al. [DM90] conducted studies on what criteria are responsible for documents to make good or bad hypertexts. Their results indicate that individuals construe texts in terms of three broad attributes: *why* read them, *what* type of information they contain, and *how* they are read.

3 CODER

The CODER project was started in 1985 to serve as a testbed for exploring the value of techniques from artificial intelligence in improving the effectiveness of automatic systems for analysis and retrieval of complex documents [FF87]. It is an example of an distributed expert-based information system [BBB+87] that is organized so as to carry out many of the functions of a trained search intermediary or reference librarian. Although there are several investigations of a related nature, most notably I^3R [CT87], CODER is unique in its aim and scope. Like CODER, I^3R is based on a blackboard architecture that coordinates retrieval processing. However, I^3R is implemented as a monolithic system in Lisp with a small number of complex experts that access a statistically analyzed document collection. CODER was conceived as a standalone system for analysis, storage, and retrieval based on a collection of knowledge bases and a modular architecture that supports the manipulation of that knowledge. Figure 1 illustrates the distributed nature of the current version of CODER.

[FIGURE 1 about here: Distributed CODER System]

An initial version of CODER was constructed to operate with a collection of free-form electronic mail messages (AIList Digest collections from the Internet). The messages were analyzed to identify their type and structure, and frames were constructed as part of the resulting knowledge representation [WFCF89]. This work clearly identified the difficulty and domain dependent nature of the automatic techniques. It is much simpler for humans to help indicate how to interpret (e.g., tag) data when it is captured, or to record that analysis using a markup scheme. Hence we have chosen SGML as the basis for dealing with a collection of roughly 300,000 documents from MEDLARS and the full MeSH thesaurus. Document type definitions (DTDs) for both MEDLARS and MeSH have been derived, and the appropriate SGML-type markup has been added to the data. The automatic indexing system makes use of the DTDs and the SGML encoded files to build vectors, frames, and relations (i.e., knowledge representations useful for computer matching and other processing).

3.1 System Architecture

The system has three components, as illustrated in Figure 2. The CODER system may be regarded as two separate subsystems sharing a central "spine" of common resources. The analysis subsystem is responsible for analyzing and storing document contents, and the retrieval subsystem is responsible for matching user queries to relevant documents. The CODER spine stores the world and domain knowledge bases. The knowledge bases for INCARD consist of a lexicon derived from the Collins English Dictionary which serves as a English language knowledge base, the MeSH thesaurus and the MEDLARS document databases which handle knowledge about individual documents.

[FIGURE 2 about here: Architecture of the CODER System]

Each subsystem is implemented as a community of experts communicating through a central blackboard (repository for session specific communication between experts). Thus the blackboard in effect moderates any ongoing sessions. Each blackboard has an attached strategist which carries out the main planning and control operations for the session. The strategist initiates the participation of each expert in the community through a set of heuristic rules based on a model of the expert's area of competence. Experts are specialists in restricted domains that are relevant to the task being addressed and are implemented relatively independent of each other. The analysis subsystem accepts a set of documents encoded in SGML, together with a description of the document structure. Document terms are identified and are stored in a central dictionary along with weights based on document frequency, after undergoing morphological analysis. The document knowledge bases also include weights based on frequency of occurrence of the terms in the documents and are used by the retrieval subsystem for screening matches. The retrieval subsystem uses these knowledge bases to match documents or portions of documents to a user's information need. Note that the retrieval subsystem is decomposed into modules along functional lines, based on studies of user and search intermediary interaction [BSW83].

The current version of the system runs as a collection of processes on a variety of networked UNIX machines making use of TCP/IP for interprocess communication (recall Figure 1). A mixed initiative interface has been developed using the Macintosh toolbox and is accessed under A/UX, a version of the UNIX operating system. Work is underway to develop an interface with the Open Software Foundation's Motif graphical user interface under the X Window System for better portability.

3.2 F^3L

As explained earlier, CODER runs as a collection of processes possibly on several networked UNIX machines. In order to facilitate the exchange of information between processes, we have developed a high-level knowledge representation language, F^3L, which supports a wide variety of both primitive and constructed data types. It is similar in spirit to the Abstract Syntax Notation ASN.1 defined in ISO 8824, but is specifically geared towards the needs of the information retrieval community. F^3L, which stands for Facts/Frames/Functions Language is built on top of the standard primitive data types of integers, real numbers, strings, and atoms. It also provides constructors to form simple structured types such as sets, lists, and tuples (vectors), as well as more complex structured types such as frames, facts, and functions. F^3L is a strongly typed language. Each F^3L expression has a unique type that determines exactly what operations can and cannot be applied to it.

F^3L is intended to be a knowledge representation language for interchange of information. As such, it has no control constructs, nor any representation of an operational procedure. Any such statement must be supplied by the embedding language. F^3L, for its part, provides advanced knowledge structures not available in the language, together with a canonical way of creating and manipulating those structures that can transfer easily from language to language. Every F^3L type, or class, has a unique standard representation used to communicate among different machine architectures and different computer languages. Since the implementation of F^3L varies from language to language, this standard representation must be converted to and from constructs in the embedding language.

3.3 Document Analysis

As stated earlier, CODER was designed as a standalone system capable of both document analysis and retrieval. The analysis subsystem of CODER has been designed to extract knowledge automatically from documents encoded with SGML markup. The Standard Generalized Markup Language (SGML), described in [ISO86b], provides a mechanism for describing classes of structured documents with a Document Type Definition (DTD), and for coding individual instances belonging to a particular class. A significant advantage to using a grammar-based representation such as SGML for a document is its ability to perform multiple applications on the same source file, e.g., analyzing the encoded text, formatting the text for display, providing interdocument linkages (with the attribute facility in SGML), etc.

The CODER system derives much of its power from its rich knowledge representation capabilities as stated earlier. It is important that the knowledge representation scheme chosen be able to model at the very least: (1) knowledge about the logical structure (collection of elements in SGML parlance) of the document, (2) the semantic restrictions on the contents of the elements, and (3) knowledge about the relationships among documents. modeling documents in a content-appropriate fashion facilitates the understanding of user queries and retrieval of appropriate documents in a manner similar to that of an expert librarian. The CODER system makes use of frames for representing documents since frames model objects with typed fields. The hierarchical structure of a document is easily modeled by having lower level frames as slot fillers for the frames representing the higher levels of the document structure hierarchy.

4 LEND

There is significant benefit to providing an integrated representation of data, information, and knowledge (document collection and lexical) that exists in the CODER system. The highly inter related nature of lexical information suggests a semantic network type of representation, similar to SNePS [Sha79]. The complex structure of documents, e.g., MEDLARS citations suggests using frames as a representation schema. Thus for information retrieval purposes, we need to be able to store and manipulate both frames and relations in an efficient and flexible manner. Our approach has been to develop an object-oriented database system capable of modeling both frames and relations as objects.

LEND is an object-oriented database system supporting abstract data types, inheritance, complex objects, semantic relations, and set-oriented and navigational access to objects. Its target applications include traditional information retrieval models, hypertexts, and SNePS style semantic networks. Based on an object-oriented model and engineered in an object-oriented fashion (entirely in C++), LEND is highly extensible for new dimensions of applications and for the enhancement of system performance. This has been made possible by our work on minimal perfect hash functions (MPHF) [FHCD90] and order preserving minimal perfect hash functions (OPMPHF) [FCDH90], which can find these functions for large key sets very rapidly and allow retrieval of any object stored in the network with a constant number of disk accesses. LEND has also been extended to handle hypertext/hypermedia and F^3L objects [Che90]. LEND has been under development since 1988 and a first version was made operational in early 1990 with approximately

70 MB of data from the CED (Collins English Dictionary) and the AIList news digest. The two databases showed exceptional retrieval performance as reported in [Che90].

4.1 LEND Architecture

The architecture of LEND is typical of other existing object-oriented database systems such as ORION [KGBW90], IRIS [WLH90], POSTGRES [SRH90] and GEMSTONE. Figure 3 shows the three layered architecture of LEND. The lowest or storage layer supports storage and retrieval of objects on disk or main memory. The intermediate or object layer provides facilities for supporting high-level structural abstractions such as generalization/ specialization, classification, etc. The topmost layer is essentially the application layer which provides user-friendly query languages for accessing the data stored in LEND.

[FIGURE 3 about here: LEND System Architecture]

The storage layer provides storage and access facilities to LEND objects. The support for both primary memory and disk resident data in LEND is handled by a memory database storage manager (MDSM) and a disk database storage manager (DDSM). The MDSM is responsible for the loading of objects from disk into main memory, construction and destruction of objects, and retrieval of objects based on their object identifiers or components. The DDSM performs essentially the same functions for the persistent objects. In addition, it also handles the placement of objects on various I/O devices and manages cache and buffers. The DDSM uses the notion of a virtual device class (an infinite sequence of bytes accessible by specifying an offset and length) to model any external storage device. The message syntax is uniform for both managers so that one can switch between the two managers as desired. This layer is coded in C++ as a set of classes to ensure generality, extensibility, and high performance. Efficiency of access is achieved by making use of AVL trees for primary memory resident objects and MPHFs and OPMPHFs for disk resident objects. Thus it is possible to load a disk resident object into primary memory in one disk access.

The object layer contains the basic classes present in the LEND model [Che90], [FCF91] as shown in Figure 4. The user defined classes are also present in this layer. A run-time support manager processes queries.

[FIGURE 4 about here: LEND Class Hierarchies]

The object class which is the root of the hierarchy shown in Figure 4a contains methods for construction/destruction of objects, reading/writing of objects to/from the disk, comparison of object components, selection of specific components of objects, etc. The object class also provides the representation of the object ID as a <class_code, instance_code> pair. Methods for this and other classes are shown in Table 1.

A composite object is composed of other (component) objects, which can either be other LEND objects or vanilla C++ objects without a LEND object ID. This facility permits the manipulation of a hierarchy of objects as a single logical entity. For a LEND component object, the 'composed-of' relation can be represented either by using its object ID directly [KC90] along with some attribute to denote the type of the relation, or making use of the LEND binary relation object. In the latter approach, we include the LEND binary relation object ID in the composite object and encode the semantics of the 'composed-of' relation inside the relation object. This approach provides more flexibility.

A collection object is a group of other objects. Sets and bags are examples of collection objects. Collection objects can be created with a number of options such as persistent/transient, primary memory/disk based, etc. The collective object, most notably the set object, tends to be very useful for information retrieval applications.

Parts b, c, and d of Figure 4 illustrate other class hierarchies implemented in LEND. The storage of objects, illustrated earlier in Figure 3, leads to classes as shown in Figure 4b. Indexes are illustrated in Figure 4c, and the supporting classes of perfect hash functions are given in Figure 4d.

Table 1: LEND Classes and Methods

CLASS NAME	METHODS	COMMENTS
object	int convertToString(bstring, int)	convert to byte string
	int convertFromString(bstring)	convert back from string
	int length()	return byte string length
	void validate()	validate the object
	void invalidate()	trash the object
	int OK()	is the object in good shape ?
	object select(int)	select a component
	int LE(object, int)	compare less
	int EQ(object, int)	compare equal
	istream read(istream)	read from input stream
	ostream write(ostream)	write to output stream
collection_object	char* myNickName()	return nickname
	int load(char*)	load objects into collection
	PixAVLSet get_pixes(q_object)	get locations for all objects that match with the query object q_object
	Pix add(object)	add object to collection
	int del(Pix)	delete object at loc. Pix
	int add_index(char*)	add new index to coll.
	void disable_index(int)	discard an existing index
	Pix first()	iteration fns over objects
	object operator()(Pix)	
	void next(Pix)	
	bstring operator[](Pix)	get the byte string version of the object
	int length()	return number of objects in the collection
	void statistcs()	return the statistics of the collection
	void to_persistent()	change to persistent
	void to_transient()	change to transient

Table 1: LEND Classes and Methods – continued

CLASS NAME	METHODS	COMMENTS
mphf	int load() int hashTo(key) int build(const char*) int del()	load the MPHF compute hash loc. for key build MPHF from key set delete this MPHF
opmphf	int load() int hashTo(char*, int) PixAVLSet rangeSearch(key1,key2)	load the OPMPHF compute hash location get Pixes for all keys lexically between keys key1 and key2.
index	Pix first() PixAVLSet operator()(Pix) void next(Pix)	iteration functions
	int ready()	get index object ready
	insert_pix(object, Pix)	add object and its Pix
	delete_pix(object, Pix)	delete object from index
	indexing()	perform actual index task, if indexing must be done in batch style (i.e., MPHF indexing)
	PixAVLSet retr(object)	retreval functions
	void kill()	kill functions
	void to_persistent() void to_transient()	status change functions

Table 1: LEND Classes and Methods – continued

CLASS NAME	METHODS	COMMENTS
walk	walk converse(walk)	reverse the walk
	walk concat(walk)	concat this walk with the argument walk
walks	Pix first(int) object get_object(int, Pix) void next(int, Pix)	iteration functions - over source/sink objects
	void set_hint()	path traversal evaluation hint
	Pix first() void next(Pix) walk operator()(Pix)	iteration function - over individual walk
	walks concat(walks)	walk join
	walks and(walks)	return a walk if all walks in the set share the same source and sink object
	walks or(walks)	return argument walk set
	walks TC(int)	Constrained transitive closure of this walk set. int specifies the depth of the traversal.
	walks kstar(objectSet, walks, int)	Constrained transitive closure starting at the objects in the objecSet
	walks kstar(objectSet, objectSet,int)	Constrained transitive closure. The starting objects are in the first objecSet. The destination objects are specified in the second objectSet.

The application layer provides the interface with applications. The top portion of Figure 3 shows four such applications currently being implemented, with a good part of the work complete on the rightmost three. The LEND query language interface allows interactive access to LEND through a user-friendly query language [FCF91]. The F^3L interface allows migration of LEND objects to other CODER modules possibly residing on different machines through the high-level constructs provided in the F^3L language. The C++ interface is the simplest and C++ programs can merely include the LEND class declaration files and bind the LEND class binary files into their executable images. The SNePS interface will allow SNePS to perform path-based inference on disk-resident semantic networks.

5 CED

Our involvement with the *Collins English Dictionary* began in 1985 with a typesetter's tape generously donated by Collins Publishing to the Oxford Text Archive. It took more than a year of tedious work to construct an analysis system that in seven stages translated the stream of text interspersed with non-ASCII typesetting codes into sets of information objects and relations [Woh86]. Subsequent years have refined this information, identifying and where possible repairing classes of errors and converting it from its first form as sets of Prolog facts into classes of texts, frames, and relations that can be loaded into LEND. Certain categories of errors have proven difficult to fix, causing data to be discarded from the collection. Other difficult to detect errors have caused us to spend a large amount of time checking and manually editing the files, and forced repeated reloading of portions of the data into various versions of LEND. The nature of the analysis process, though, and particularly the monolithic character of the analysis system, have thus far discouraged anyone from repeating the analysis.

Despite the problems with the data, the collection has proved an invaluable aid to research; as a tool for natural language understanding, as a basis against which to test hypotheses about lexicons, information retrieval and large knowledge bases, and as an inspiration to new directions of research. Information about the forms of English words, particularly irregular forms and derived variants, is used to drive the morphological analysis and word-recognition software used in the last several CODER systems, including INCARD. Word definitions and sample uses from the CED are displayed to CODER users as part of browsing and query construction, aiding the formation of richer and more precise queries. Problems of storage and retrieval of the highly connected net formed by CED senses (see Figure 5) provided a major motivation for the research that culminated in LEND. And research is still progressing on use of lexical relations in automatic query construction. Thus, the CED lexicon remains a robust and important part of our research environment.

[FIGURE 5 about here: CED Entities and Relationships]

While statistics have been reported for the early Prolog version of CED [FWS+86], none have been given for the LEND version. Table 2 provides some illustrative information. Table 2a gives sizes or counts for the various types of entities. Table 2b gives in and out-degree values for some of the node types present in the LEND storage implementation of CED. This data can be used to help in optimizing access, and further data can be obtained to help better understand word usage in dictionaries.

Table 2: CED Statistics

a) Object Occurrences

Object	No. objects on disk	No. objects in memory
sense	354254	5174
subSuper	33	0
variant	48845	0
cedSampleText	20626	0
classDesc	76	0
hasCategory	27583	0
	28492	0
hasPos	98028	0
	100565	0
lemEntry	82862	0
occursInCedDef	973023	0

b) Distribution of Links

Relation	Total No.	Nodes with Rel.	Mean	Std.Dev	Min	Max
In degree	934688	48792	19.16	106.02	1	6485
Out degree	934688	77838	12.01	14.85	1	422

6 MEDLARS Documents

Most retrieval systems are limited in the amount of information content that is actually represented. Based on prior experience [WFCF89], we believe that users can perform more effective retrieval by using the hierarchical organization of documents as well as concepts such as names, dates, and citations. Thus our approach is oriented towards deriving knowledge structures from documents rather than the traditional keyword indexing. We discuss here our experience with analyzing the cardiology subset of the MEDLARS citations, and the resulting knowledge representation.

6.1 MEDLARS Document Analysis

The MEDLARS document collection is typical of current commercial collections in that it contains a large amount of information that is not used for retrieval. One goal of our analysis efforts was to reveal this implicit information in such a way that it could be effectively used during search, browsing, and presentation. To this end we examined both the published definitions of MEDLARS records and the records themselves, encoding both what information we found and the format in which it was presented. The resultant document type definition (DTD) was then used to construct both a translator from the initial MEDLARS corpus into an SGML format, and a parser that operated on the SGML corpus to produce databases of information objects.

This two-step process has proved far more satisfactory than the monolithic process used to analyze the CED. When data was discovered that did not fit the DTD, we have been able to examine the SGML file, find the problematic data, and decide where in the process to cope with it. Since the SGML files are composed exclusively of printable characters, they can be examined and split using standard UNIX tools, and if necessary edited by hand. Since SGML tags reflect the content of the fields they identify, and since the tags used for this corpus have easily read symbolic names, no time is lost in identifying where the parse or translation has failed. Finally, since the translator and parser use different technologies to achieve their different tasks, data exceptions, variations, and errors can be dealt with at either of the stages, using whichever techniques prove most efficacious.

Different corpora will no doubt require different translation techniques. In the case of MEDLARS, translation to SGML is accomplished by a hand-crafted C program. Corpora using similar representation conventions may elicit more standardized programs or program toolkits. As SGML becomes more standard as a data interchange language, this stage may become either automatic – translating from one system of tags and attributes to another – or completely unnecessary.

The document parser developed for INCARD, in contrast, can be re-used for other document collections. Three factors contribute to this. First, the SGML input means that field and token recognition have been standardized. Second, the parser is constructed in a regular fashion out of standardized parsing and data structuring components. Third, the parser itself is built using high-level tools that permit easy re-structuring of the document grammar. The first factor should be clear from the discussion above. The second will be discussed below. The surprise about the third is that the tools are LEX and YACC, the UNIX standard compiler generators. LEX and YACC are notably versatile tools, and have found many applications outside compiler construction. They have even been used to analyze natural language texts, although we have found them difficult to use for this purpose. For the analysis of structured documents into their components, or repetitive data into its elements, however, we have found them very effective. As a bonus, it is exceptionally easy to translate an abstract DTD into an abstract grammar, and thence into a YACC grammar. We have hopes that as we practice this approach on further corpora it will be possible to further automate this process, until a parser can be generated automatically from the DTD and a set of class-based methods.

Analyzing the MEDLARS corpus has been streamlined by exploiting the class structure of the information involved. Although each record in the corpus has a complex internal structure, the primitive objects are drawn from a relatively limited set of classes: various sorts of numbers, strings, and text objects. Each of these classes has been paired with a parsing function designed to recover class instances from their representations in the corpus. The most complex of these is, of course, the function for mapping text objects into relationships between the components of the text (words, names, numbers and so forth) and the document field in which the text occurs. Less complex document components like chemical registry numbers or page spans are paired with parsing functions for discovering, respectively, contiguous strings of letters, numbers, and hyphens and integer interval representations within character strings. Finally, regularities in the information objects produced – frame-based descriptions, linking relationships, and text object vectors – are exploited to direct the synthesizing actions of the parser.

6.2 MEDLARS Representation

As an example, consider the representation of running text by text object matrices. Text object matrices are a generalization of the word vectors used in several classical text retrieval systems. If a piece of text is abstracted as a flat sequence of words, it is natural to represent it by the set of all unique words in the collection, each weighted by the number of times it occurs in the text. For a text collection with W words, one way to consider this representation is as a W-dimensional vector, most of the components of which will be zero for any given text. Taking the vectors as rows of a matrix, each of which describes a text, we note that the columns of the matrix describe the usage of a given word. Texts in INCARD are represented by matrices of information objects, including word roots, word senses, numbers, ranges, and of course, unclassified strings. The objects are typed, so that retrieval can be conditioned by the operations appropriate to the different classes. The matrix organization makes it equally easy to retrieve all documents with a given text object, or linear combination of text objects, and all text objects that occur within a given document. Since the matrices produced are extremely sparse, only the non-zero cells are actually stored, defining a database of "X occurs in T" relation instances.

The system constructs such a matrix using a fixed process: once a piece of running text is isolated within a structured document, it is reduced to a sequence of lexical tokens. The tokens are then scanned linearly to identify higher-level text objects. Digit strings, for instance, trigger a set of rules that check surrounding tokens for the punctuation and other strings used to represent integers, real numbers, fractions, percentages, and number spans. Capitalized lower case strings following a period are treated differently from those following another letter string token. Other rules govern treatment of hyphenated words, possessives, and words broken across lines. In all cases, letter strings are checked against the CED root and variant string classes using a morphological analyzer that checks up to 24 different transformations (more were not deemed necessary, as infrequent transformations are listed explicitly in the CED). As each string is identified with an English word, the ID of that word is inserted into a hash table, together with the position of the word in the text. When the text has been fully processed, the IDs for every unique object in the text, each with its total frequency and the set of its places in the text where it occurs, are output, and the process begun again. As each text object is resolved it is also cached in a memory-resident dynamic hash table, so that the morphological transformations and database accesses do not need to be repeated.

This process is performed on each text field in the MEDLARS collection: document title, both in English and (where applicable) in the original language of the document cited, document abstract, indexers comments, and (for the unique journals and institutions mentioned in the collection) journal title and institution name. The exact process is also used on natural language queries submitted to the system, has been used on the definition and sample texts of the CED, and with additions and refinements, will be used in future CODER research.

The result of the analysis is a LEND representation, similar to that for CED (recall Figure 5), which is shown in Figure 6. The relations shown to boxes near the periphery reflect connections to real-world entities like journals, authors, and chemical substances. The relations to descriptors are of particular importance for access through the MeSH thesaurus, as discussed in Section 7.

[FIGURE 6 about here: MEDLARS Representation]

6.3 An Example of MEDLARS Document Analysis

It was evident in looking at the MEDLARS document collection that there is a large amount of implicit (structural) information that could be gainfully employed for effective analysis, search and presentation of documents to the user. Figure 7 shows an example citation in MEDLARS (somewhat simplified for the sake of illustration) in its original form (Figure 7a) and after being processed by the document formatter (Figure 7b). The document formatter converts the source text into a document conforming to SGML syntax, and splits some of the entity descriptions into their constituent parts. A LEX tokenizer picks up tokens from the SGML encoded document and hands it to a YACC parser which has knowledge of the MEDLARS DTD. Frames are then constructed for low level elements such as dates, authors, and journals. A higher level frame is also constructed for the entire citation. The searchable data elements are vectorized by consulting the full word list from the CED. Figure 7c illustrates the final representation in CODER/LEND.

[FIGURE 7 about here: MEDLARS Document Analysis Example]

7 The MeSH Thesaurus

As stated earlier, the MeSH thesaurus has been stored in LEND. We have designed separate classes for each of the major categories of MeSH terms, i.e., vocabulary terms comprising of the major and minor descriptors, qualifier terms and chemical terms. The term-term relations are represented by appropriate relations (links) between instances of these objects.

7.1 Representation

Figure 8 illustrates the representation used for MeSH. It is similar to that for CED (recall Figure 5) and MEDLARS (recall Figure 6). However, as a thesaurus it contains several types of cross-references (i.e., see relations), hierarchical relations (i.e., tree numbers and both sub and super links), and qualifying relations (i.e., allow qualifier). There are also links for pharmaceutical actions resulting from chemicals.

[FIGURE 8 about here: MeSH Representation]

Note that the representation for INCARD is actually an integrated one, not separated into CED, MEDLARS, and MeSH components. This is illustrated in Figure 9.

[FIGURE 9 about here: Integrated INCARD Representation]

Table 3 provides statistics regarding the MeSH data as stored in LEND.

Table 3a gives counts for the various types of nodes and relations. Table 3b gives details on the relations, illustrating distributions for the numbers of links.

Table 3: MeSH Statistics

a) Object Occurrences

Object	Occurrences	Comments
Major Descriptor	12560	Subject Heading (SH) in Index Medicus
Minor Descriptor	3287	Not a SH, refers to a Major Desc.
Qualifier	717	Modifies the scope of a Descriptor term
Chemical Term	32618	CAS Registry numbers and links to Desc.
See CX	10355	Entry term to Descriptor
See Under	3015	Minor to Major Descriptor
See Related (Backward)	1715	Minor/Major Descriptor to Major
See Related (Forward)	1818	Major to Major Descriptor

b) Distribution of Links

Relation	Occurrences	Desc.with Rel.	Mean	Std.Dev	Min	Max
See	10355	5952	1.74	1.22	1	15
See Under	3015	1586	1.9	1.7	1	19
See Related (bwd)	1715	1393	1.23	0.58	1	7
See Related (fwd)	1818	1237	1.47	1.1	1	19
Tree Number	27039	15831	1.7	1.03	1	10

If one views the MeSH data in terms of these nodes and links, the pattern language facility in LEND allows associative access to objects related by links, e.g., finding the position of a term within a hierarchy. However, a direct manipulation interface obviates the need for a user to learn the pattern language in order to browse the MeSH thesaurus.

7.2 Interface to Thesauri

In designing the interface to the MeSH thesaurus, we have benefited from the findings of several researchers, most notably the work reported in [Gon87], [RT88], and [MTR89] (recall Section 2.3). McMath et al. [MTR89] provided access to MeSH in an extended version of their TraverseNet system and recorded user's reaction to their system. They have reported that the inability to retrieve the neighborhood of a term directly caused great disorientation for the users, chiefly due to MeSH being very broad and deep. Thus in addition to allowing browsing through an index node that contains an alphabetical list of preferred and non-preferred terms, we have also provided a query interface for experienced users who wish to enter terms directly. Users can browse the thesaurus by specifying the term of interest and then navigating in its vicinity. Figure 10 shows the user entering a MeSH term. The response depends on the type of the term entered. A major or minor descriptor will be displayed with full definitional and hierarchical information as shown in Figure 11. Qualifier and chemical terms will have only definitional information displayed. The items listed in the relation areas (See, See Under, and See Related) are link anchors to nodes that represent the corresponding items.

[FIGURE 10 about here: MeSH Query Window]

[FIGURE 11 about here: MeSH Descriptor Information Window]

Users can continue browsing the thesaurus by clicking on the link anchors. If the user enters an incorrect term, then the closest matching terms are displayed so that he/she can pick the correct one. Users can also copy any MeSH term and use it as a query term in the MEDLARS query window. We are currently working towards providing a 'click-and-go' style interface to retrieve MEDLARS records by selecting terms of interest in the MeSH term description windows.

8 Cardiology Textbook

Frisse [Fri88], in his work on the Dynamic Medical Handbook Project has identified the need for research regarding the delivery of large-scale hypertext systems to medical settings. In our work with the cardiology course notes, we have tried to identify the desiderata for converting printed full-text documents with realistic amounts of text and graphics into a hypertext. The chapter on cardiology in the course notes on Abnormal Human Biology has been written to provide students with a systematic approach to the evaluation of patients with heart and lung disease based on an understanding of physiologic concepts [Dru90]. It is about 200 pages in length with a rich set of illustrative diagrams and citations that can be located in MEDLARS. Thus in dealing with the course notes, we found it necessary to have at least some familiarity with both the document contents and its users.

In analyzing the cardiology chapter for structure we were able to identify thirteen broad topics each with several sections and subsections to them and seven case histories (protocols) of people with cardiac disorders. Each topic was found to have numerous implicit cross references to other topics. The text of each topic was also found to have some structure containing either all or some of the following parts: Objective, Prerequisite knowledge, Topic Discussion, Glossary of Terms, Tables, Figures, and MEDLARS References. The text of the document is roughly about 360 KB in size, with the approximately 90 figures and tables contributing another 1 MB. The text of the document is loaded into LEND treating each topic as a composite LEND object. This allows us to retrieve an entire topic, (excluding the diagrams and tables) in one disk access. The design of the user interface has been guided largely by the experience of other researchers cited above. Users can browse the document using the table of contents, or search the document using keywords, and follow links among topics, to the MeSH Thesaurus and the MEDLARS citations. This part of the system is currently under construction, but we envisage our interface as shown in Figure 12.

[FIGURE 12 about here: Cardiology Course Notes Window]

The user arrives at this window either by choosing the Cardiology textbook from the Browse menu in which case the window will have a table of contents displayed or by opting to search the Cardiology textbook from the search menu. The latter will display the appropriate topic in the window. In either case, the user can continue browsing the textbook. Paragraphs of interest can be marked for visiting later by clicking on the bookmark button, which provides the option of either setting a new bookmark or visiting a previous one. The user can also make notes while browsing the textbook.

We regard this as a particularly valuable facility for the audience we have in mind. Clicking on the Figure button displays a list of the figures and tables in the topic currently being displayed and the user can select the one of interest. The Next and Previous buttons allow sequential browsing of the topics in the Cardiology textbook.

9 Conclusions

We have demonstrated that the CODER/LEND approach is capable of providing integrated access to diverse kinds of information, i.e., dictionary entries, bibliographic records, thesauri, and a full-text document. Throughout the design and development of the system, we have followed the overall IAIMS goal: to integrate rather than innovate. Although the full-text document is fairly small and is still in the "proof of concept" stage, we feel that the approach we have adopted can easily scale to larger documents. Clearly, the size of the dictionary and thesaurus, as well as the number of bibliographic entries, prove that CODER/LEND can be used to carry out an in-depth analysis and detailed representation of large text and knowledge bases.

10 Acknowledgments

Our work on Project INCARD was funded in part by Columbia University, in connection with their IAIMS project. We are grateful for the data provided, and for the numerous discussions and guidance given, especially by Nancy Roderer and Anthony Aguirre.

The VPI&SU Computing Center has assisted with staff help, data storage, and other computer related support. Students in several courses or doing M.S. and Ph.D. work in the Department of Computer Science, including Ajit Naidu, and Rajesh ?, have also worked on phases of the project. Secretarial support has been provided by Mahesh Ursekar.

References

[AG87] Jean Aitchison and A. Gilchrist. *Thesaurus construction: A practical manual.* Aslib, London, UK, 1987.

[Bar90] Richard Barnhart. The Advanced Naval Message Analyzer. Videotape production, VPI&SU, November 1990. Richard Barnhart as host, producer, script-writer; Edward Fox as executive producer, project director. Discusses the CODER system.

[BBB+87] N. J. Belkin, C. Borgman, H. Brooks, T. Bylander, W. B. Croft, P. Daniels, S. Deerwester, E. A. Fox, P. Ingwersen, R. Rada, K. Sparck Jones, R. Thompson, and D. Walker. Distributed expert-based information systems: An interdisciplinary approach. *Information Processing & Management*, 23(5):395–409, 1987.

[BGD86] Suzanne Bertrand-Gastaldy and Colin H. Davidson. Improved design of graphic displays in thesauri - through technology and ergonomics. *Journal of Documentation*, 42(4):225–251, December 1986.

[BSW83] Nicholas J. Belkin, T. Seeger, and G. Wersig. Distributed expert problem treatment as a model for information system analysis and design. *Journal of Information Science*, 5:153–167, 1983.

[CD87] Y. Chiaramella and B. Defude. Prototype of an intelligent system for information retrieval: IOTA. *Information Processing & Management*, 23(4):285–303, 1987.

[CD87] Hsinchun Chen and Vasant Dhar. Reducing indeterminism in consultation: a cognitive model of user/librarian interactions. In *Proceedings of the American Associate for Artificial Intelligence (AAAI-87)*, July 87. Also a report in the Working Paper Series for New York University CRIS #158 or GBA #87-60.

[Che90] Qi Fan Chen. An object-oriented network database system for information retrieval applications. VPI&SU Dept. of Computer Science, August 1990. Dissertation Proposal.

[CP88] L. M. Chan and R. Pollard. *Thesauri used in online databases: An Analytical guide*. Greenwood Press, 1988.

[CT87] W. Bruce Croft and Roger Thompson. I3R: A new approach to natural language processing for document retrieval systems. *Journal of the American Society for Information Science*, 1987.

[CUM85] D. G. Covell, G. C. Uman, and P. R. Manning. Academia and clinic: Information needs in office practice: are they being met? *Annals of Internal Medicine*, 103:596–599, 1985.

[DeJ82] G. DeJong. An overview of the FRUMP system. In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1982.

[DM90] Andrew Dillon and Cliff McKnight. Towards a classification of text types: a repertory approach. *International Journal of Man-Machine Studies*, 33:623–636, 1990.

[Dru90] Ronald Drusin. *Cardiology Section.* Columbia University, 1990. Course notes for Abnormal Human Biology, at Columbia University.

[FCDH90] Edward A. Fox, Qi Fan Chen, Amjad M. Daoud, and Lenwood S. Heath. Order preserving minimal perfect hash functions and information retrieval. In *Proc. SIGIR 90, 13th Int'l Conference on R&D in Information Retrieval*, pages 279–311, Brussels, Belgium, September 1990.

[FCF91] Edward A. Fox, Qi Fan Chen, and Robert K. France. A general reference model for hypermedia and information retrieval and its implementation in CODER/LEND. In Emily Berk and Peter Morley, editors, *Hypertext/Hypermedia Handbook*. McGraw-Hill, Inc., New York, 1991. To appear.

[FF87] E. A. Fox and Robert K. France. Architecture of an expert system for composite document analysis, representation and retrieval. *International Journal of Approximate Reasoning*, 1(1):151–175, 1987.

[FHCD90] Edward A. Fox, Lenwood S. Heath, Qi Fan Chen, and Amjad M. Daoud. Practical minimal perfect hash functions for large databases. Technical Report TR 90-41, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, August 1990. Submitted to CACM in 1989.

[Fid87] Raya Fidel. Controlled vocabulary and free-text searching: Searcher's selection of search keys. In *Proceedings of the American Society for Information Science Annual Meeting*, pages 71–73, 1987.

[Fox87] Edward A. Fox. Development of the CODER system: A testbed for artificial intelligence methods in information retrieval. *Information Processing & Management*, 23(4):341–366, 1987.

[Fri88] Mark E. Frisse. Searching for information in a hypertext medical handbook. *Communications of the Association for Computing Machinery*, 31(7):880–886, 1988.

[FWS+86] E. A. Fox, Robert C. Wohlwend, Phyllis R. Sheldon, Qi Fan Chen, and Robert K. France. Building the CODER lexicon: The Collins English Dictionary and its adverb definitions. Technical Report TR-86-23, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, October 1986.

[Gon87] G. H. Gonnet. Examples of PAT. Technical Report OED-87-02, UW Center for the New Oxford English Dictionary, Waterloo, Ontario, August 1987.

[GW87] G. H. Gonnet and Tompa F. W. Mind your grammar: a new approach to modelling text. In *Proceedings of the Thirteenth Conference on Very Large Databases*, pages 339–346, Brighton, 1987.

[HAL86] Gerry Henderson, Rachael K. Anderson, and Robert I. Levy. IAIMS at Columbia: Development of a strategic plan and model project. In *Bulletin of the Medical Library Association*, volume 74, pages 243–248, 1986.

[HL89] Betsy L. Humphreys and Donald A. B. Lindberg. Building the Unified Medical Language System. In L. C. Kingston, editor, *Proceedings of the 13th Annual Symposium on computer applications in medical care*, pages 475–480. IEEE Computer Society Press, 1989.

[HM86] Susan M. Humphrey and Nancy E. Miller. The NLM indexing aid project. In *Proceedings of the 49th Annual Meeting of the American Society for Information Science*, pages 106–112, Chicago, IL, September 1986.

[HR84] U. Hahn and U. Reimer. Heuristic text parsing in 'Topic': Methodological issues in a knowledge-based text condensation system. In Hans J. Dietschmann, editor, *Representation and Exchange of Knowledge as a Basis of Information Processes*, pages 143–163. North-Holland, New York, NY, 1984.

[Hun86] Stanley R. Huntting. Open wide and say dataa. In *CD ROM: The New Papyrus*. Microsoft Press, Washington, 1986.

[ISO86a] ISO. *Documentation - Guidelines for the establishment and development of monolingual thesauri.* International Organization for Standardization, 1986. ISO 2788-1986.

[ISO86b] ISO. *Information Processing – Text and Office Systems – Standard Generalize Markup Language (SGML), First Edition.* International Organization for Standardization, October 1986. ISO 8879-1986.

[JT84] K. Sparck Jones and J.I. Tait. Automatic search term variant generation. *Journal of Documentation*, 40(1):50–66, March 1984.

[KC90] Setrag N. Khoshaflan and George P. Copeland. Object identity. In Stanley B. Zdonik and David Maier, editors, *Readings in Object-oriented Database Systems*, pages 37–46. Morgan Kaufmann Publishers, Inc., 1990.

[KGBW90] Won Kim, J.F. Garza, N. Boolo, and D. Woelk. Architecture of the ORION next-generation database system. *IEEE Trans-*

actions on *Knowledge and Data Engineering*, 2:109–124, March 1990.

[Lan72] F. W. Lancaster. *Vocabulary Control for Information Retrieval.* Information Resources Press, VA, 1972.

[Mac90] Ian A. Macleod. Storage and retrieval of structured documents. *Information Processing & Management*, 26(2):197–208, 1990.

[MTR89] Charles F. McMath, Robert S. Tamaru, and Roy Rada. A graphical thesaurus-based information retrieval system. *International Journal of Man-Machine Studies*, 31:121–147, 1989.

[Ore87] Tim Oren. The architecture of static hypertexts. In *Proc. Hypertext '87*, pages 291–306, University of North Carolina, Chapel Hill, November 1987.

[Pol87] Steven Pollitt. CANSEARCH: An expert systems approach to document retrieval. *Information Processing & Management*, 23(2):119–138, 87.

[RGL87] J. R. Remde, L. M. Gomez, and T. K. Landauer. SuperBook: An automatic tool for information exploration-hypertext? In *Proc. Hypertext '87*, pages 175–188, University of North Carolina, Chapel Hill, November 1987.

[RJZ89] L. F. Rau, P. S. Jacobs, and U. Zernik. Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing & Management*, 23(4):419–428, 89.

[RL89] Roy Rada and L. F. Lunin. Introduction and overview to perspectives on hypertext. *Journal of the American Society for Information Science*, 40(3):159–163, 1989.

[RT88] Darrell R. Raymond and Frank Wm. Tompa. Hypertext and the Oxford English Dictionary. *Communications of the Association for Computing Machinery*, 31(7):871–879, 1988.

[Sha79] Stuart C. Shapiro. The SNePS semantic network processing system. In Nick Findler, editor, *Associative Networks*, pages 179–203. Academic Press, New York, NY, 1979.

[SRH90] M. Stonebraker, L. A. Rowe, and M. Hirohama. The implementation of POSTGRES. *IEEE Transactions on Knowledge and Data Engineering*, 2:125–142, March 1990.

[SSGC89] Philip J. Smith, Stephen J. Shute, Deb Galdes, and Mark H. Chignell. Knowledge-based search tactics for an intelligent intermediary system. *ACM Transactions on Office Information Systems*, 7(3):246–270, July 1989.

[Wal87] Janet Walker. Document Examiner: Delivery interface for hypertext documents. In *Proc. Hypertext '87*, pages 307–323, University of North Carolina, Chapel Hill, November 1987.

[WF88] M. Weaver and E. Fox. Implementing an intelligent retrieval system: The CODER system, version 1.0. Technical Report TR-88-6, Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, February 1988.

[WFCF89] M. Weaver, R. France, Q. Chen, and E. Fox. Using a frame-based language for information retrieval. *International Journal of Intelligent Systems*, 4(3):223–257, 1989.

[WLH90] K. Wilkinson, P. Lyngboek, and W. Hasan. The Iris architecture and implementation. *IEEE Transactions on Knowledge and Data Engineering*, 2:63–75, March 1990.

[Woh86] Robert C. Wohlwend. Creation of a Prolog fact base from the Collins English Dictionary. M.S. Report, March 1986.

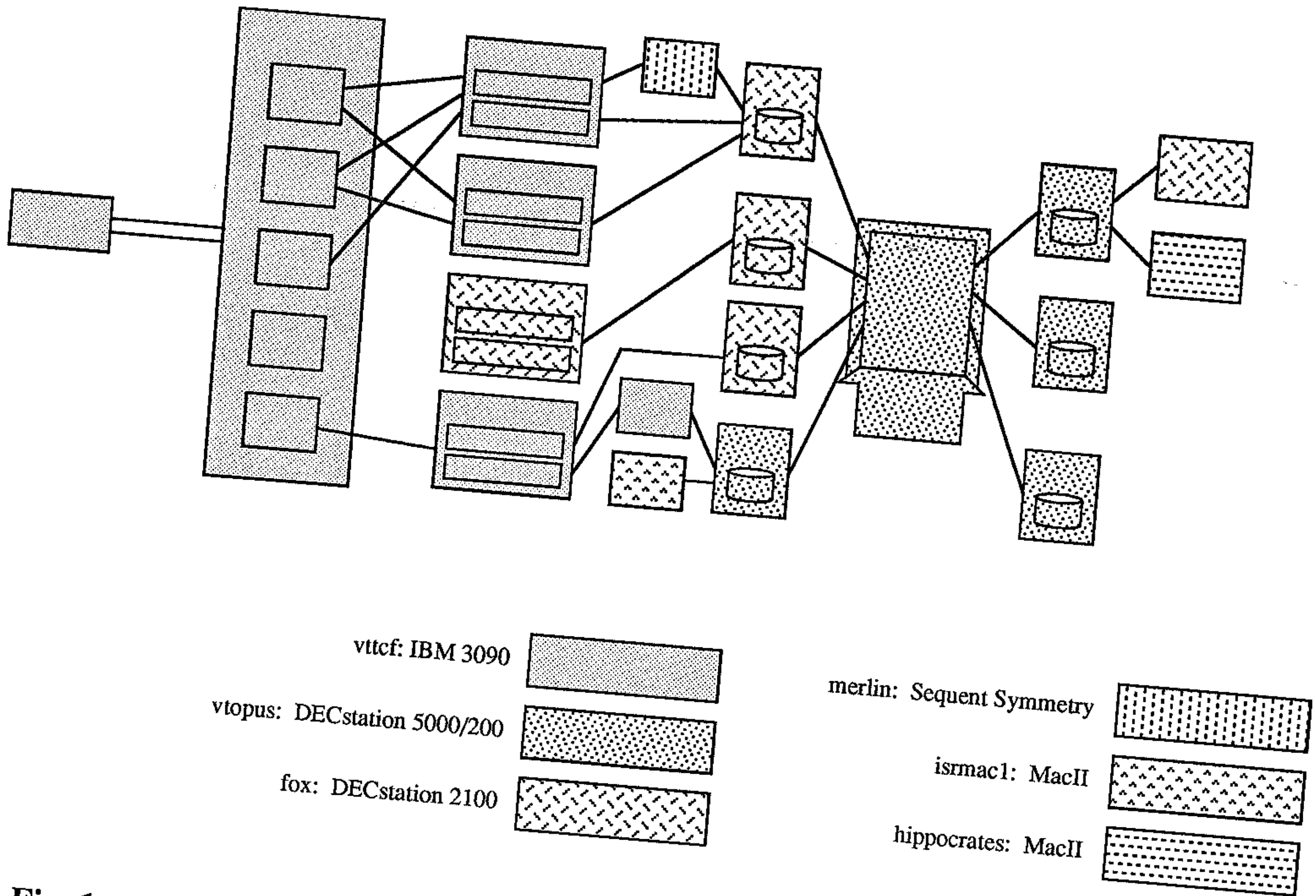


Fig. 1: CODER: A distributed system

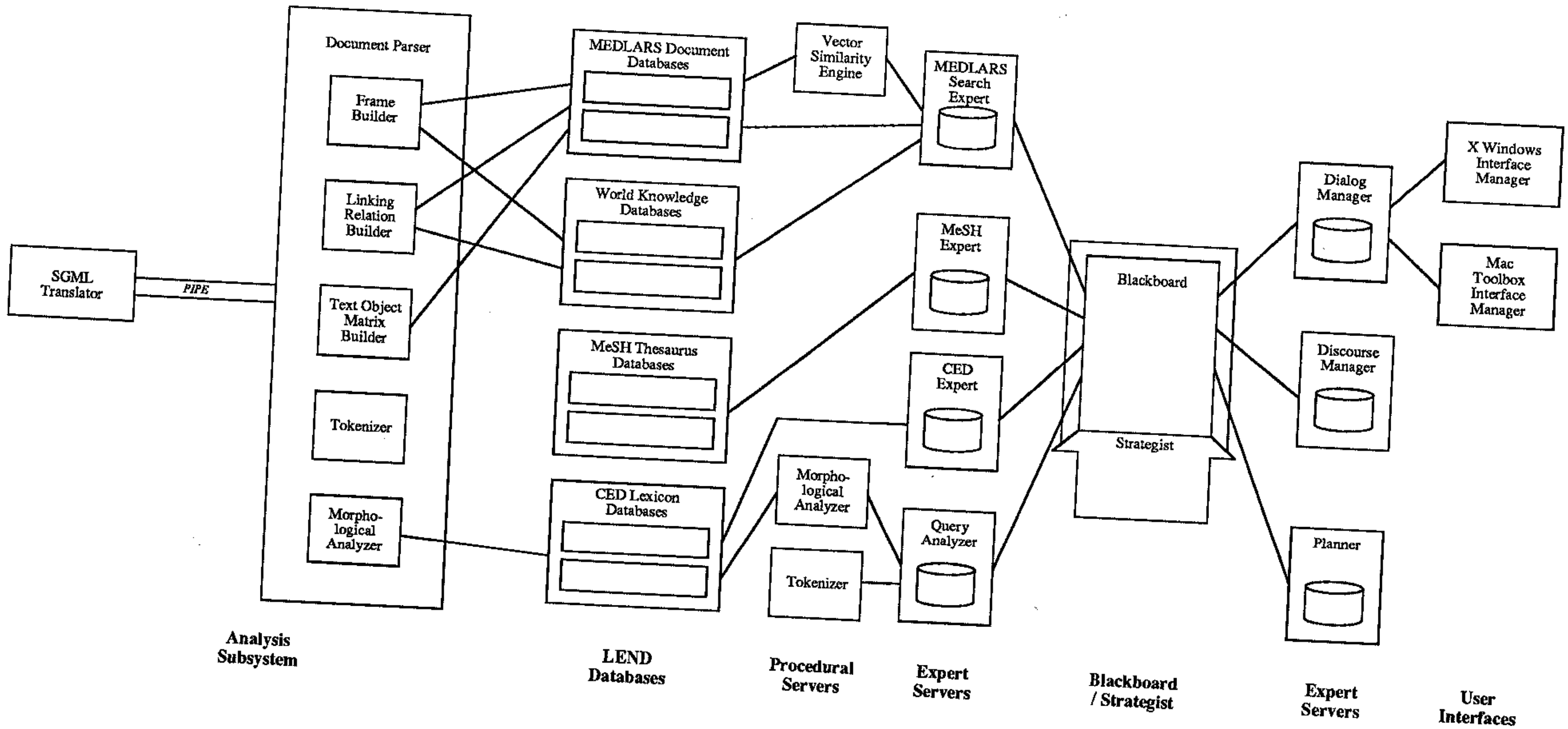


Fig. 2: CODER architecture.

Figure 3. LEND Architecture

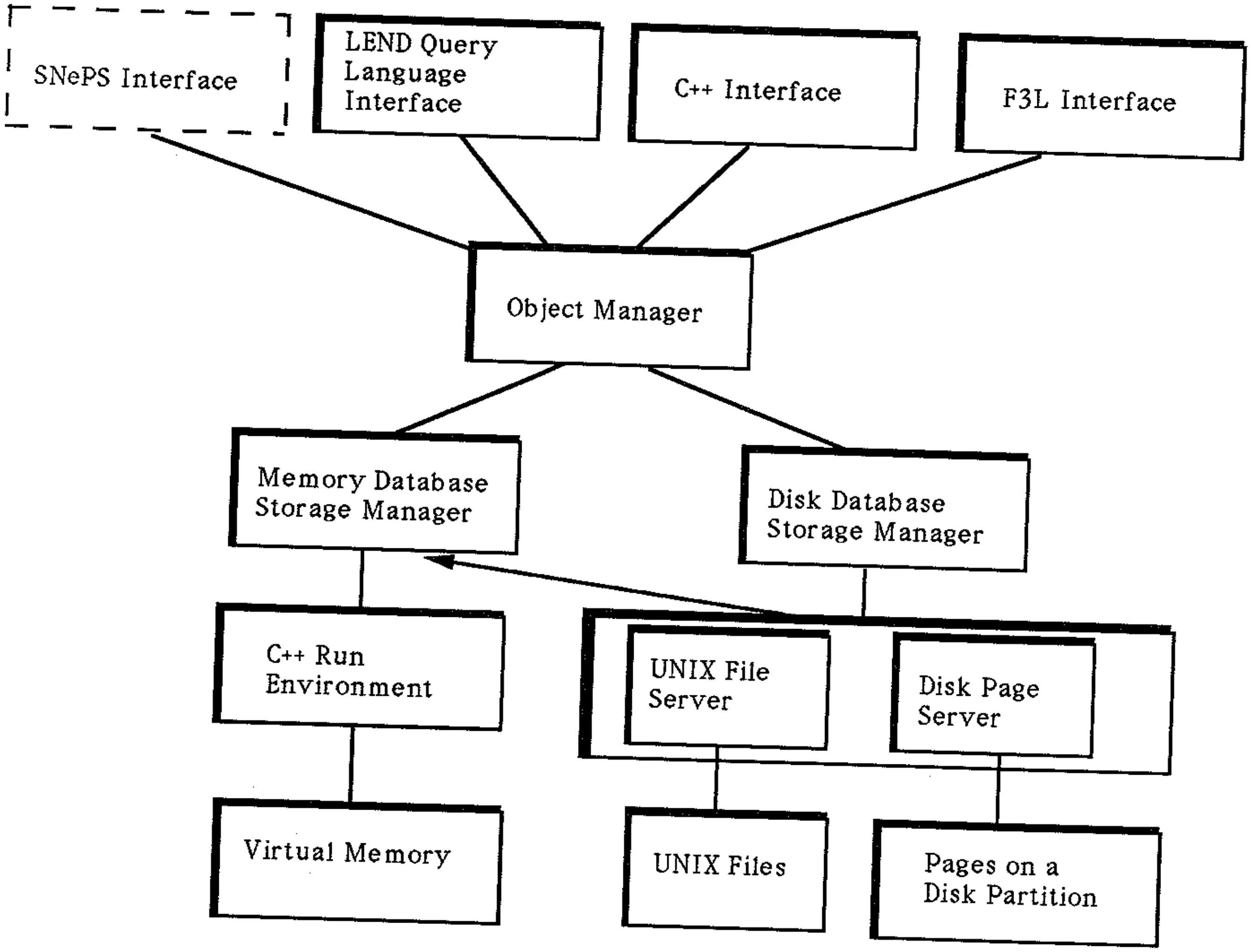
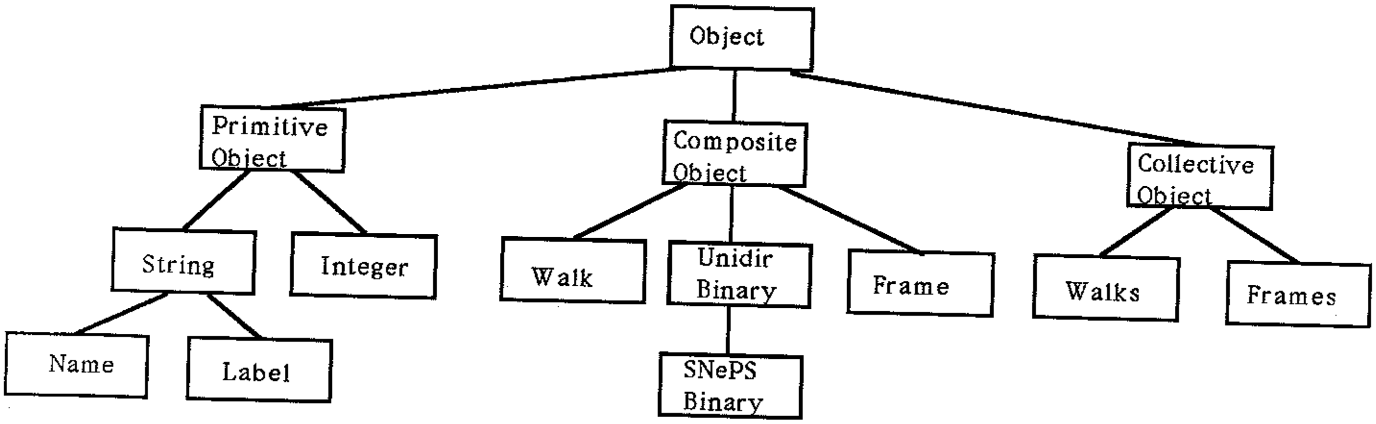
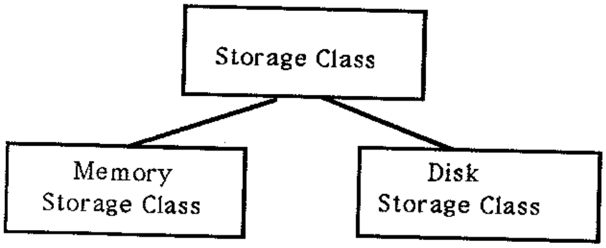


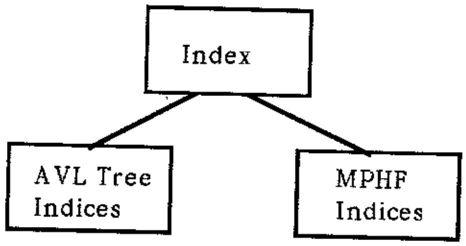
Figure 4. LEND Class Hierarchies



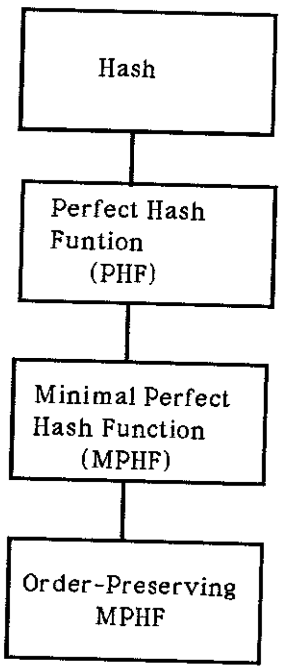
(a) Object Hierarchy



(b) Storage Hierarchy



(c) Index Hierarchy



(d) Partial Hash Function Hierarchy

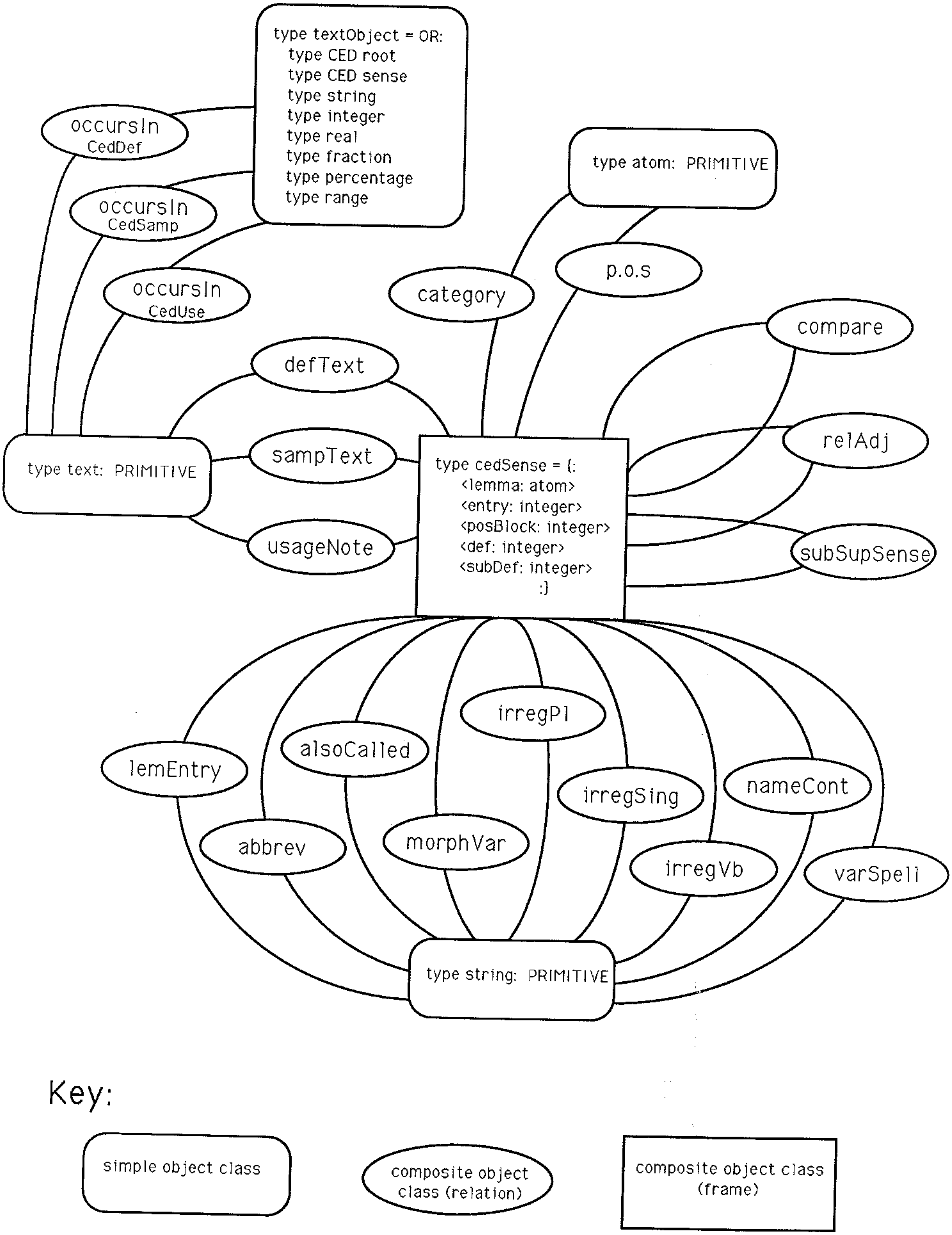


Fig. 5: CED entities and relations.

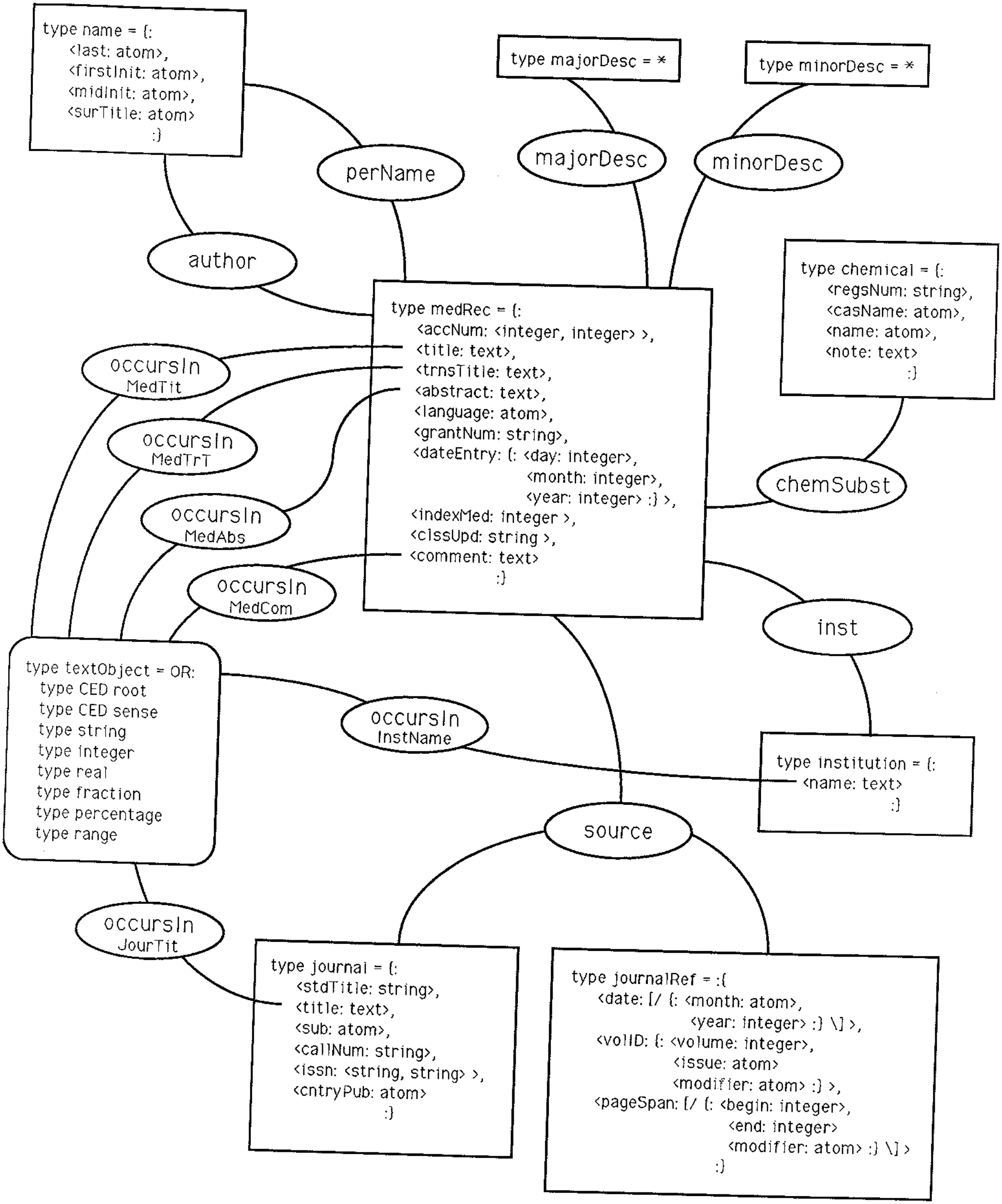


Fig. 6: MEDLARS document representation.

Figure 7(a) Example of Document Analysis - Raw Data

```
DOCUMENT   5236
AN 90166108.  90052.
AU Brattstrom-L.  Israelsson-B.  Norrving-B.  Bergqvist-D.  Thorne-J.
     Hultberg-B.  Hamfelt-A.
IN Department of Neurology, University Hospital, University of Lund,
     Sweden.
TI Impaired homocysteine metabolism in early-onset cerebral and
     peripheral occlusive arterial disease.  Effects of pyridoxine and
     folic acid treatment.
SO Atherosclerosis. 1990 Feb. 81(1).  P 51-60.  (Review).
JT ATHEROSCLEROSIS.
LG EN.
AB Severe homocysteinemia due to genetic defects either of pyridoxal
     5-phosphate (PLP)-dependent cystathionine beta-synthase (CBS) or of
     enzymes in vitamin B12 and folate metabolism is associated with very
     ...
     ...
     metabolism, which may contribute to vascular disease, and that the
     impaired metabolism can be improved easily and without side effects.
     Author-abstract. 42 Refs.
MJ ARTERIAL-OCCLUSIVE-DISEASES: dt.  CEREBRAL-ARTERY-DISEASES: dt.
     CYSTATHIONINE-BETA-SYNTHASE: me.  FOLIC-ACID: tu.  HOMOCYSTEINE: me.
     HYDRO-LYASES: me.  PYRIDOXINE: tu.
MN ADULT.  FEMALE.  HOMOCYSTEINE: bl.  HUMAN.  MALE.  MIDDLE-AGE.
     REVIEW.  REVIEW-TUTORIAL.  SUPPORT-NON-U-S-GOVT.  TIME-FACTORS.
RN EC-4-2-1 -- Hydro-Lyases.
     EC-4-2-1-22 -- Cystathionine-beta-Synthase.
     454-28-4 -- Homocysteine.
     59-30-3 -- Folic-Acid.
     65-23-6 -- Pyridoxine.
SB M.
YR 90.
IS 0021-9150.  95X.
ZN Z1.542.651.
IM 9005.
ED 900328.
```

Note: We have shown only part of the citation in the interest of brevity

Figure 7(b) Example of Document Analysis - SGML Tagged data

```
<docNo>5236</docNo>
<medRec>
<acNo><no1>90166108</no1><no2>90052</no2></acNo>
<auth><last>Brattstrom</last><fisrt>L</first><last>Israelsson</last>
<first>B</first>.........</auth>
<inst>Department of Neurology, University Hospital, University of Lund,
    Sweden</inst>
<title>Impaired homocysteine metabolism in early-onset cerebral and
    peripheral occlusive arterial disease. Effects of pyridoxine and
    folic acid treatment.</title>
<src><jrnl>Atherosclerosis</jrnl><date><year>1990</year><month>Feb</month>
 </date><volId><vol>81</vol><issue>1</issue></volId><pageNo><bgPg>51</bgPg>
<endPg>60</endPg> <mod>Review</mod></src>
<lang>EN</lang>
<abst>Severe homocysteinemia due to genetic defects either of pyridoxal
    5-phosphate (PLP)-dependent cystathionine beta-synthase (CBS) or of
    enzymes in vitamin B12 and folate metabolism is associated with very
    ...
    ...
    metabolism, which may contribute to vascular disease, and that the
    impaired metabolism can be improved easily and without side effects.
    Author-abstract. 42 Refs</abst>
<majDes><desc>ARTERIAL-OCCLUSIVE-DISEASES</desc><mod>dt</mod>
    ........</majDes>
<minDes><desc>ADULT. FEMALE. HOMOCYSTEINE</desc><mod>bl</mod>
    ...........</minDes>
<rgNo><<num>EC-4-2-1</num><name>Hydro-Lyases</name>..........</rgNo>
<issn>0021-9150. 95X</issn>
<cntrPub>Z1.542.651</cntrPub>
<indxMed>9005</indxMed>
<datNtry>900328</datNtry>
</medRec>
```

Note: We have shown only part of the tagged data in the interest of brevity

Figure 7(c) Example of Document Analysis - Frames and Realtions

Frames

medRec

ID	5236
accNum	<90166108, 90052>
title	21
abstract	
language	EN
grantNum	
dateEntry	<28 03 90>
indexMed	9005
classUpd	
comment	

journal

ID	1
stdTitle	ATHEROSCLEROSIS
title	atherosclerosis
issn	<0021-9150 95X>
cntryPub	Z1.542.651

journalRef

ID	1
date	<Feb 1990>
volId	<81 1>
pageSpan	<51 60 Review>

name

ID	1
last	Brattstorm
firstInit	L
midInit	
surTitle	
.....	
ID	7
last	Hamfelt
firstInit	A
midInit	
surTitle	

Relations

source

docInstId	5236
journalInstId	1
journalAnchor	<3948,2095>
journalRefInstId	1
journalRefAnchor	<3948,60>

majorDesc

docInstId	5236
desc	ARTERIAL-OCCLUSIVE-DISEASES
mod	dt
anchor	<5528,36>

occursInMedAbs

textObject	#101:58782#
docInstID	5236
numOccurs	6
anchorSet	{<1195,8>, <952,8>, <870,8>, <468,8>, <419,8>, <286,8>}

textObject	#101:34420#
docInstID	5236
numOccurs	3
anchorSet	{<1230,7>, <357,7>, <142,7>}

textObject	#4:240#
docInstID	5236
numOccurs	1
anchorSet	{<1003,3>}

Note: We have shown only a few of the frames and relations in the interest of brevity.

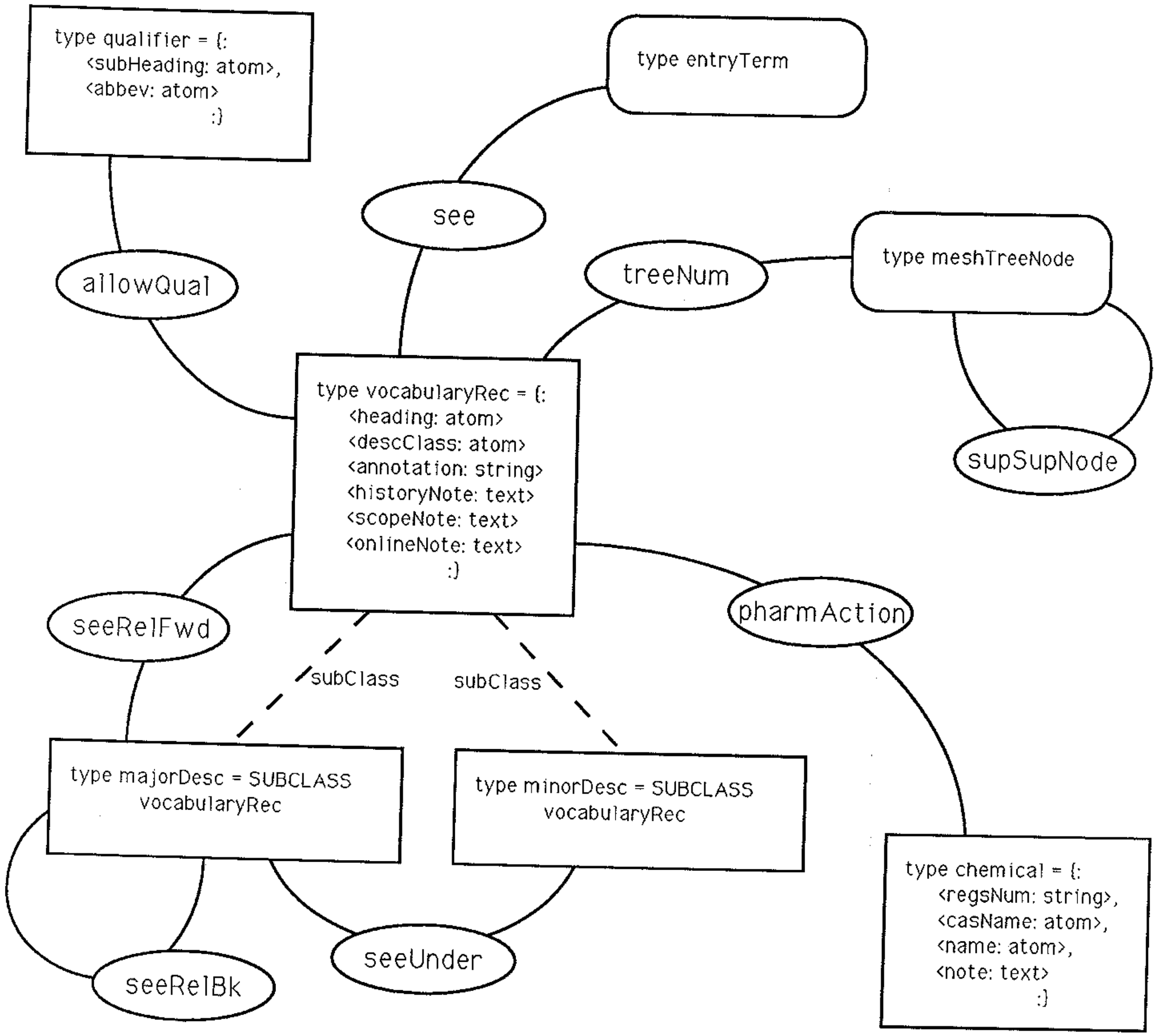


Fig. 8: MeSH thesaurus represention.

Figure 10. MeSH Descriptor Query Window

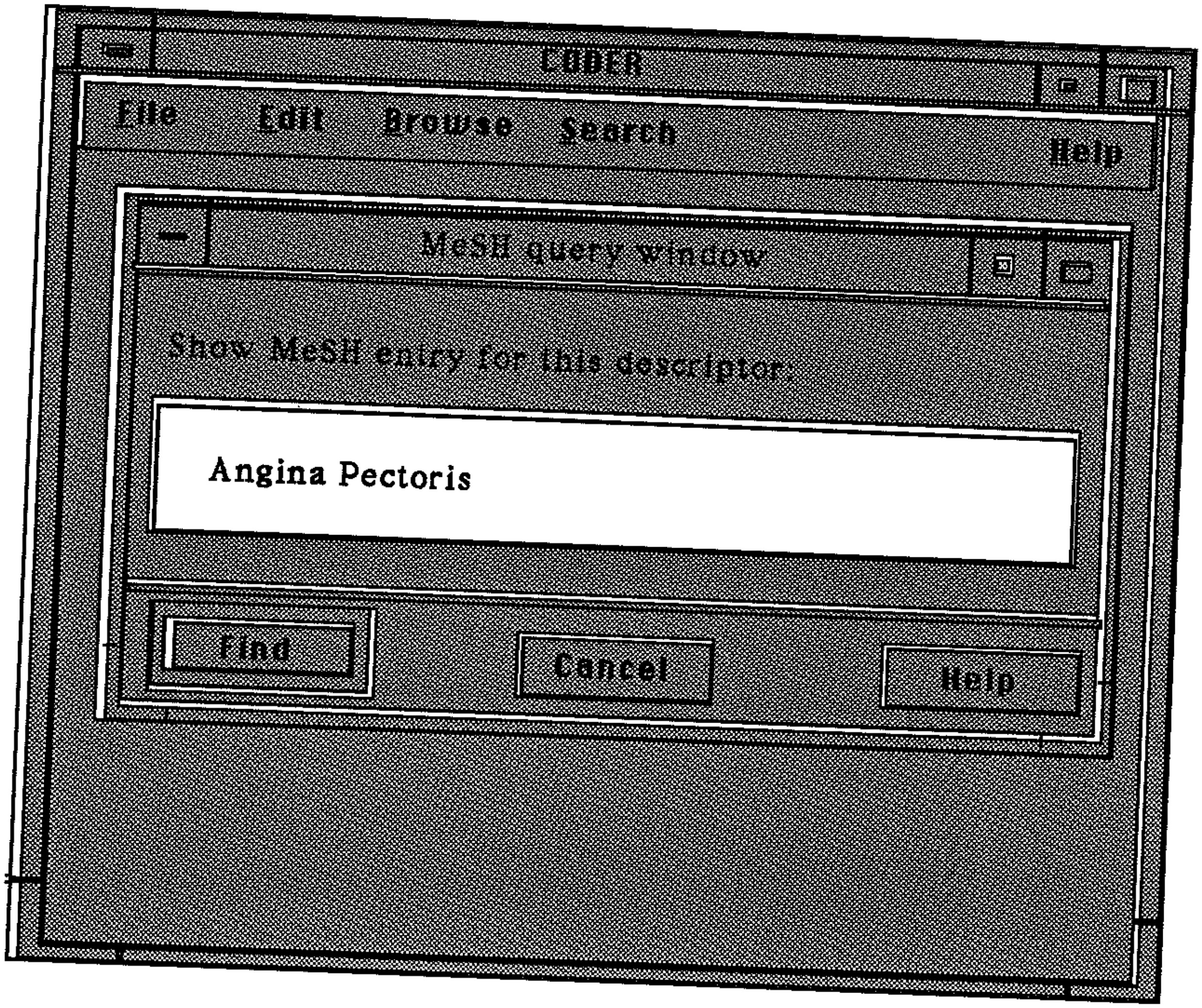


Figure 11. MeSH Descriptor Information Window

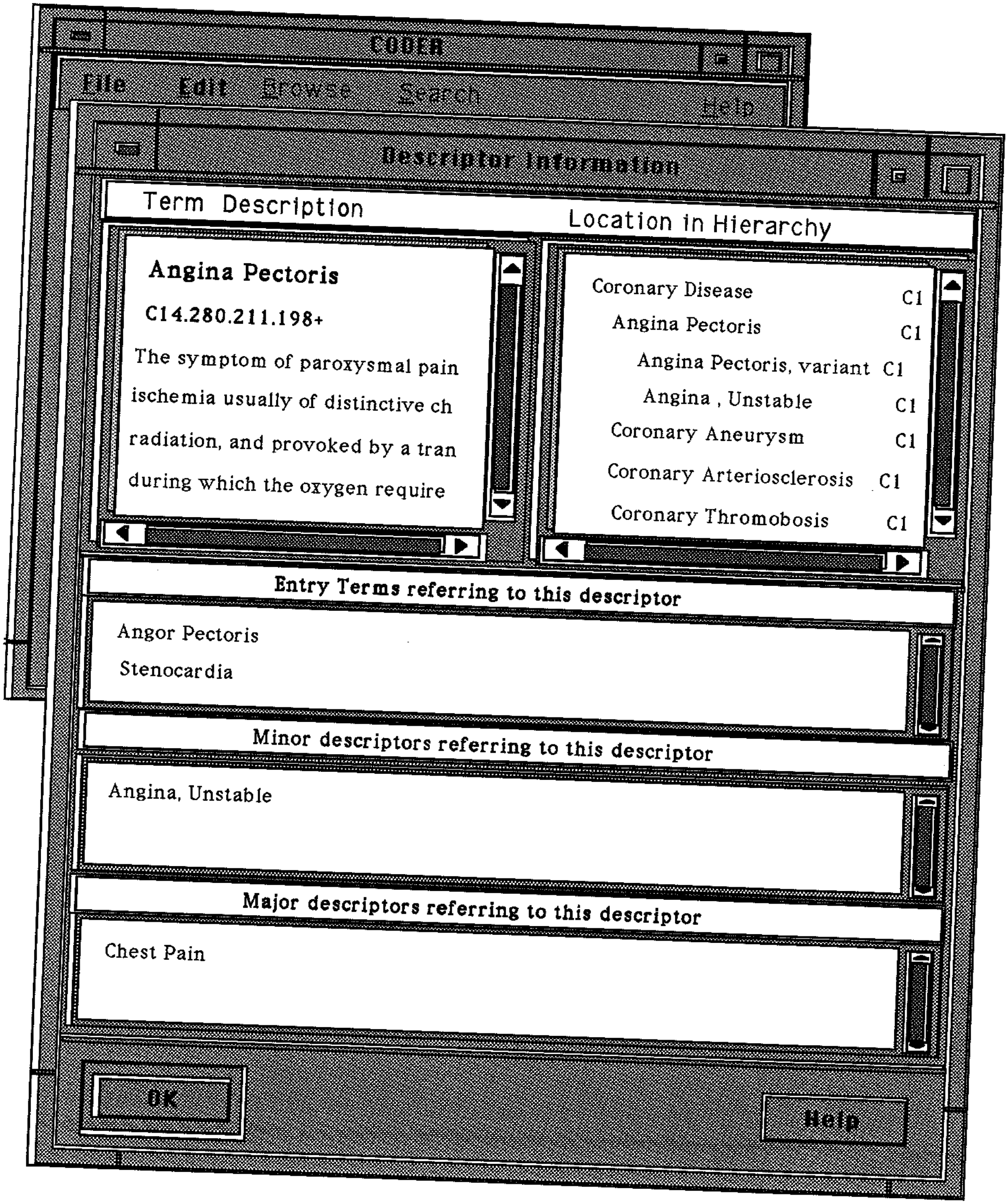


Figure 12. Cardiology Course Notes Window