1988

COGMIR: A Computer Model for Knowledge Integration.

Zheng Xin Chen
Louisiana State University and Agricultural & Mechanical College

Order Number 8917804

COGMIR: A computer model for knowledge integration

Chen, Zheng Xin, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

COGMIR: A COMPUTER MODEL FOR KNOWLEDGE INTEGRATION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Zheng Xin Chen
B.S., East China Normal University, 1982
M.S., Louisiana State University, 1985
December 1988

ACKNOWLEDGEMENTS

Sincere thanks are due to my major professor, Dr. Bush Jones for his guidance in the important methodology used to carry out my dissertation research. So many times, he has shown me the way to do creative work and pointed out the pitfalls of which I should be aware. From the first course I took from him four years ago, he has been one of my most respected professors, and I am glad to be able to complete my dissertation work under his guidance. Special thanks are also due to Dr. Donald H. Kraft, Chairman of Computer Science Department, for his encouragement of my doctoral research, for the ideas rooted in his papers, and for his input on a previous version of this dissertation.

I would also thank Professor Leslie P. Jones, for his guidance of my graduate studies in general, and for my dissertation in particular. In addition, I am very glad to have Professor Doris Carver to be a committee member, her warm encouragement has been important to me.

Outside the Computer Science Department, for three years, Professor Robert C. Mathews of the Department of Psychology has guided me into the area of cognitive science, and has helped me a lot in organizing my thoughts. Special thanks are also due to Professor M. El-Amawy of the Department of Electrical and Computer Engineering for his excellent teaching. I would also thank Dr. William H. Herke, Mr. Barton D. Rogers and Mr. E. Eric Kneudsen for their kind support and friendship.

Table of Contents

List of Figures

ABSTRACT

Knowledge integration is an important topic for knowledge engineering. In this dissertation, we explore some aspects of knowledge integration, namely, accumulation of scientific knowledge and performing analogical reasoning on the acquired knowledge.

Knowledge to be integrated is conveyed by paragraph-like pieces, these pieces will be referred to as documents. By incorporating some results from cognitive science, the Deutsch-Kraft model of information retrieval is extended to a model for knowledge engineering, which integrates acquired knowledge and performs intelligent retrieval. The resulting computer model is termed COGMIR, which stands for a COGnitive Model for Intelligent Retrieval. A scheme, named query invoked memory reorganization, is used in COGMIR for knowledge integration. Unlike some other schemes which realize knowledge integration through subjective understanding by representing new knowledge in terms of existing knowledge, the proposed scheme suggests at storage time only recording the possible connection of knowledge acquired from different documents. The actual binding of the knowledge acquired from different documents is deferred to query time, depending on the actual needs of the query. Therefore, although there is only one way to store knowledge, there are potentially numerous ways to utilize the knowledge.

From the classical information retrieval viewpoint, we have extended the original model in the following sense, not only each document be represented as a whole, but also the meaning of each document can be represented. In addition, since facts are

constructed from the documents, document retrieval and fact retrieval are treated in a unified way. Moreover, when the requested knowledge is not available, query invoked memory reorganization can generate suggestion based on available knowledge through analogical reasoning. This is done by revising the algorithms developed for document retrieval and fact retrieval, and by incorporating Gentner's structure mapping theory. Analogical reasoning is treated as a natural extension of intelligent retrieval, so that two previously separate research areas are thus combined.

A case study is provided to demonstrate the fundamental ideas. All the components are implemented as list structures, which bears an interesting similarity to relational databases.

CHAPTER 1

INTRODUCTION

1.1 Integration of scientific knowledge

Knowledge acquisition has been identified as the "bottleneck" in knowledge engineering, particularly in the building of expert systems [Coom84]. For a human being, knowledge acquisition is a tedious process. Knowledge is acquired piece by piece, chunk by chunk, therefore, there is a need for knowledge integration. Since knowledge is acquired with so much difficulty, there is also a natural desire to fully utilize this acquired knowledge.

How does a computerized information system deal with knowledge integration? Knowledge integration differs from the plain storage of knowledge in that it is not the original form of the input is stored word by word, but its internal representation (i.e., its "meaning") is stored, and the knowledge obtained from the different inputs should be organized to facilitate later use.

There have been some research which address some aspects of knowledge integration. We start our investigation from a computer program, CYRUS [Kolo83], which records diplomatic activities about two former Secretaries of States, Vance and Muskie. Preprocessed by FRUPMP [DeJo79], the actual input to CYRUS takes the form of events which record the diplomatic activities of these two persons. For any event of a "diplomatic meeting", the person involved in this event (the actor), the counterpart of this actor, and the topic of that meeting are recorded. For instance, one event may record the actor as Vance, the counterpart as Begin, and the topic is the

Camp David Accords. There may exist another event which takes Vance as the actor, but with Gromyko as the counterpart, and the topic is the SALT talks. The features common to these events are put in a content frame (which forms a "node" in a gradually constructed tree-like structure), while the difference between these events is used as indexing (each event is stored as a "leaf"). Therefore, by using a scheme "indexing by difference," a discrimination tree is constructed to store the events, and knowledge integration is thus realized. When new input enters the system, it is compared with the existing events, the process of constructing the discrimination tree thus continues in a similar manner. Various retrieval strategies in CYRUS have been described, and through some kind of inference, the system is able to answer questions which were not explicitly stated. Another work that concerns knowledge integration is IPP [Lebo83, Lebo85], which takes terrorism stories as input. For instance, one story may be a short input describing three gunmen attacking a U.S. army vehicle at a certain place, and another story reporting that the IRA killed an old, unarmed security guard in some other place. IPP integrates the input through generalized memory.

Although some differences exist between these two projects (e.g., CYRUS is able to deal with greater details about events, while IPP can deal with a larger quantity of events) [Lebo83], both these works realize Schank's theory of dynamic memory [Scha82, Scha83]. Schank discusses the integration of various events such as "eating in restaurants" or "car accidents," and proposes some "knowledge structures" to handle such. What are the common features of the input for these works? All of them concern episodic memory (the memory concerning personal experience). Each of them is pertinent to a certain topic and is restricted in a small knowledge domain.

Moreover, all the input for a project look alike; they may be viewed as different instances of a more general phenomenon (eg. diplomatic activities or terrorism). Consequently, integration for this kind of input implies: *(1) generalization,* i.e., organizing the input instances; and *(2) interpreting new input in terms of old.* That is, to choose appropriate existing structures, to extract features and traverse indices, and to construct a discrimination tree.

This kind of integration is usually referred to as *subjective,* because new input is interpreted through existing knowledge. Subjective understanding is also the theme of some other works (eg. [Carb81]). Consequently, the memory (the place to store knowledge) is reorganized whenever a new input item is acquired. These works do show some important aspects of knowledge integration. Nevertheless, knowledge integration still means more. For one thing, episodic memory is not the only type of memory that should be investigated. Moreover, sometimes it is preferable that some kind of knowledge is stored in a "neutral", i.e., non-subjective manner, so that it can be used later through various ways, depending on various query needs. This cannot be realized through the models such as CYRUS or IPP. What is more, it has been pointed out that the structure of CYRUS lacks predictive power [Coom84], which is a crucial point in knowledge engineering.

The term "knowledge" may mean quite different things for different people. In this thesis, we will exclusively use it to refer to "compressed information" [Mich83] contained in the input. In order to explain what we are trying to do, consider the following four items, each carrying a piece of scientific knowledge.

SAMPLE INPUT OF SCIENTIFIC KNOWLEDGE

Harvey discovered that there are two kinds of blood vessels, the veins and the arteries. The veins carry blood to the right side of the heart. The arteries carry blood away from the left side of the heart.

Newcomen devised a system for passing steam into the cylinder of his engine. The steam was then cooled, and as it cooled, it condensed, or turned to water. Since water takes up so much less room than steam the pressure dropped in the cylinder and a piston slipped down. The up-and-down movement of the piston operated a pump that sucked water out of the mine.

The tiny blood vessels, or capillaries, connect the arteries and veins. With the aid of his microscope Malpighi examined the lungs of a frog and saw for the first time the blood traveling from the arteries into the capillaries and then into the veins.

People became convinced that bats found their way by hearing, and that they did this by emitting ultra-sonic sound which was reflected from obstacles close by and served to warn the bat so that it could avoid the objects.

These paragraph-like inputs are directly taken from junior science books [Greg63, Burk78, Gera68] with slight revision, but they may also be abstracts obtained by using a "skimmer" to longer articles [Maud87]. What are the features of these inputs? They are not "events" that concerns some person's particular experi-

ence. Each of these inputs is a partial conclusion or can be viewed as a "mental model" (i.e., the way people view the physical world). The contents of these input bring us knowledge about objects (or concepts) and their relationships. Here the objects are "hearts", "arteries", and so on, and these objects are associated through relationships such as arteries "carry" blood "away from" the left side of the heart. We also notice that these inputs do not necessarily fall in the same knowledge domain. For instance, in the previous four inputs, the second input falls in a domain different from the first and the third input. In fact, the formation of domains themselves are the intermediate results of knowledge integration, therefore they are not suitable to be used as the starting point of knowledge integration. A more reasonable assumption is to allow input fall in more than one domain. In fact, even the domain experts do not work in isolated domains; rather, knowledge can be transferred from some other domains to improve their expertise, and vise versa.

The term "integration" used here has a somewhat meaning different from that seen in CYRUS and IPP. An aspect of integration for the sample input on page 4 is to *accumulate partial, incomplete knowledge* and *to utilize the acquired knowledge to solve problems implied in the query.* We human beings have this kind of ability. People construct mental models and then reason by manipulating the model [John83]. And what happens after these mental models are constructed is our major concern. Consider the following example: assume that a person's knowledge only consists of the previous four items; after he has read through these four pieces and has them stored in his memory, if he is asked to tell some thing about heart and capillaries, he should be able to tell not only the role of arteries and veins (from Harvey's original

discovery) but also the involvement of the capillaries (Malpighi's discovery). That is, he should be able to construct an answer such as:

> There are two kinds of blood vessels, the veins and the arteries. The veins carry blood to the right side of the heart. The arteries carry blood away from the left side of the heart. The tiny blood vessels, or capillaries, connect the arteries and veins. The blood travels from the arteries into the capillaries and then into the veins.

This text-like paragraph is none of the four inputs shown above; rather, it is constructed by extracting relevant material from these inputs. This example indicates the way we human beings integrate knowledge. Knowledge acquired from different sources must be tied together in some way when it is stored ("remembered"), otherwise it cannot be used later. But knowledge integration is made explicit only when some queries come. When a person is asked (by somebody or by himself) about the relationship between "heart" and "capillaries", he will search his memory so that the related knowledge can be reorganized to form an answer. He may reason in such a way: "Nobody has ever told me anything about their relationship before, but I can trace my memory, starting from those items that are at least partially relevant to this query. I was told once that the heart is connected with arteries and veins. I also know something about Newcomen's engine, but that does not seem to be relevant to the topic. What else do I know? I was also told that arteries and veins are connected by capillaries. So the heart and capillaries must be connected through arteries and veins in such such a manner." If a person has more knowledge stored, he will still follow a similar process, and will reorganize only those parts of his knowledge that are relevant

to the query.

Integration is more than just tying things together. The stored knowledge has predictive power so that it can be used to do inference in some way. For instance, historically, the knowledge about the "bat" (as appeared in the previous example) was used for the invention of radar. How could it happen? The person who made this discovery did not invent radar. The knowledge about "bat" is utilized only when some special kind of need (e.g., use something to detect an enemy's plane) comes, which takes the form of a query. Notice here that what is originally requested is not the knowledge about "bat"; but since the requested knowledge is not available, previously acquired knowledge which is *similar* to the current situation is reorganized, retrieved, and is used to generate a kind of "pseudo" knowledge -- that is, suggestions to the query. In this example, the similarity exists because the task "to detect and avoid the invisible obstacle" for the bats is similar to the task "to detect the plane which cannot be seen" for human beings. Since bats can emit a special kind of sound to reflect from the obstacle, why don't we human beings do similar things to detect the enemy's plane? Stored knowledge is thus used to solve problems implied in the queries through a kind of inductive inference. What we want to point out here is that memory reorganization happens only when some query comes.

Therefore, the query plays the same role of an *exam* which is used to test whether a student understands some acquired knowledge. The query is also related to creativity that is shown in the process of solving the problem implied in the query. Knowledge integration is made explicit by locating a portion of memory at the time of answering a query. This kind of memory manipulation is referred to as *memory*

reorganization. Notice that memory reorganization does not require physical reconstruction, although it may be accompanied by new knowledge generation (as seen in the "radar" example).

To summarize, we have seen a kind of human memory behavior which can be termed as *query invoked memory reorganization,* and we will use it as a scheme to realize knowledge integration in a computer model. Here the term "memory" refers to the place to store the knowledge which will be called the *knowledge space.* According to this scheme, at storage time only possible ways of "binding" new input to old knowledge are recorded. The actual realization of knowledge integration is deferred to query time. The advantage of this scheme is, *although there is only one way to store knowledge, there are potentially numerous ways to utilize knowledge. The actual way of utilizing knowledge depends on the query.*

From the examples above, we also notice that an important feature of this scheme is that knowledge is not stored using content or domain relevant criteria. Rather, knowledge conveyed by each knowledge piece is represented in the same way -- through objects and their relationships. Domain related knowledge can be organized through the structure of the model when query comes.

The motivation of this dissertation is to explore the possibility of using a computer model to do a kind of knowledge integration as illustrated in the "heart" and "bat" examples. We will emphasize this aspect of knowledge retrieval, and point out how our scheme can be extended so that knowledge retrieval becomes the starting point of analogical reasoning. This kind of investigation makes sense, because it facilitates knowledge acquisition, which is an important topic in knowledge

engineering.

Since the problem to be investigated covers a wide area, we will focus on finding a small set of principles to build the most fundamental framework of intelligent knowledge retrieval. Basically, our problem is concerned with three phases: (i) process the inputs into memory (including natural language understanding and knowledge representation), (ii) operate on the memory, and (iii) reconstruct related part from the internal structure to answer queries (including schemes for memory reorganization and natural language text generation). We will focus on the second phase, and the language understanding features in the first and third phases will be simplified.

To summarize, we may rephrase the purpose of this dissertation in the following, slightly different, way: we are interested in ways of "reproducing" some aspects of knowledge integration that may be involved in scientific discovery and technical invention by using a computer model. The history of science has been investigated both from information science perspective [Pric68, Pric77], and from an artificial intelligence (AI) perspective, where AI has been viewed as a compiled hindsight of history of science [Dard87]. Our investigation will add one more dimension that so far has been ignored, from the perspective of an integrated cognitive process. In addition, this investigation is also a study of "parts and whole," an important topic discussed in general system theory [Klir85, Jone85]; here the term "parts" refers to acquired knowledge pieces while "whole" refers to the integrated result for some queries. But since we have emphasized the involved cognitive *process* through a symbolic approach, our work is complementary to the existing quantitative analysis of

cognition [Jone85].

1.2 Knowledge integration and intelligent information retrieval

What are the structure requirements to realize knowledge integration? We have mentioned that there must be a common place (a net constructed from objects and relationships) so that knowledge from different knowledge pieces can be stored; that is, this place contains the internal structure of the knowledge acquired from various input pieces. Now imagine that many knowledge pieces have entered the system, each of them converted into internal form (i. e., its meaning is stored), and each occupies a certain area in a large geometrical expansion (i e., the knowledge space). It will be hard to find any specific object if the information from this object is not recorded. But, finding a particular object is required for integration. From this consideration comes another requirement; that although the knowledge from different pieces is tied together, these pieces should be *distinguishable*, so that these pieces of knowledge can serve as "clues" for starting searching in the knowledge space, just as we saw from the previous "heart" example where previously obtained inputs were scanned. This is a reasonable requirement, because each input is a cognitively meaningful unit.

To meet these requirements, we extend the classical model of information retrieval (IR) to develop a new framework. General descussion on classical IR can be found in [Salt84, vanR79]. The reason to resort to IR is obvious, because the knowledge pieces can be treated as cognitive documents (for instance, in the previous example there are four such documents), and the topics of storage and retrieval of these documents (which are required for integrating these documents) have been been studied from the point of view of IR. Although we cannot directly use the IR model,

starting from the existing model is better than starting from scratch. What is lacking with the classical IR approach is that it does not concern internal representation nor the manipulation on this representation (for instance, classical IR is not interested in the issues as illustrated in the example given in section 1.1). But the classical IR model can be enriched.

Basically, the task of information retrieval is to acquire, analyze, index and store the information for later retrieval. The traditional area of information retrieval concerns only the storage and retrieval of documents. Central to the research in document retrieval is the carrying out this task efficiently and effectively. The traditional IR systems are concern with "where to find information in it." In order to change this to "know a subject", recently fact retrieval (sometimes also called intelligent retrieval) has been developed. Here to "understand" the contents of the documents becomes a central task, as shown in [Kolo83] and [Lebo83]. But the research of fact retrieval usually starts with a philosophy different from document retrieval, so there is a gap between these two types of retrieval. What is more, although "fact retrieval" is concerned with the retrieval of the contents of the documents, the term "fact" itself never has a generally accepted definition. An assumption used (either explicitly or implicitly) is to view the content contained in a single document as a fact, but this is not consistent with the original purpose of fact retrieval which is supposed to deal with content retrieval in general.

Our approach provides one possible way to get around this problem. We start from the basic considerations of document retrieval to approach fact retrieval. In terms of the document retrieval, query invoked memory reorganization can be re-

stated as follows: we view a *fact* as a generated "document" constructed by extracting contents from relevant documents. Starting from this viewpoint, we can deal with document retrieval and fact retrieval in a unified manner: both of them are retrieval units. A fact differs from a document in that a document must be an input unit while a fact is not. Our approach can be described as answering a query by "writing" a text on user's demand.

What are the relationship and difference between classical IR and the model we propose? To summarize, from the previous example we may claim that the topics in traditional IR, such as retrieval by author, year, etc., is no longer our major concern. For instance, for the four documents given in the previous example, we are not interested in a query of the type, say, "retrieve the document extracted from a book published in 1978 where the first four letters of the author's last name are 'Burk'." On the other hand, unlike the classical IR systems that only identify the relevant documents and ignore the task of determining how these documents are related to the query, a new approach is proposed so that the computerized system can present the user a *text* (either a document previous acquired by the system or something similar to a document which is written by extracting related material available from the documents).

The relationship and difference between classical IR and intelligent retrieval can be stated as follows. For classical retrieval, the task is

> ***Given query Q, get identifications of D, where D is a set of documents which satisfies the features or keywords requested in Q.***

In intelligent retrieval, as in our approach, the task is

> *Given query Q, get T, where T is a text (either a document reconstructed from its internal structure or a fact constructed from the internal structure of several documents).*

Roughly speaking, we have some common interests with Schank, but we are more interested in scientific knowledge, and there is also a difference in the ways of dealing with memory reorganization.

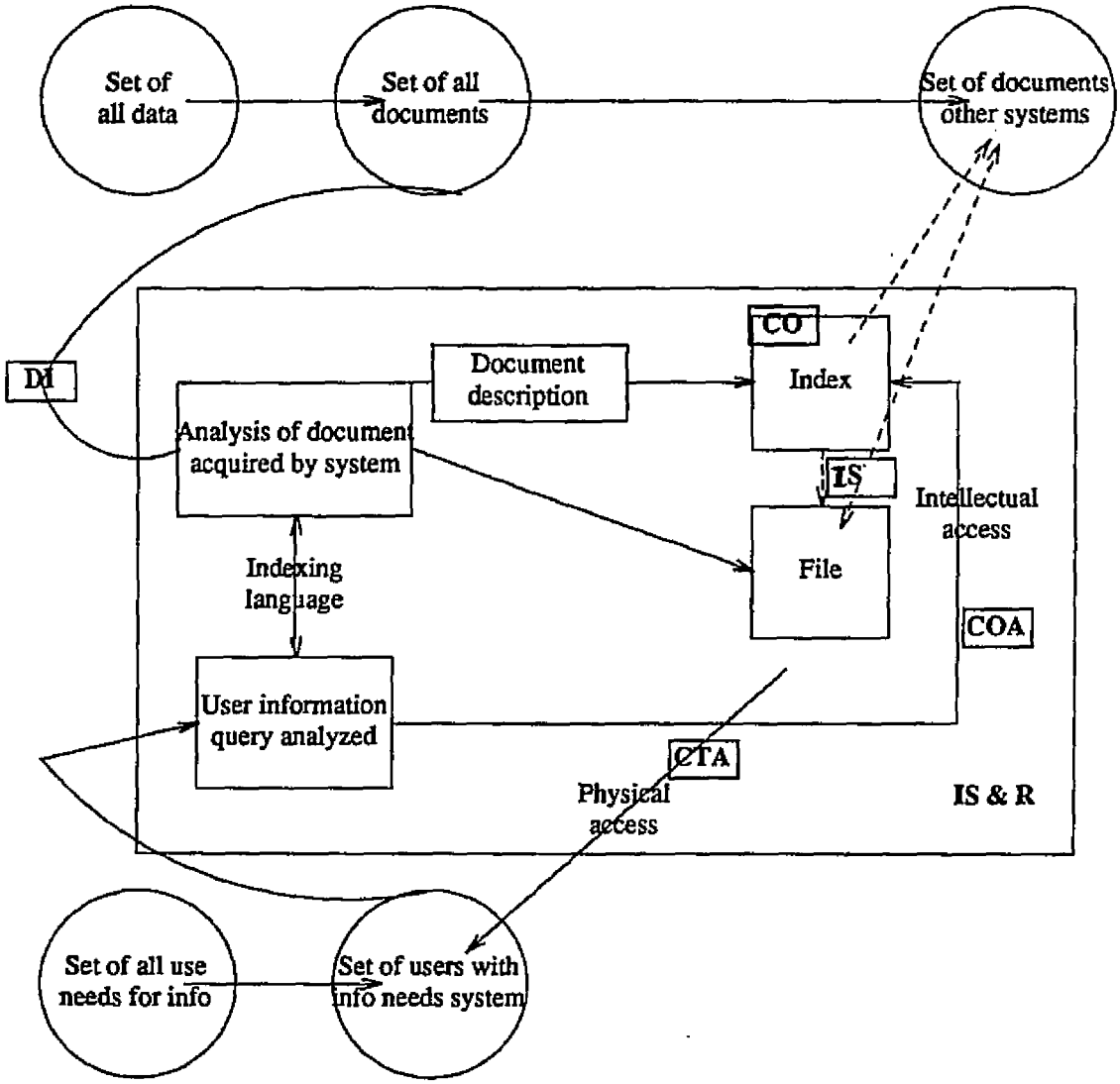
A model without inference ability is useless. We have claimed that knowledge integration not only means accumulating knowledge, but it also means utilizing this accumulated knowledge in a holistic manner. In classical knowledge engineering systems (such as expert systems), reasoning is usually done by utilizing rules driven by an inference engine. But there are some disadvantages for rule based systems (to be discussed in Chapter 2), and our approach basically is not rule-based. How is knowledge utilized in our approach? For instance, how can a computer model "invent" radar (or something equivalent) based on its knowledge of "bats"? Again, we will use the human being as a guide, therefore, in the approach we take, inductive reasoning plays an important role. Related issues involve generalization, abstraction, and specific inductive methods such as analogical reasoning. These issues are similar to those in the case-based reasoning discussed by Kolodner and Lebowitz, while differing from these authors in that we do not discuss those issues related to episodes, but related to compressed scientific knowledge.

1.3 Useful ideas from classical information retrieval

Information retrieval is a topic which intrigues both psychologists and computer scientists. According to [Salt83], computerized information retrieval (IR) is concerned with the representation, storage, organization, and accessing of information items. According to the view of computer science, a central concept in computing is the notion of the data structure as a logical construct of data items and relationships. Inherent in this notion of a data structure is a set of algorithms for manipulating the data structure, especially for the purposes of storing and retrieving selected data items. Expanding on this theme, there is a natural desire for those working on database management systems to seek better mechanisms for storing large amounts of data, modifying the data over time, and processing queries in order to retrieve the needed data [Kraf85].

These observations have revealed the very nature of information retrieval. A computerized IR model provides a kind of virtual machine, which is sometimes implemented in hardware (e.g., the Utah text retrieval machine [Holl79] or the scheme described in [Walt87] utilizing a connection machine [Hill85]).

The architecture of the model of information retrieval which is taken from [Kraf83], and will be referred to as **Deutsch-Kraft model,** is reproduced in Figure 1.1. In the following we examine its components and identify the basic principles, denoted in the small boxes close to related components.



DI: *Discrete input principle* **CO:** *Concept principle* **IS:** *Index and storage principle*
COA: *Conceptual access principle* **CTA:** *Content access principle*

Fig. 1.1 The Deutsch-Kraft model of information retrieval
(Reproduced from [Kraf83], with principles added)

Daia. Since the original concern of this model is data, we are able to cover a broad range of information systems. Usually the data concerned in IR systems take the form of documents which are basic units to be stored and retrieved. This observation can be summarized as following principle.

> **(DI) Discrete input principle.** *The input of an IR system should be in discrete units.*

Documents. In order to identify different documents so that they can be retrieved later, these documents must be represented in some way. This is stated in the following principle.

> **(DR) Document representation principle.** Documents must be represented to identify their existence.

Indexing. In a typical IR system, retrieval is accomplished by utilizing an index, which is a set of terms used as document descriptors. In conventional IR systems the key ingredient is the indexing language by which the indexing process can communicate with the query processing mechanism [Kraf85]. Indexing realizes the following storage-indexing principle.

> **(SI) Storage-indexing principle.** The discrete input data must be stored and the discrete input files must be indexed to make them easily retrievable

Thesaurus. In a conventional IR system, a thesaurus may be used. A thesaurus provides a grouping or classification of the terms used in a given topic area into categories (thesaurus classes) [Salt83]. The idea underlying a thesaurus is that the conceptually related terms (the primitives in information retrieval) should be aggre-

gated in some way, which realizes the following:

(CO) Concept principle. *Conceptually related terms should be organized together to form a concept.*

Query. In order to answer a query with information needs, two processes have been identified in the Deutsch-Kraft model: intelligent access (analysis of queries and search of the index) and physical access (obtain the relevant records and present them in a suitable way). From these two processes we can identify the following two principles.

(COA) Conceptual access principle. *An IR system must be able to locate relevant information by searching related concepts.*

(CTA) Content access principle. *An IR system must be able to present the contents of the documents in a way understandable to the user.*

1.4 Advantage of examing IR model from knowledge engineering

While the fundamental functions of the IR systems are to analyze, store and retrieve the discrete input units, the functions of a knowledge representation system, have been identified as knowledge acquisition, perception, and planning to act [Wood86]. From a knowledge engineering point of view, the classical IR model as summarized above, is both potentially adoptable and questionable. After all, the traditional document retrieval model does not consider the key issues in knowledge engineering, such as knowledge representation and retrieval of the contents. It concerns only plain storage rather than intelligent integration of the knowledge the documents convey. But, on the other hand, the IR model, a fruitful result of a proliferate

research area, does provide a good framework for some abilities that current knowledge engineering lacks, such as acquiring knowledge pieces and retrieval of these knowledge pieces. After the components ("parameters") of the classical IR model are properly re-interpreted, it is possible to use this model to carry out some tasks in knowledge engineering. For instance, if we use documents as an intermediate level to to start searching the memory (instead of direct search in the memory), then it may be helpful to deal with the problem of "how to generate and search a space of possible hypotheses without combinatorial explosion" [Wood86]. As another example, if we can develop some method to construct "facts" from some documents, then it may shed some light on another problem, i.e., "how to find the relationships between elements that have been identified and the roles they could play in larger percepts" [Wood86]. What is more, the integration of knowledge investigated in this dissertation also supports an important viewpoint, that is, to build a knowledge base incrementally [Ries84].

Since information retrieval is an area which has been well-explored, it is worthwhile to find the missing principles in the current model of information retrieval, and to establish a new model which will be beneficial for knowledge engineering.

1.5 Issues to be investigated in this work

In this work we develop a framework that deals with knowledge storage and retrieval. The scheme of query invoked memory reorganization is further demonstrated through a case study. Although much of the current work in the intersection of IR and AI considers the use of AI for IR to improve the task of retrieval, this dissertation addresses some topics in the reversed direction, a similar effort of applying

database concepts in AI was briefly discussed in [Brod86].

The following topics are to be covered in our investigation:

(1) Establish a model for knowledge integration. We develop a model that integrates acquired knowledge. We identify the principles needed (particularly from cognitive science) to develop such a model. The components of this model will be described.

(2) Explore the scheme of query-invoked memory reorganization. We will explore the scheme of query-invoked memory reorganization in detail. We will explain different types of this reorganization, from the simplest case of document retrieval, through more sophisticated fact retrieval, to structure generation associated with query-invoked memory reorganization.

(3) Inductive reasoning. Query invoked memory reorganization is related to inductive reasoning, particularly to new structure generation, which realizes a theory of analogical reasoning [Gent83a, Gent83b]. It has been argued, from the cognitive science viewpoint, that there is a theoretical necessity for using a computing machine which uses reasoning primitives of analogy as opposed to reasoning primitives of deduction in order to accomplish the real time integration of a human adultsized knowledge base into the semantic analysis process [Rout85]. In this dissertation we will show that our proposed model is suitable for performing "structure mapping," a theory of analogical reasoning as initiated by [Gent83a]. This kind of ability shows that our framework does provide a kind of predictive power, an important feature that systems like CYRUS are lacking. Due to the complexity of analogical reasoning, in this dissertation, we are not intending to develop a comprehensive theory of analogical reasoning. Instead, we point out how our framework can become a starting point of a new

direction of studying analogical reasoning, particularly, the aspect of analogical reasoning as extended knowledge retrieval will be investigated.

(4) Related IR and database issues. We have already addressed the relationship between our current work and the classical IR, and this relationship will be examined throughout this dissertation. In addition, in order to demonstrate our fundamental idea, knowledge should be processed into a formatted form. This also leads us to examine the relationships between our scheme and database systems, which deals with formatted data. Finally, there are some design issues that should be considered in an implementation of an experimental system.

So far as the methodology is concerned, we agree with Newell and Simon [Newe75] viewing computer science as an empirical inquiry, that is, developing scientific hypotheses and then seeking to verify them by empirical inquiry. We raise hypotheses, then build machines and programs as a way of discovering new phenomena and analyzing phenomena we already know about. These considerations also roughly match the following discipline in general system theory: first, we obtain a model descriptive of many specific algorithms in a structure sense. Based on this model, we then derive principles or rules to serve in constructing algorithms fitting the model [Jone83].

1.6 Chapter overview

In this introductory chapter we have discussed the purpose of our research, and provided an overview for the later chapters. Some assumptions and new principles are developed in Chapter 2 through a survey of related works, particularly from considerations of cognitive science.

The cognitive model that realizes these principles (to be given the name as COGMIR) is actually developed in Chapter 3. We start from an introductory example, then define this model. The scheme of query invoked memory reorganization on this model is further discussed in Chapter 4. Since this model is general in nature, we further consider some restrictions imposed on this model. This results in a working model, and will be referred to as COGMIR1, which provides a case study of the general model. Data structures for this restricted model as well as related manipulations are discussed in chapter 5. In Chapter 6 we further discuss query invoked new structure generation. Some design issues is discussed in Chapter 7. This dissertation is ended with Chapter 8, where we summarize our major contributions, the assumptions involved, the methodology, as well as future research. Two appendices are attached at the end of the work. The first is sample input and the second is an annotated output.

1.7 Remarks

The major contribution of this work will be summarized in Chapter 8. Here we briefly point out what we are not intending to do. This dissertation does *not* address some traditional topics in IR. In fact, the topics that intrigue us go far beyond the areas that have been traditionally tied with IR. Particularly true is the role of analogy; it is unlikely a librarian would tell a patron: "The book you requested which consists of a certain plot does not exist. May I write a book for you by analogy based on the information contained in the other books available in the library?"

Finally, we have the following remarks on the writing of this dissertation. In organizing the material, we assumed the potential readers may have different backgrounds. Some required background not provided in the core of computer science,

particularly that is related to Schank or his previous students work, is summarized in different places throughout this dissertation.

Although our purpose is not to provide a strict mathematical foundation in the discussed area, to make the discussion succinct, all the definitions and the discussions will take a mathematical form where possible. On the other hand, although we do not like to use diagrams, since they lack representation power, they will be used whenever they are helpful in illustrating our basic idea.

CHAPTER 2

RELATED WORKS

Some research efforts related to our work were briefly discussed in Chapter 1. Some other related works are surveyed below, and we will summarize our findings in the last section.

2.1 AI in document retrieval

There are already some AI work being done in IR. Some discussions can be found in [Smit76, Coop84]. These give us some hints on how to make the document retrieval system more intelligent.

Components of documents. As we pointed out in Chapter 1, in the classical IR model, any document is represented as a whole. An important step toward a more intelligent system is to consider the decomposition of documents into more fundamental units. Starting from the idea of representing text in terms of concepts or units of meaning, [Crof87] describes an approach to natural language processing for document retrieval. [Kwok86] has a theory to decompose a document into components. A document is viewed as consisting of components, each component being independent and can be judged as to its relevancy property. Kwok's purpose is to develop a theory of indexing that interprets the index term weighting schemes in the classical sense of information retrieval.

Thesaurus. The purpose of decomposition of documents is for proper indexing. On the other hand, to make indexing more flexible, a thesaurus is used to organize knowledge. The thesaurus can be used to broaden the terms to be searched. Since the

thesaurus is a place to incorporate background knowledge about the terms, many works aimed for more intelligent document information retrieval are related to a thesaurus. [DeJa86] recognized a thesaurus as a semantic network rather than as a simple list of words. This is an extremely important idea, because it implies that conceptually related terms can be organized to form a concept.

Indexing. Although there are some disadvantages (such as reducing I/O activity) associated with indexing, and although recent parallel query processing goes without any indexing at all [Stan86a, Stan86b, Ston87], the advantage of indexing [Dewi81] is still obvious. It permits a very small region of a database to be read from auxiliary memory, whereas when no index is available, the operation may need to retrieve the entire database [DeJo83]. Moreover, indexing can be knowledge-based [Tesk87].

Process through primitives. Kwok's work, as summarized before, concerns the components of each individual document. Taking this direction further, one can treat the *components* common to all the documents as the fundamental elements of the entire system, and it is possible to process the knowledge carried by different documents through some *primitives.* This is the common idea underlying many different schemes, although the details may vary. A computational system developed by [Alte85] uses a dictionary of 100 to 150 event/state concepts to construct representations of narrative text. It provides a theory which claims that representations of narrative text can be generated by a process of matching text against a dictionary of concepts, which are related by a small set of relation-types. It uses the organization of the concepts in the dictionary to organize the instances of event/state concepts which appear in the text. The primary importance of the representation produced by the

system is that it provides an internal structure of the text. It supports the process of interpreting the text. It provides a computational scheme to handle the text by analyzing it into structured coherent chunks of event descriptions. If we push this strategy to one extreme, the primitive may be *objects* as discussed in Chapter 1. This case is very close to the dictionary-based reminding as discussed by [Scha82]. Moreover, these objects can be organized to form concepts by using the scheme proposed by [DeJo83] just mentioned above. As we will see, this final consideration becomes the starting point of our approach.

To summarize these AI works in IR, it is idealizable to process the documents into smaller units or to use some kind of primitive structures, so that around these primitives the contents conveyed by the documents can be organized. Also, by properly grouping these primitives, the task of identifying the documents can be facilitated, that is, documents can be represented through some of these primitives. Although these additional structures are still not the "internal structures", which stand for the meaning, of the documents, as we will see later, the recent progress in IR does encourage this kind of new direction.

2.2 Query processing in IR and database systems

The query is another type of input in information systems. Both of the two types of informations systems -- database systems and information retrieval systems -- have developed rich techniques of query processing. In information retrieval, several query processing models have been developed [Kraf85]. Among them, the most relevant to our interest is the vector space model [Kraf85, Salt83], because it concerns the representation of a document. In the vector space model, each document, as well as

each query, is assumed to be represented as a vector of terms, and the task of retrieval is to compute the similarity between the query vector and various document vectors, from which a kind of RSV (retrieval status value) is computed to determine the relevance of each document to the query. The somewhat symmetrical treatment between texts and queries can also be found in [Cate87] and [Maud85]. In [Maud85], both documents and queries are parsed into abstracts, and the degree of matching is determined by breadth-first case frame matching.

On the other hand, database systems, particularly relational database systems, have developed much richer query processing schemes for formatted data, of which a particularly interesting one is the relational database [Date86, Ullm82]. Is there any way to extend the basic idea of database theory to non-formated data? An early attempt along this direction was [Find79]. [McGr81] tried to represent a "fact" (the contents of a document) as a tuple in a relational database, each taking a certain format; but the representation power of this scheme is very restricted. In later works, such as [Kolo83], the use of database is at a lower level to store data. Generally speaking, we cannot find a unified way to represent the contents of documents, since they are so diverse. But if we can develop a way to process the input knowledge so that the internal structure is well organized, then some basic ideas in database theory can be extended to knowledge engineering [Brod86, Abar87].

There are a rich set of manipulations that operate on the databases, and a much less sophisticated set for the manipulations on an IR system. It is desirable that some of the database manipulations can be extended to IR to make it more intelligent.

2.3 Meaningful insight from cognitive science

Although it has been argued that computerized information retrieval need not follow human information retrieval [Korf85], the discussion in Chapter 1 indicates that our ideal framework works in a manner much like a human being, so that it can be viewed by the human user as his "partner machine" [Koss85]. In this section we identify some important assumptions from considerations of cognitive science in order to form a new model.

We start our investigation from input. Documents are the input units that consist of data. To embody the concept of "document" into knowledge engineering, we will assume these cognitive documents are cognitively meaningful units, or mental models as already stated in Chapter 1.

Some meaningful insights of cognitive science are indicated in the cognitive architecture proposed by [Ande83], in which the role of human information retrieval is fully examined, which goes far beyond computerized information retrieval. Some assumptions suggested by [Ande83, Ande84] as well as some other psychologists, are useful for our purpose.

First of all, in order to facilitate the task of processing the documents, we will make the following assumption:

Limited capacity assumption: the amount of information the (human intelligent) system can handle at one time is limited [Bour82].

Based on this assumption, the size of the documents will be made small.

Although fact retrieval is a kind of "content-accessible" retrieval, it can be exam-

ined from another perspective. Recent research in cognitive science has produced several computerized memory models; the recent developments in fact retrieval may be viewed as a kind of investigation along this direction. It is safe to say that just as in computer science the default meaning of information retrieval refers to document retrieval, the default meaning of information retrieval in cognitive science is more closely related to fact retrieval. This examination is important for our task of intelligently utilizing the acquired knowledge. To enrich classical information retrieval by absorbing achievements in cognitive science, the system should be able to organize the contents of different documents to construct fact and then actually retrieve them.

What is more, cognitive science also suggests to view information retrieval and other intelligent behavior as a whole. The following assumption from cognitive science will be adopted:

> *Integrated process assumption:* intelligent information retrieval is done as an integrated intelligent process. All the higher cognitive processes, such as memory, problem solving, deduction and induction, are different manifestations of the same underlying system [Ande83].

According to this assumption, the role of query in information retrieval can be viewed more actively than that taken in the current IR systems. That is, we will treat the query as the invoker of a series of activities which involves intelligently utilizing the acquired knowledge. The process of solving the problem implied in the query can thus be treated as a kind of intelligent knowledge retrieval.

This later observation can be incorporated with the following assumption from cognitive science.

Biased attention assumption: cognitive processing is serial in many respects because only one goal can be attended to at a time. The current goal becomes a powerful source of activation, so pattern-matching is biased to match structures involving that goal [Ande83].

From this assumption we will restrict our system to deal with a task (either storage or retrieval) at a system level in a serial manner. It also supports our scheme of query invoked memory reorganization in that memory reorganization is query-biased.

A prevalent type of structure used in knowledge engineering is a rule-based system. The advantages of rule-based systems have been discussed by many authors, but critiques also exist both from software verification, validation and maintenance [Gold86], and from psychology [John83]. We agree with [Scha82] that in addition to the conventional approach in which "compiled knowledge" (such as rules) is used, there is an alternative which attempts to model the raw memory of the expert. But differing from Schank and his followers, instead of using lower level episodes as input, the documents in our model are assumed to be mental models, providing a more flexible way of utilizing the acquired knowledge. Nevertheless, we do not want to exclude the use of rules, although they are only used for assistance of operations.

2.4 Knowledge representation and knowledge integration

As we mentioned before, a document in the vector space model is represented by a vector [Salt84]. This kind of representation does not concern content retrieval as discussed in [Kolo83] and [Lebo85]. With the coming of knowledge engineering, the representation of the contents of the documents has been considered, mainly by researchers in AI and by some other researchers in fact retrieval. This is usually done

through a process which maps the documents into their internal structure. In addition to the use of a relational database, as mentioned before, other knowledge representation schemes exist. Mathematical logics such as predicate calculus, support general logical inference, but seldom provide organizational support for *grouping* facts so that the facts can be efficiently used [Tani87]. Due to the nature of the scientific knowledge to be investigated in this thesis, we will use the declarative representations with a slot-and-filler structure, as summarized in [Rich83], although none of these existing structures (such as semantic nets or frames) can be directly used. Semantic nets are general enough, and frames are good for representing complex objects. The hierarchical requirement in semantic nets is not particularly suitable for more general relationship between objects, but the net-like structure can be adopted. On the other hand, frames are good for single objects or single relationships, but are not powerful enough to represent knowledge as a whole. These considerations result in a representation scheme of a net-like structure as already mentioned in Chapter 1 with objects and relationships as their constructs, and each is frame-structured. By doing so we have partly realized a fundamental belief since Aristotle [Wata85] and extended one of the successful uses in the logic database design, where the semantics of databases is captured through entities and relationships [Chen76]. No matter what kind of representation scheme is used, there is a need for processing the original input into its internal structure.

Knowledge representation through a net-like structure and the use of objects were discussed by many authors in the last two decades [Quil69, Fahl79, Kobs84, Wass85, Wass87]. One of major differences between our work and these previous

works is that we do not focus on language comprehension or knowledge representation related to single sentences [Quil68]; rather, we are interested in organizing knowledge in a level higher than single sentences. A similar situation can be found in Prolog; knowledge organization in Prolog is realized as pattern manipulation, but a pattern-directed module is limited to a single Prolog clause [Brat86]. However, we are looking for an intermediate level where patterns are constructed from memory units (objects) and their relationships which are not restricted to any single clause in order to extend the representation power.

A recent release of [Ullm88], based on [Ullm82], has extended object-oriented database languages to knowledge-based systems. In addition, the issue of integration in a multiple database, as discussed recently by several authors [Metr87, Litw85], also shares some common concern of integrating knowledge through proper representation. The use of knowledge representation has made possible the synthesizing approach of investigating "part and whole" through symbolic manipulation, which is complementary to the traditional analytical approach to numerical data [Klir85, Jone85].

The power of a knowledge representation scheme also lies in the the ability of inference on the knowledge represented by it. The ability to update the acquired knowledge invoked by queries, as discussed in Chapter 1, implies a kind of inference. In a computer model that incorporates principles of cognitive science, induction reasoning [Gent87] plays an important role. Particularly true is a special kind of inductive inference that is related to analogical reasoning [Gent83a, Gent83b, Forb86, Gick80, Gick83, Wins80, Wins84, Klin71]. Previously, the research in analogical

reasoning and in information retrieval have been separate, although the relationship between these two directions has been addressed occasionally. The importance of analogical reasoning in knowledge engineering has also been noticed [Elio86]. It is generally agreed that analogical reasoning concerns a kind of mapping; the final solution or the intermediate steps for an old problem can be mapped to a new problem [Carb85], or some abstract schema for an old situation can be mapped to a new situation [Holy83]. We agree to view analogical reasoning as structure mapping [Gent83]; therefore, in order to provide inference ability such as analogical reasoning, the net-like structure provided by our model should be designed to facilitate this kind of mapping.

2.5 Summary of the finding

To summarize our finding, all the important components of the classical IR model can be adapted to the purpose of knowledge engineering, although some of its components should be re-interpreted. We have also seen many important principles for enhancing the current IR model to meet the requirements of knowledge engineering. Particularly, we have discussed the task of integration. After the documents are integrated into an overall knowledge space, to retrieve them is not only to identify their names (or some ID), but also to actually identify their existence in the knowledge space and to extract the related contents, or approximatedly reconstruct the documents.

Integrating the internal structure of the documents adds additional tasks to manipulation, but it also provides a way of controlling the use of the contents acquired from the documents. New structures can be generated, which is a topic meaningful to

knowledge engineering and is a topic that cannot be covered by the traditional IR model. A novel memory manipulation scheme, called query invoked memory reorganization, is suggested to reach these goals.

New assumptions adapted from cognitive science are:

Limited capacity assumption;

Integrated process assumption; and

Biased attention assumption.

New principles for realizing a computer model so that this scheme can work have been identified from the literature. We have found that those considerations related to memory organization from cognitive science are particularly helpful.

To summarize the additional principles found in this chapter, along with the IR principles identified in Chapter 1, we have the following principles to develop a model for knowledge engineering. Principles started with a * indicate that they are adopted from the classical IR model, principles started with a ** indicate that they are extended from the classical IR model, and principles without these symbols indicate they are new principles implied by the previous sections.

* **DI** -- *Discrete input principle.* The input to the system should be discrete units.

MP -- *Mapping principle.* There must be a way to map the documents into internal structure.

** **KR** -- *Knowledge representation principle.* Knowledge carried from documents must be represented in a proper way. (Extended from **DR** principle).

** **ID** -- *Integration and distinction principle.* The contents of documents should be tied together in a proper way while the documents should still be distinguishable. (Extended from **ID** principle).

FC -- *Fact construction principle.* It should be able to organize contents from the different documents to construct facts and actually retrieve them.

* **CO** -- *Concept principle.* It should be able to identify conceptually related primitives.

SC -- *Structure comparison principle.* The structure of the internal representation must be comparable.

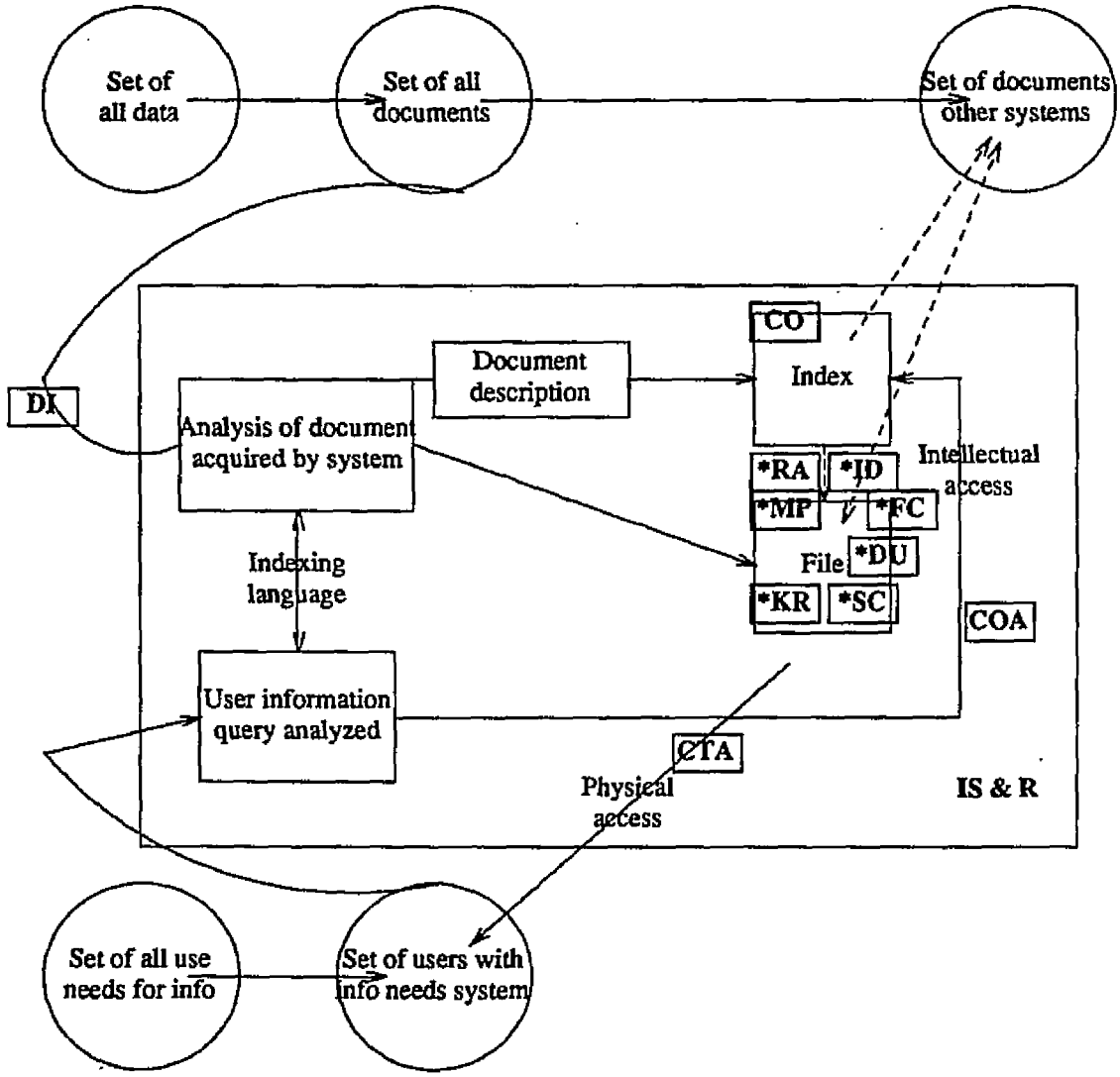
DU -- *Dynamic updating principle.* The structure of the knowledge base should be dynamically updatable through query-invoked memory reorganization.

RA -- *Rule-aiding principle.* It is desirable to use a rule base to aid the manipulation on the knowledge base.

* **COA** -- *Conceptual access principle.* An IR system must be able to locate relevant information by searching related concepts.

* **CTA** -- *Content access principle.* An IR system must be able to present the contents of the documents in a way understandable to the user.

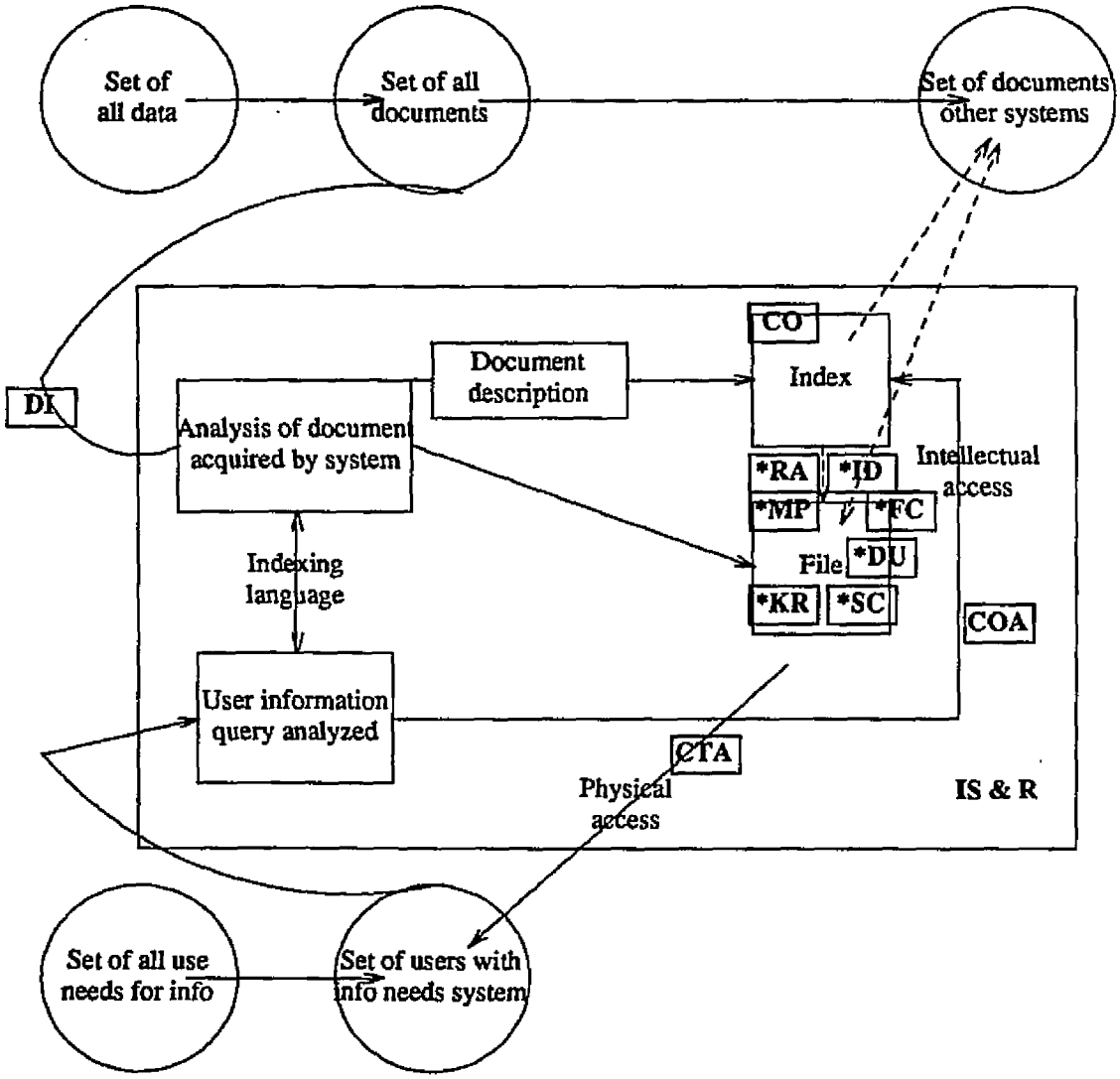
Figure 2.1 shows the original Deutsch-Kraft model of information retrieval with new principles added. All the names of the principles are in places close to the components with which they are involved. The new model for the needs of knowledge engineering will be discussed in the next chapter.



*NOTE: * indicates new principle (or extended principle)*

DI: *Discrete input principle* **CO:** *Concept principle* ***ID:** Integration and distinction principle*
COA: *Conceptual access principle* **CTA:** *Content access principle*
***MP:** Mapping principle* ***KR:** Knowledge representation principle*
***FC:** Fact construction principle* ***DU:** Dynamic updating principle*
***SC:** Structure comparison principle* ***RA:** Rule aiding principle*

Fig. 2.1 The Deutsch-Kraft model with new principles added



*NOTE: * indicates new principle (or extended principle)*

DI: *Discrete input principle* **CO:** *Concept principle* ***ID:** Integration and distinction principle*
COA: *Conceptual access principle* **CTA:** *Content access principle*
***MP:** Mapping principle* ***KR:** Knowledge representation principle*
***FC:** Fact construction principle* ***DU:** Dynamic updating principle*
***SC:** Structure comparison principle* ***RA:** Rule aiding principle*

Fig. 2.1 The Deutsch-Kraft model with new principles added

CHAPTER 3

COGMIR: A COGNITIVE MODEL FOR INTELLIGENT RETRIEVAL

In this chapter we propose a model which satisfies the principles identified in chapter 2. This model will be referred to as COGMIR, which stands for a COGnitive Model for Intelligent Retrieval. We are interested in practical issues, but in order to provide a model we can actually follow, we will formalize it. Based on the model defined in this chapter, we will further describe the query invoked memory reorganization in the next chapter.

3.1 An introductory example

In this section we give an introductory example to explain our approach of realizing integration of scientific knowledge. We also intuitively introduce some terminologies which will be defined in later sections.

Suppose the following three documents d_1, d_2, d_3 are acquired by the system. These documents are similar to those in the abstracts in Chapter 1, but are written according to a certain grammar to simplify the task of processing into internal structure (to be discussed in the case study in Chapter 5). This kind of restriction is not required by the model itself; the basic idea demonstrated here can be extended to more complex situations.

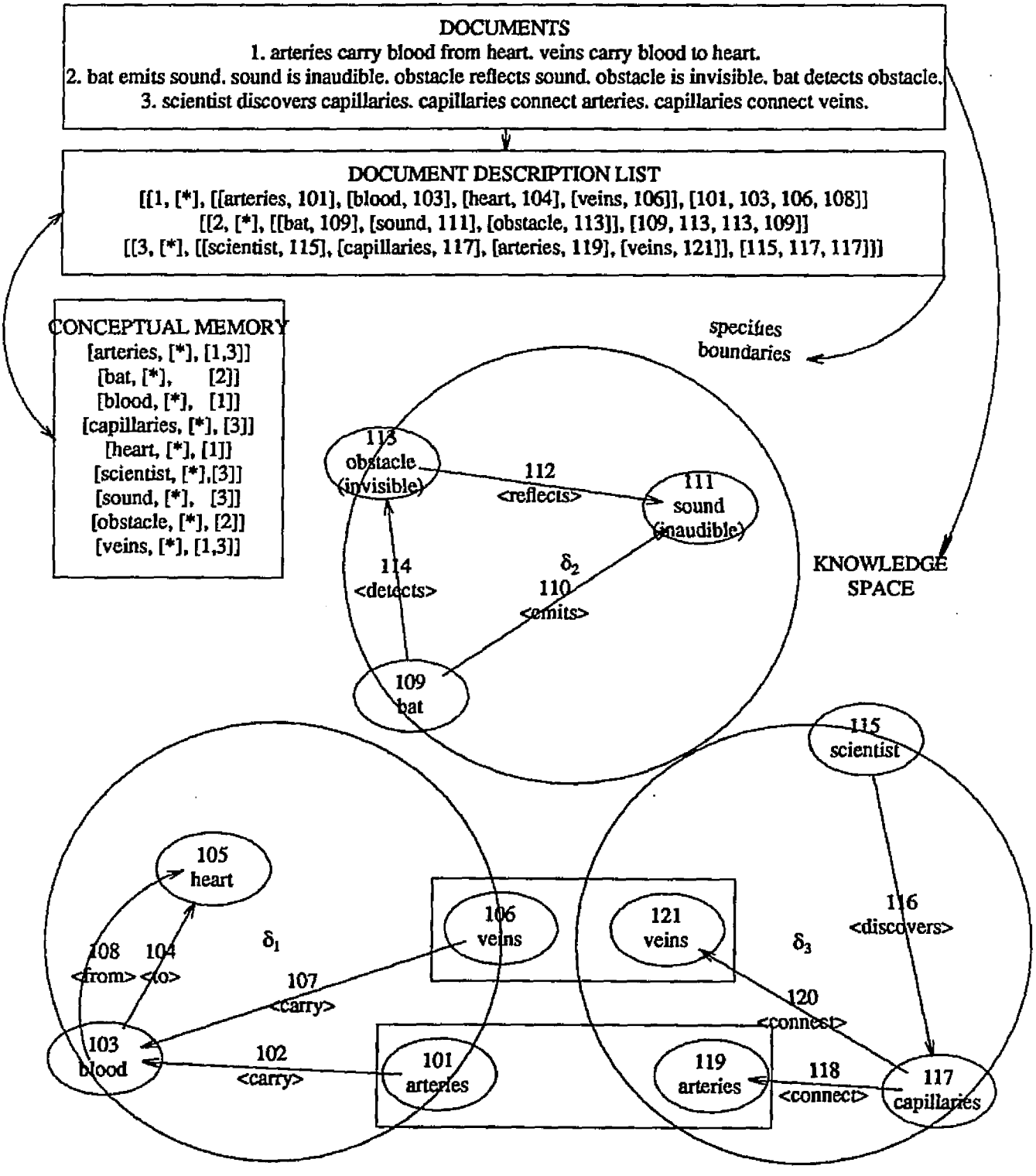
1. *the arteries carry blood from the heart. the veins carry blood to the heart.*

2. *bats emit sound. the sound is inaudible. the bat approaches an obstacle. the obstacle reflects the sound. the obstacle is invisible. the bat detects the obstacle.*

3. *A scientist discovers capillaries. the capillaries connect arteries. the capillaries connect the veins.*

We use Figure 3.1 to explain what happens in the system, where the major components of a model, along with original documents, and their stored internal structures are shown. The figure is only used to illustrate the function of some of the most important components in our proposed model.

Notice that in Figure 3.1 there are two places related to the contents of the documents: the input documents are shown at the top of the figure, but they are not actually stored like that. The actual place to store the knowledge is the knowledge space, the net structure appeared at the right part of the figure, where objects are represented as small ellipses, while relationships are represented as arcs between these ellipses. Each object or relationship is assigned a number to indicate its location in the knowledge space. The big circles indicate the boundaries of the document stems (but, remember, the information about these boundaries are actually stored in the document description list, not in the knowledge space). Another two components in the figure are: document description list (which records information about each document) and conceptual memory (which extends the role of indexing). The two small rectangles indicate the possible connections recorded in the conceptual memory. (In the case study, lists, instead of the figures shown here, are used).



Note: attributes for objects in parenthesis

Figure 3.1 An introductory example

In order to explain how the model works, we start out using familiar terminology in classical information retrieval. Based on some algorithms which will be explained later, each document will be represented by an item in the *document description list.* The document description list records the general information related to each document (such as the identifier of the document), thus bearing some similarity with the "vectors" in the vector space model.

The similarity between the classical IR system and our approach stops here. Unlike the classical IR systems, which store the files word by word, for intelligent systems in which an internal structure is used, the knowledge carried by the documents can be represented as a net-like structure. This is done by using a processor to map a document into its internal structure (named *document stem),* and the various document stems are tied together through the conceptual memory. Each document stem occupies a certain area of the knowledge space, and is confined in the boundary which is imposed by the document description list. In Figure 3.1, all the objects in the boundary are the ellipses which appear on the circle (such as "veins" at location 106). Moreover, conceptual memory is associated with the document description list by recording the identification of related documents, and the document description list is further connected with the net-like structure; therefore, the conceptual memory is able to do inference on the net-like structure.

To illustrate manipulations on the stored knowledge, suppose the query "heart, capillaries" is given. A fact constructed for this query can be constructed and the query can be answered as follows:

the arteries carry blood to the heart. the veins carry blood from the heart. the capillaries connect the arteries. the capillaries connect the veins.

This answer is none of the three input documents. How could this new text be generated? The conceptual memory records *possible connections* about a certain object name that has more than one object scattered in different documents. In Figure 3.1 these possible connections are indicated by rectangles. When a query comes, it *invokes* a process so that the contents appearing in different documents but relevant to the same object will be *integrated.* In this particular example of query, only part of d_1 and d_3 are used to construct a fact; these two involved documents form a *document cover,* or *D-cover* of the fact, and the objects with possible connections will be virtually combined as if they were the same object at this query time.

But what can a classical IR model do for this example? It can only identify that documents 1 and 3 are partially relevant to the query, but it does not know how these two documents are related to the answer, because this is left to the user. However, an intelligent retrieval system can handle this.

What is more, based on the structures contained in the contents of the document, the system is able to carry out a kind of inductive reasoning. Suppose a person wants to retrieve some knowledge about how to detect an enemy's plane. He may type in some words to form a query description list, but nothing can be retrieved, because there is no relevant knowledge previously acquired. In this case, a conventional IR system will just stop here. But COGMIR can do something for the user. That is, the user may submit a query which takes the form of an "incomplete" document (it takes the form of a document but contains an unknown part, and in the case study in

Chapter 5 it will be indicated "how"), and requests the following:

people live in a city. enemies dispatch a plane. the plane is invisible. the plane is flying. the plane approaches the city. how the people detect the plane.

This incomplete document describes a situation and implies a problem to be solved. Since there is no previously stored knowledge available to answer this query, COGMIR will try to generate an answer based on current available knowledge through simple analogical reasoning. The following is a possible answer provided by the system:

people live in a city. enemies dispatch a plane. the plane is invisible. the plane is flying. the plane approaches the city. people **emit sound-like. sound-like is inaudible. the plane reflects sound-like.** *people detect the plane.*

Apparently the term "sound-like" is the alias for "radar". So this answer suggests people use some inaudible thing like sound, and let it be reflected from the plane to detect the plane. This answer can be generated, because the knowledge of document 2 is used for *analogical reasoning* to *generate* a kind of suggestion when requested knowledge is not available.

These examples illustrate the way COGMIR deals with the key problem as stated in Chapter 1. In the following sections, the COGMIR model will be established step by step.

3.2 Some fundamental ideas of a framework for intelligent retrieval

How do we develop a model to realize the principles identified in Chapter 2 in the way illustrated in the introductory example? The following are some considera-

tions.

Two kinds of representation for a document.

From the integration-distinguishing principle and the knowledge representation principle, we consider ways of representing the documents. Each document has two representations. There is a component in the model, the *document description list,* which contains rough representations of each document. This is the extension of the "vector" in the vector space model, but is much enriched. Each document also has an internal representation, the *document stem,* which captures the semantics of the document. All the document stems are stored in a common area, a net-like structured *knowledge space.* Each document stem occupies a certain area of this overall knowledge space. The rough representation of the document contains the information about the *boundary* of this area. The existence of two representations facilitates the tasks of retrieval and integration. Operations can be defined on the document stems to form a new document stem, therefore, operations on these document stems play the role of symbolic pattern manipulations as usually discussed in artificial intelligence.

Mapping documents into the knowledge space.

To satisfy the mapping principle, a mapping mechanism must be provided. The nature of the mapping mechanism is a processor which processes the input into its internal structure, which consists of objects and relationships, each occupying a memory location. Consequently, there are two inversely related problems: the map from input to internal structure for storage and the inverse map from the internal structure for retrieval. A third mapping was also illustrated by the introductory example, in which an "incomplete" document was mapped into its internal form. These

considerations also support the principles (FC), (SC), (DU), and (CTA).

D-cover and fact construction.

To satisfy the fact constructing principle, we point out under the previous described knowledge representation how to construct a fact from the documents to answer a query. As we said before, documents and facts are basically the same, except that a document must be an input unit, while a fact usually is constructed from such input units. The documents used to construct a fact form a **D-cover** of the fact. The task of fact retrieval is thus to find the D-cover and then actually construct the fact from the D-cover.

Conceptual memory ties things together.

Document stems constructed from different documents are not connected together, they occupy separate areas in the knowledge space. In order to integrate knowledge, these disconnected document stems must be tied together in certain ways. This is realized through conceptual memory. And this function of conceptual memory is extended from indexing in classical IR, different documents are tied together through common terms from which they are indexed. But classical IR takes documents as units (the documents that involve the same term are tied together), while for COGMIR, *through the document description list,* the *contents* of the documents that are relevant to the same concept are tied together. Particularly, the objects with the same name but are scattered in different document stems can be collected to form an object class. In a sense, conceptual memory provides a way to organize domain related knowledge. In addition, "if ... then" type rules can be used to deal with the problem of same word having different meanings.

Reference range of concepts.

Another function of conceptual memory is indexing. To provide a good, non-conventional indexing scheme, the following observation makes sense. Associated with a concept are some objects (in fact, concepts may be chosen from object names). Each object in the knowledge space is at the *reference range* (i.e., conceptually close enough) of a concept, so that the task of searching an object may be replaced by searching a related concept. For instance, if the only objects in the knowledge space are o_1, o_2, o_3, and if concept c_1 has reference range o_2, o_3 and concept c_2 has reference range o_1, then the set of concepts $\{c_1, c_2\}$ is qualified to form conceptual memory, because all the objects (i.e., o_1, o_2, o_3) can be referred from this set. Apparently, one role of conceptual memory is the combination of the index and the thesaurus in the classical IR model. There may exist different strategies to operate on the conceptual memory.

Requirements for the conceptual memory.

To summarize the discussion about the conceptual memory, it should satisfy the following requirements: (1) it groups conceptually related objects into concepts; (2) it should provide the information to locate the related area in the knowledge space by identifying proper items in the document description list; (3) it should provide the way to satisfy the (COA) principle.

The following is an example of part of the conceptual memory, although the actual form may vary in different implementation. Suppose documents d_6 and d_9 are relevant to "doctor" and document d_8 is relevant to "engineer". If the concept "intellectual" takes "doctor, engineer" as its reference range, then documents d_6, d_8, d_9, will

all be relevant to the concept "intellectual". In addition, to facilitate search (although this requires more space), some or all of the objects can be made directly accessible in the conceptual memory, even though these objects themselves are not concepts. In a case study in Chapters 5 and 6, we will allow each concept to be directly accessible, and using the format defined there, the part of conceptual memory can be written as

[intellectual, [doctor, engineer], [d_6, d_8, d_9]],

[doctor, [], [d_6, d_9]],

[engineer, [], [d_8]]].

Here we assume that "doctor" and "engineer" cannot refer any objects other than themselves, therefore their reference ranges are represented by empty lists [].

Summary of the considerations.

These considerations result in a model shown in Figure 3.2, The introductory example given in the beginning of this chapter is based on this model. In the rest of chapter, we are going to define the components of this model.

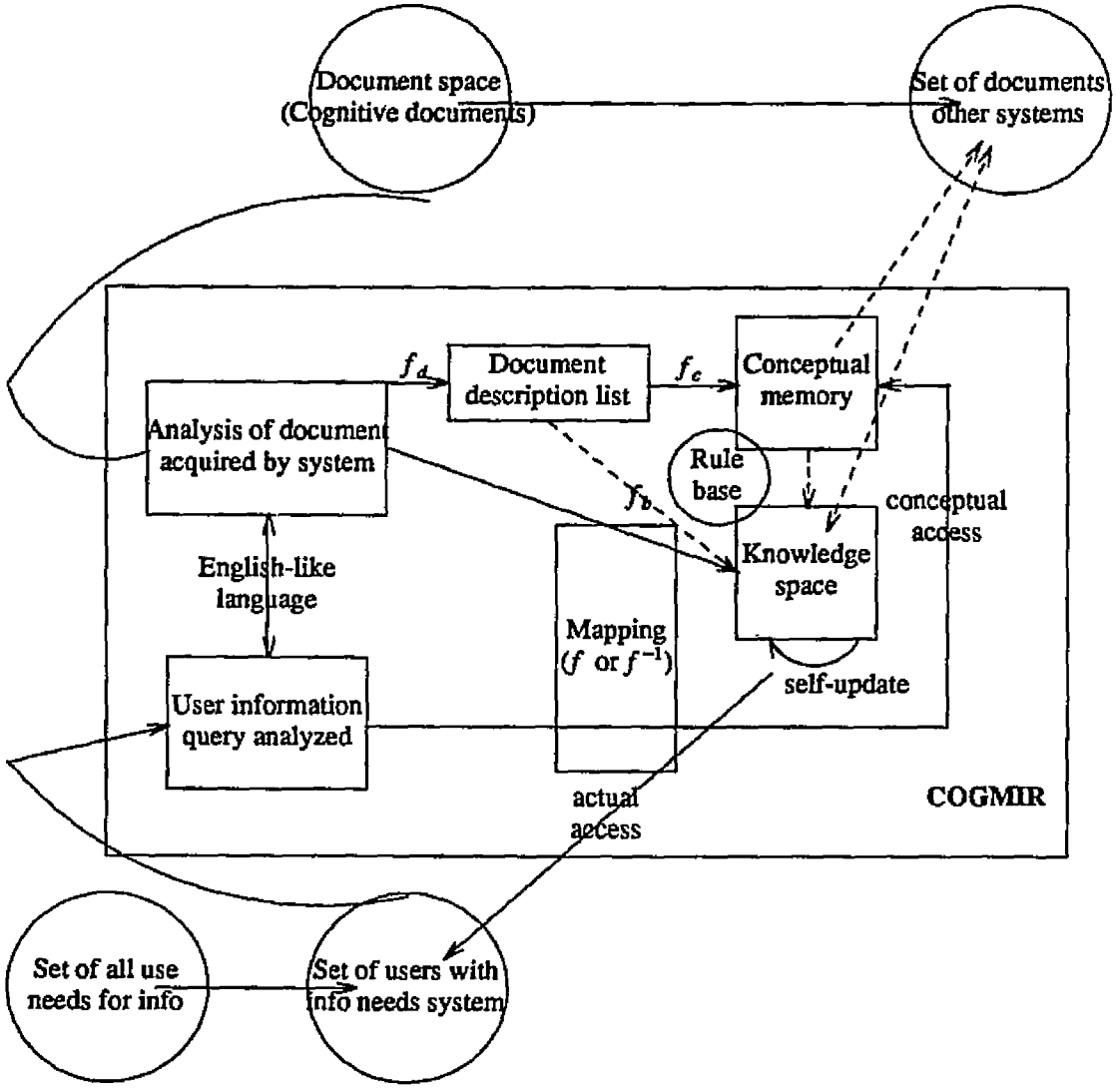


Figure 3.2 The COGMIR model

3.3 Document space and knowledge space

To start defining our model, we will use the term **cognitive document** (or simply document) to refer to any cognitively meaningful unit which takes a specific form and can be processed.

We briefly explain what "cognitively meaningful" connotes. Since our model deals with knowledge and since we adopt the assumption that knowledge is compressed information [Mich85], it is reasonable to assume that the input pieces can be treated as mental models which reflect the way a person views the physical world. For instance, a concise description of a machine, a concise description of a process, as well as the paragraph-like short knowledge pieces previously shown in the introductory example, are all examples of mental models. These mental models will be treated as cognitive documents.

Definition (Cognitive document space, or Document space). The document space (denoted **D**) is a set of cognitive documents.

Notice that the knowledge space, which will be discussed in detail below, is different from document space itself. These documents are not actually stored as such, and no operations will be defined directly on the documents. However, the term "document space" facilitates user imagination and also coordinates our work with classical information retrieval. The knowledge conveyed by the documents is actually stored in the knowledge space. In order to define the knowledge space, we will use two kinds of constructs: objects and relationships between these objects.

Definition (Object). An object is a non-decomposable unit that carries knowledge which at least contains its name and its location identifier (a number), and can be

self-structured. All the occurrences of an object in a certain document will be treated as a single object. The same object name may have one or more objects in different documents. In principle, we do not specify the actual form of the objects.

Definition (Relationship). A relationship specifies an ordered pair of objects

$$(o_1, o_2).$$

It can have attributes of its own, and must have a name and location identifier (a number). Just like objects, the actual form of relationships may vary.

Definition (Knowledge Space). The knowledge space (denoted **K**) is a bi-tuple (O, R), where O is a set of objects $o_1, ..., o_n$, and R is a set of relationships $r_{ij} = (o_i, o_j)$, $i, j = 1, ..., n, i \neq j$.

This definition indicates that knowledge space is a net-like structure, with objects as nodes associated through relationships.

To describe the internal structure of a document, we give the following definition.

Definition (Document stem). A document stem δ is a part of the knowledge space which consists of some objects and some relationships betweens these objects, i. e., $(O, R(O))$, where O stands for a set of objects, and $R(O)$ stands for the relationships.

The document stem is used to describe, from a knowledge space point of view, what a document is. The document stem is the form in which any document is actually stored. Each document has its correspondent document stem; moreover, as we will seen in section 3.4, operations can be defined on these document stems to construct new document stems. The term "document stem" is named for this reason: operating

on document stems may result in something new which is similar to a document.

A document, if its document stem involves objects $o_1, ..., o_n$, will be referred to as $d(o_1, ..., o_n)$.

Definition (Area and Boundary). A document stem is also called an area of the knowledge space. The boundary of this area is the set of objects from which the whole area can be accessed.

Definition (Interior). The interior of a document stem contains all the objects of that document stem that are not on the boundary.

We have some observations on the relationship between documents. Two documents, written in different ways, may carry the same knowledge. That is, they have the same document stem. It is reasonable to treat them as equivalent. Therefore, we need the following definition (Note: this definition will be slightly revised later).

Definition (Document equivalence). Two documents d_1 and d_2 are equivalent, if have same representation in the knowledge space.

In order to record information for the documents, we define document description list.

Definition (Document description list). The document description list, or **Dd**, contains the document reference number d_i for each document, along with the general information about each document, and the information about the boundary of its document stem in the knowledge space.

In order to describe the relationship between the document description list and the document space **D** and the knowledge space **K**, we use the following two func-

tions. There is a function

$$f_d: \quad \mathbf{D} \rightarrow \mathbf{Dd},$$

which maps each document into its correspondent item in the document description list **Dd.** The result of this mapping is the creation of a new item recording the general information of a document, such as document identifier, and several concepts involved in this document. On the other hand, the boundary information in the document description list is obtained through the function f_b^{-1}, which is the inverse of the following function:

$$f_b: \quad \mathbf{Dd} \rightarrow \mathbf{K}.$$

As we will see later, f_b will be used for retrieval. In order to retrieve a document, the boundary information stored in the document description list will be mapped back to the knowledge space **K** so that an area in the knowledge space can be identified.

Here we will not specify the actual form of these two functions, the actual form may vary. Nevertheless, we may point out that in the previous introductory example, the arrow from the document space to the document description list indicates function f_d, while the arrow from document description list to knowledge space indicates function f_b.

Now we consider the relationship between the document space and the relationship space.

Definition (Mapping from D to K). A mapping f from **D** to **K** is a process of converting any document $d \in \mathbf{D}$ into its internal representation $\delta \in \mathbf{K}$, denoted

$$f: \quad \mathbf{D} \rightarrow \mathbf{K},$$

or

$$\mathbf{K} = f(\mathbf{D}).$$

The information about the boundary of the document stem can also be obtained when a document is mapped into its correspondent document stem. This boundary information is then mapped into document description list by function f_b^{-1} (as discussed before).

Definition (Inverse of mapping). The inverse of the mapping f, denoted f^{-1}, is defined as

$$f^{-1}: \qquad \mathbf{K} \rightarrow \mathbf{D},$$

or

$$\mathbf{D} = f^{-1}(\mathbf{K})$$

in the following sense: when $f^{-1}(\delta))$ is treated as a document, where $\delta \in \mathbf{K}$, then

$$f(f^{-1}(\delta)) = \delta.$$

Notice the definitions of f and f^{-1} are not symmetrical. This is because the mapping from **D** to **K** must take documents as units while when mapping back from **K** to **D**, what is obtained is not necessarily a previously stored document. This is the result of the manipulations on the document stems. Manipulations on the document stems may result in new documents, which are not created from any single document. The existence of these documents (will be defined as *facts* in the next section) is an important feature of COGMIR.

Definition (Document reconstruction). If δ is the document stem for document d, the process of obtaining $d' = f^{-1}(\delta)$, where d is equivalent to d', is called the reconstruction of d.

One important reason of introducing the term "document equivalence" is due to the needs of document reconstruction. The reconstructed document need not be necessarily identical to the original one, if they carry the same meaning.

3.4 Manipulations in the knowledge space

Operations on the internal structure of the documents embodies grouping some related items together in the knowledge space according to certain criteria. We already know that the internal representation of a document takes the form of a document stem. Therefore, each document has its correspondent document stem. Moreover, in the following we will define operations on these document stems, therefore, a new document stem can be constructed from existing document stems. Thus a document stem does not always necessarily stand for the internal structure of an original document. Rather, it may also be used to construct facts.

Apparently, any document should have its corresponding document stem. But the inverse is not true. On the contrary, the reason to introduce the concept of document stem is to introduce operations on the original document stems so that new document stems can be formed, and thus, in turn, facts can be constructed. In the following we are going to discuss the join operation on the document stems, but first we will introduce the following definition.

Definition (Inclusion of document stem). A document stem δ_1 is included in document stem δ_2, denoted $\delta_1 \subset \delta_2$, if any object o in δ_1 is also an object in δ_2, and any relationship r in d_1 is also a relationship in d_2.

Definition (Union of document stems). The union of document stems $\delta_1, \delta_2, ..., \delta_n$,

denoted $\bigcup_{i=1}^{n} \delta_i$, is also a document stem, with all the objects in δ_i as its objects, and all the relationships in δ_i as its relationships. These objects and relationships may be ordered.

Definition (Partial union of documents, or fact). A partial union of documents $d_1, d_2, ..., d_n$, or a fact F, is defined as

$$F = f^{-1}(\Delta),$$

where $\Delta = \bigcup_{i=1}^{n} \delta_i^*$, $\delta_i^* \subset \delta_i$, and $f(d_i) = \delta_i$, for all $i = 1, ..., n$, where f is the mapping from D to K.

Definition (Union of documents). The union of documents $d_1 \cup d_2 \cdots \cup d_n$ is defined as

$$d_1 \cup d_2 \cdots \cup d_n = f^{-1}(\delta_1 \cup \cdots \delta_n),$$

where $\delta_i = f(d_i)$, and f is the mapping function from **D** to **K.**

Notice that we define the union of documents through document stems instead of defining it directly. This is to make it consistent to our basic idea that the internal structure of the documents, rather than the documents themselves, are the things on which operations will occur.

Remember that we already assumed that input documents are mental models, the union of documents implies that the mental models can be made more complete.

Definition (D-cover of fact, Minimum D-cover). A D-cover D of a fact is a set of documents from whom this fact is formed. A D-cover D is a minimum D-cover of fact F if when any document $d \in D$ is removed from D, then D is no longer a D-cover for that fact.

Since usually the minimum D-cover is our major concern, unless specially mentioned, we will use the term D-cover to refer the minimum D-cover. From the definition of the term "fact", it is obvious that for any fact, the D-cover as well as the minimum D-cover exists, although they may be not unique.

In order to avoid possible confusion, we will use lower case English letters and their correspondent Greek letters to denote documents and their document stems, respectively; we will use upper case English letters and their correspondent Greek letters to denote the D-cover of a fact (or the fact itself) and the document stems from which it is constructed, respectively. If we do not care whether we are considering documents or not, we will use upper case letters.

The definition of the mapping from **D** to **K** only concerns one document. But, to construct a fact several documents may be involved. To facilitate the discussion, we give the following definition.

Definition (Mapping of the D-cover). The mapping of the D-cover $D = (d_1, ..., d_n)$ for some fact F is defined as

$$f(D) = \bigcup_{i=1}^{n} f(d_i).$$

where f is the mapping function from **D** to **K.**

The inverse of this mapping is already included in the definition of function f^{-1}.

3.5 Structure mapping and temporary area

We introduce another kind of mapping, which concerns two document stems.

Definition (Structure mapping between document stems). The structure mapping between document stems δ_1 and δ_2, denoted $\delta_1(S_1) = \Phi(\delta_2(S_2))$ exists, so does Φ^{-1}, if there

exists a one-to-one correspondence between S_1 and S_2, where S_1 and S_2 are two subsets of objects and relationships of d_1 and d_2, respectively.

Definition (Equivalence of document stems). A document stem δ_1 is equivalent to another document stem δ_2, if there is a structure mapping Φ so that $\delta_1 = \Phi(\delta_2)$ where Φ always maps an object in δ_1 into another object in δ_2 which are identical except for memory locations, and maps a relationship in δ_1 into another relationship in δ_2 which are identical except for memory locations. These two involved document stems δ_1 and δ_2 will then be called equivalent to each other.

Because of this definition, the previous definition of document equivalence can be extended to the following: two documents are equivalent if their internal representations (i.e., their corresponding document stems) are equivalent.

Generally speaking, the concept of equivalence of document stems is not particularly interesting, because there is no need to store equivalent knowledge more than once, even from different documents. However, this concept is useful when it is applied to the structure mapping between the knowledge space **K** and a special area of **K**, which is defined below.

Definition (Temporary area of knowledge space). The temporary area of the knowledge space is an area separate from the rest of knowledge space, and contains objects and relationships that are not converted from acquired documents.

The temporary area plays the role of short term memory, while the knowledge space plays the role of long term memory. The temporary area is the place where the knowledge space *self-updating* occurs. That is, the system is able to update its

knowledge space by using its available knowledge. The topic of using the temporary area will be explored in Chapter 6.

3.6 Conceptual memory

We will first define a single concept, then define conceptual memory.

Definition (Concept, reference range). A concept contains a name (usually an object name), its **reference range,** which is a set of object names conceptually related to it, and the documents relevant to it or the objects in its reference range. The reference range for concept c_i is denoted $R_c(c_i)$.

Moreover, for ease of use, in the following, we will use the reference range for a set of concepts $c_1, ..., c_n$, which is defined as

$$R_c(c_1, ..., c_n) = R_c(c_1) \cup ... \cup R_c(c_n),$$

where the symbol $\cup$ denotes the conventional union operation on the sets $R_c(c_i), i = 1, ..., n$.

Definition (Conceptual memory). Conceptual memory is a set of concepts determined in such a way so that their reference range contains all the objects in the knowledge space.

The function of conceptual memory facilitates retrieval. Generally speaking, the number of concepts used in conceptual memory is inversely related to the width of reference ranges of concepts. There is a trade-off between time for searching in the conceptual memory (when performing retrieval) and the size of conceptual memory. The actual implementation of conceptual memory may vary, depending on different needs.

The definition of the conceptual memory indicates that it serves as an interaction between the document space **D** and the knowledge space **K.** Therefore, we need the following definitions.

Definition (Document identification function). The document identification function f_c is defined as

$$f_c: \qquad d_i \rightarrow C',$$

where C' is a set of concepts in the conceptual memory and d_i stands for a document. Here d_i can be called as the document *associated* with concept c for any $c \in C'$.

Notice that this mapping is not a one-to-one mapping between a concept and a document. For instance, for a concept "people", there may be several documents related to it, such as documents d_3, d_4, d_{22} (assume documents are identified by a pre-assigned number). On the other hand, a document may be related to several concepts; for instance, document d_3 may be related to concepts "people, machine", d_4 may be related to concepts "heart, vein", and d_{22} may be related to "people". Therefore, we need the following definition.

Definition (Inverse mapping of document identification function). The inverse of the document identification function f_c^{-1} is defined as

$$f_c^{-1}: \qquad c_j \rightarrow D_j,$$

where c_j is a concept in conceptual memory, D_j is the set of the documents associated with c_j.

The relationship between the document identification function and the inverse document identification function is indicated below.

Theorem 3.1 If

$$C_0 = f_c(d_i),$$

then for any $c \in C_0$,

$$d_i \in f_c^{-1}(c).$$

Proof. According to the definition of the document identification function, d_i is associated with c as stated in the theorem. Therefore, $d \in D$, where D is the set of documents as stated in the definition of the inverse document identification function. []

For instance, in the previous example, for $i = 3$, $C = \{\text{people, machine}\}$, f_c^{-1} (people) $= \{d_3, d_{44}\}$, f_c^{-1} (machine) $= \{d_3\}$. Apparently $d_3 \in f_c^{-1}$(people) and $d_3 \in f_c^{-1}$ (machine).

So far we have defined four functions: f_b, f_d, f_c, and f. Function f deals with the contents of the documents, while all the other functions are concerned with the general information of the documents (such as document identifier). These functions can work together to perform tasks of storage and retrieval. The functions of f, f_c, f_b and f_d, as well as most components are summarized in Figure 3.3. The role of these functions in storage of a document is exemplified in Figure 3.4, which illustrates a possible way in which a document $d(o_1, o_2)$ is stored, where $o_k \in R_c(c_{i_k})$, where $k = 1$, 2.

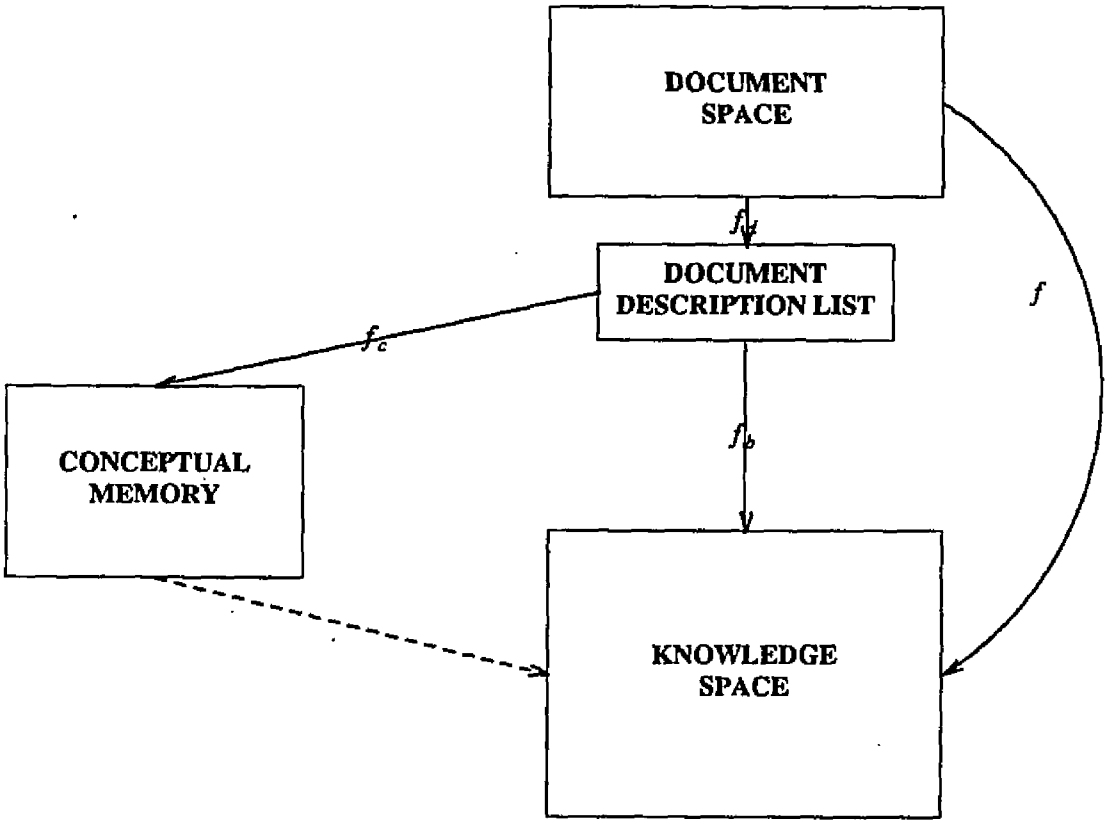
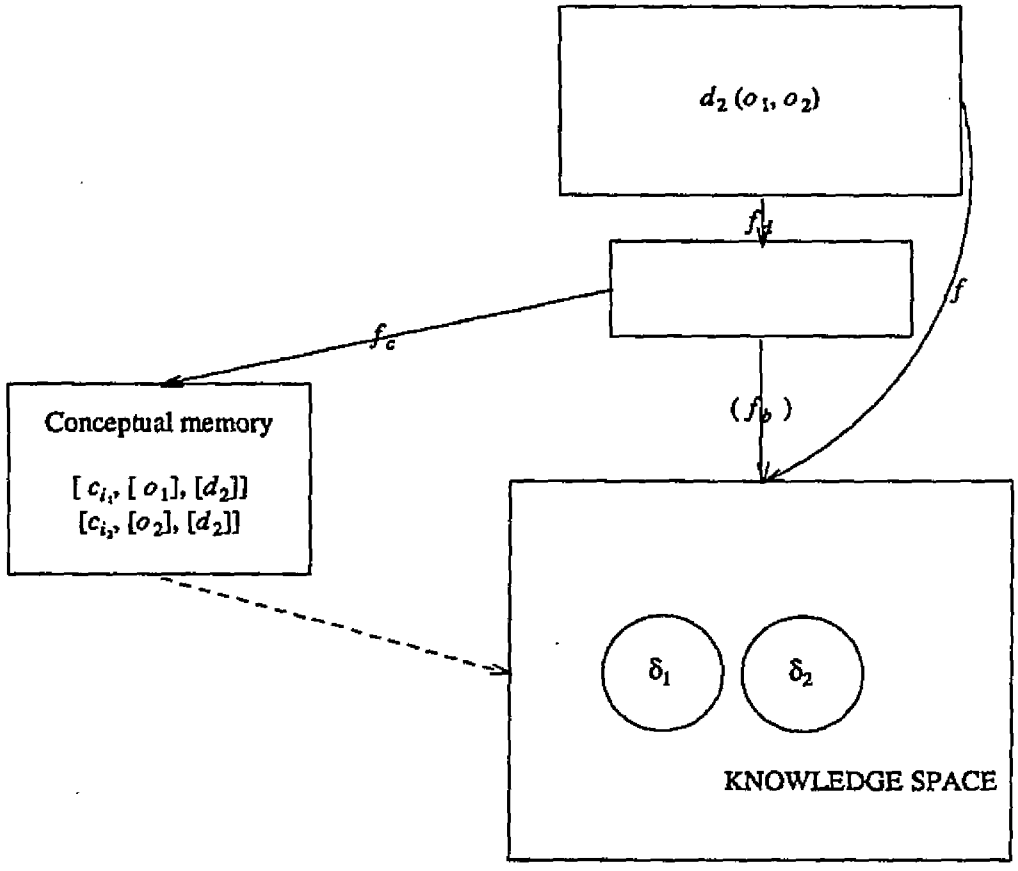


Figure 3.3 Components and Functions in COGMIR



Note: $f(d_2) = \delta_2$

Figure 3.4 Example of storage

Finally, we will use the **reference range of the boundary** to refer to all the objects that can be accessed through boundary, denoted $R_c(B)$, where B is the boundary for the document stem D.

Of course all the objects in the document stem fall in this range, but this range can contain more. We have the following theorem, which is crucial in query processing in COGMIR, as will be discussed in Chapter 4.

Theorem 3.2 The following relationship holds:

$$R_c(B) = \bigcup_{i=1}^{n} (R_c(c_{i_j}) \cup R_c(c_{i_k})),$$

where B is the boundary for document stem Δ, and c_{i_j} and c_{i_k} refer to the following: if $b_i \in B$, $i = 1, ..., n$, and t_{i_l} is an object which belongs to the interior of Δ and associates with b_i some relationship $\in \Delta$, then c_{i_j} is a concept satisfying $b_i \in R_c(c_{i_j})$, c_{i_k} is a concept satisfying $t_{i_l} \in R_c(c_{i_k})$.

Proof. Apparently, for any $o \in R_c(B)$, there are only two possibilities: either $o \in R_c(c_{i_j})$ or $o \in R_c(c_{i_k})$, so

$$R_c(B) \subset \bigcup_{i=1}^{n} (R_c(c_{i_j}) \cup R_c(c_{i_k})).$$

But B is the boundary; from the definition of the reference range of the boundary we have

$$R_c(B) \supset \bigcup_{i=1}^{n} (R_c(c_{i_j}) \cup R_c(c_{i_k})).$$

Therefore, the formula stated in the theorem holds. □

3.7 Note on the use of rules

In addition to the components defined before, there may exist an additional com-

ponent of the rule base, which is defined below.

Definition (Rule base). A rule base consists of rules that are used to coordinate the functions of the two kinds of memory (i.e., the knowledge space and the conceptual memory).

The purpose of these additional components is to enhance the ability of knowledge integration. There may be different rules to integrate knowledge at different levels:

(i) Rules at system level. These rules play a role of *controlling* the execution of the tasks.

(ii) Performing generalization and abstraction. For instance, there may exist rules for deciding task priority or rules for deciding structure similarity.

(iii) Dealing with conflicting information. Rules are used to keep the integrity of the knowledge, so that whenever new knowledge is not consistent with the old, there is a way to deal with this conflict.

(iv) Dealing with relationships between relationships. In order to compare the structure of different document stems, sometimes relationships between different documents should be compared, which is not necessarily a straightforward match. Rules will be used to compare the relationships.

(v) Other inference rules. For instance, some fundamental laws such as transitivity.

In this dissertation, we will not discuss the use of rules in detail, since they are already frequently discussed. In the structure to be discussed later, rules are implicitly built in the system architecture (for instance, we use a simple time stamp mechanism

to deal with conflicting information in Chapter 5), and no separate rule base will be included here (since they are not crucial in COGMIR).

3.8 Summary and discussion of the COGMIR model

Definition (COGMIR). The COGMIR model is a 8-tuple $(D, K, M, R, C, Dd, T, Q)$, where D is the document space, K is the knowledge space, M is the mapping mechanism, R is the rule base, C is the conceptual memory, Dd is the document description list, T is the temporary area, Q is the query.

The contents of this chapter can be summarized through the following two tables. Table 3.1 summarizes the components and the principles they realize. Table 3.2 is a comparison of the original Deutsch-Kraft model and COGMIR. The operating features on COGMIR will be discussed further in the next chapter.

On the other hand, we notice that the actual form of the components and the functions are not specified in their definitions. An example of implementation will be given in Chapter 5, where the data structures for COGMIR1 (a restricted version of COGMIR) are described.

COMPONENTS	REALIZED PRINCIPLE(S)
documents	DI
mapping	KR
document description list	II, CTA, SC
knowledge space	KR, II, SC
conceptual memory	CO, COA, CTA, SC

Table 3.1 Components and realized principles

IR	COGMIR
documents	cognitive documents (mental models)
--	mapping
storage of documents	integration of internal structure
--	knowledge space (long term memory)
thesaurus	conceptual memory
index	conceptual memory
document description list	document description list
query	query
--	problem type query
--	rule base
--	temporary area (short term memory)

Table 3.2 IR vs. COGMIR

CHAPTER 4

QUERY PROCESSING AND QUERY INVOKED MEMORY REORGANIZATION

In this chapter, we describe query processing in COGMIR. Retrieval in COGMIR is much more complicated than in a conventional IR system. It consists of the following steps: (i) determine the document relevant to the query; (ii) identify the corresponding document stems in the knowledge space; and (iii) from the document stems reconstruct the text. In this chapter, we will focus on (i) and (ii). After the general aspects of the query are examined, we will explain how to realize query invoked memory organization in COGMIR.

4.1 Query processing in COGMIR

We define a query in COGMIR as follows.

Definition (Query). A query Q in COGMIR is a task to invoke a process of locating a specific area in the knowledge space **K**, either (i) given object names from which the boundary of that area can be specified, or (ii) given part of the requested area along with object names from which the boundary of the requested area can be specified. That is, a query is given through the following formula

$$(i) \qquad R_c(B(\Delta)) \supset Q,$$

where Δ is the requested area for text (answer) D, and Q is a query;

or a query is given through another formula

$$(ii) \qquad \Delta \supset \delta,\ R_c(B(\Delta)) \supset Q',$$

where Δ' is a requested area that takes the form of document stems and satisfies $\Delta' \cup \delta = \Delta$ (Δ is the document stem for the answer D which is a text), δ is a given part of that area, and Q' some object names from which the boundary of the whole area can be specified.

This definition of a query satisfies the COA and CTA principle given in Chapter 2. A type (i) query is closer to a conventional query, while a type (ii) query has extended type (i) by providing partial structural information of the requested area. As we will see in Chapter 6, the definition of type (ii) query facilitates structure comparison and new structure generation.

Here we will have a brief discussion on the type (i) query. Just like a query in conventional information retrieval systems, a type (i) query in COGMIR takes the form of a query description list (a "vector" of object names) as defined below.

Definition (Query description list). A query description list (*QDL*) Q is an unordered set $\{q_1, ..., q_n\}$, where q_i $(i=1, ..., n)$ are objects to be retrieved.

According to Figure 3.3 and the definition of a query, the relationship in Figure 4.1 holds. In order to answer a query, the conceptual memory is searched, the relevant documents are determined, and through the boundary information stored in the document description list, the proper area in the knowledge space determined by these relevant document stems can be located, and finally, a text can be generated to answer the query.

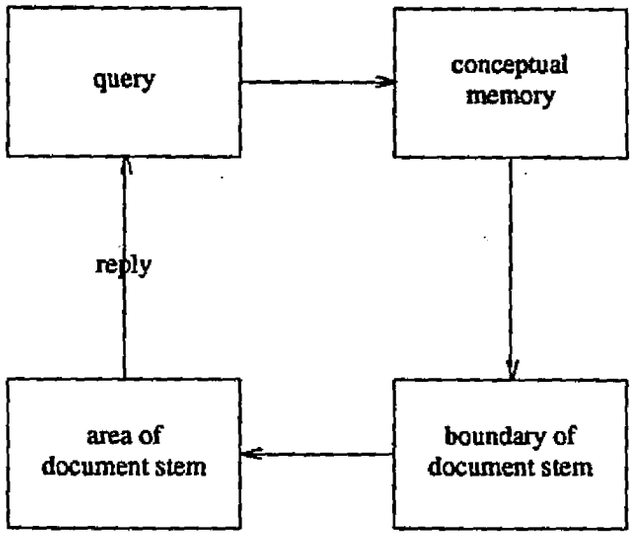


Figure 4.1 Query processing in COGMIR

The following theorems indicate the role of various components of COGMIR in answering a query of type (i).

Theorem 4.1 If a query $Q = \{q_1, ..., q_n\}$ can be answered, then there must exist a set C^*, $C^* = \{c_{k_1}, ..., c_{k_n}\}$, where $c_{k_i} \in C$ (conceptual memory).

Proof. If Q can be answered, according to the definition of a query, there must exist a document stem Δ so that $R_c(\Delta) \supset \{q_1, ..., q_n\}$. From the definition of the conceptual memory, for any q_i, there must exist some c_k so that $R_c(c_k) \supset q_i$, $c_k \in C$. []

Theorem 4.2 A query $Q = \{q_1, ..., q_n\}$ can be answered if and only if the boundary of an area in the knowledge space can be identified through the conceptual memory.

Proof. First, suppose the query Q can be answered, D is the answer (a text), and Δ is its document stem. Then according to Theorem 4.1, for any q_i, there must exist some

$$c_k \in C,$$

where C is the conceptual memory. Now,

$$f_c^{-1}(c_k) = D_i,$$

where D_i is the set of documents associated with the concept c_k, $D_i = \{d_{1_i}, d_{2_i}, ..., d_{n_i}\}$. Since the query about q_i can be answered, there must exist some $d_{j_i} \in D_i$ and some object o_{j_i}, so that either $q_i = o_{j_i}$, or q_i is able to be accessed by o_{j_i}. According to the definition of boundary, o_{j_i} must be on the boundary. All the other objects on the boundary can thus be found. On the other hand, if the boundary of an area in the knowledge space can be identified, so that starting from this boundary, and through conceptual memory, the requested term in the query can be accessed, then the query

can be answered. []

From this theorem we see the importance of identifying the boundary in the knowledge space. The task of the retrieval will thus be based on the boundary.

4.2 Two types of retrieval states

In doing retrieval, COGMIR differs from a conventional IR system in that two different spaces are involved: the document space and the knowledge space. Relevant documents are identified as units of the document space, while the knowledge to be retrieved is actually stored in the knowledge space. To reflect this kind of relationship, we use the *retrieval state* to indicate what is actually retrieved in terms of D and K. Since search is through conceptual memory in order to find the boundary of the documents (while the actual contents are stored in the knowledge space), two types of retrieval states are needed: the document retrieval state (DRS) and the knowledge space retrieval state (KRS), which are defined below.

Definition (Document retrieval state). The document retrieval state (DRS) is a pair (q_i, D) where $q_i \in Q$, Q is a query and D is a set $\{d_1, \cdots d_n\}$ where each d_i $(i = 1, ..., n)$ is a document relevant to q_i.

Definition (Knowledge space retrieval state). A knowledge space retrieval state (KRS) is a pair (c_j, O), where c_j is a concept, and O is a set of object locations $o_1, ..., o_n$ such that o_i is the set of memory addresses of objects that are related to C_j.

The relationship between these two kinds of states can be stated as:

Theorem 4.3 For any DRS, it is possible to construct a KRS, and vice versa.

Proof. If q_i is a query term associated with some DRS, this implies that there is a set

of documents D_i ($D_i = \{ d_1, ..., d_n \}$), D_i is relevant to q_i. Since D_i can only be obtained through the conceptual memory C, there must exist some $c_k \in C$ that is associated with q_i, and the relationship between q_i and D_i can be stated as

$$f_c^{-1}(c_k) = D_i,$$

where f_c is the document identification function defined in Chapter 3. According to the definition of the conceptual memory, $R_c(c_k)$ must contain a set of objects from D_i. Let this set be O_i, then $O_i \neq \varnothing$, and (c_k, O_i) forms a KRS. Therefore, any DRS determines a KRS. Conversely, reversing this process, from any KRS, the set of all documents related to these primitives can form a DRS. []

The purpose of having two kinds of search states is to make the task of retrieval meet the system structure. In COGMIR, the KRS is used to perform actual searching. This fundamental idea is shown in Figure 4.2, where the searching and matching goal state in DRS is conducted as searching and matching in KRS then converted back to DRS.

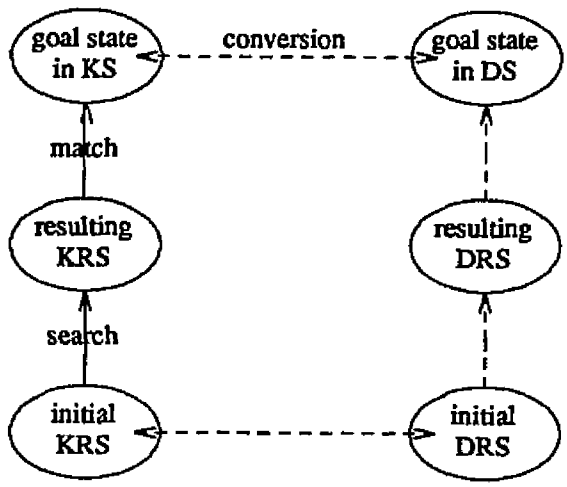


Figure 4.2 Conversion between DRS and KRS

To illustrate the relationship between the two kinds of retrieval states, we give the following example. Suppose, as the result of searching for a query $Q = (q_1, q_2, q_3)$, that $q_1 \in R_c(c_1)$, where c_1 is associated with documents d_1 and d_3 at object locations o_{11}, o_{13}, respectively, and $q_2 \in R_c(c_2)$ where c_2 is associated with documents d_2, d_3, d_5 at locations o_{22}, o_{23}, o_{25} respectively, and $q_3 \in R_c(c_3)$ where c_3 is associated with d_3 and d_5 at memory locations o_{33}, o_{35}, respectively. Then the KRS can be constituted as below:

Knowledge Space Retrieval States (KRS)

[concept c_i, memory location number of objects]

$[c_1, [o_{11}, o_{13}]]$

$[c_2, [o_{22}, o_{23}, o_{25}]]$

$[c_3, [o_{33}, o_{35}]]$

The DRS's correspond to the KRS's as follows:

Document retrieval states (DRS)

[query q_i, attached documents]

$[q_1, [d_1, d_3]]$

$[q_2, [d_2, d_3, d_5]]$

$[q_3, [d_3, d_5]]$

The DRS embodies the traditional interests of document retrieval, whereas the KRS is used to describe the contents of the documents. Therefore, these two state concepts, in a sense, unify document retrieval and fact retrieval. This unification also reveals the underlying nature of query processing in COGMIR.

4.3 Query invoked memory organization in COGMIR

In the previous section, we explained what should be done to answer a query. In this section, we discuss the actual process of retrieval and utilization of the stored knowledge. We explain how the fundamental idea behind the scheme of *query invoked memory reorganization* can be realized in COGMIR. In COGMIR, there is a need to reorganize memory for a query. And this scheme is particularly important when the retrieved knowledge is not acquired from different documents. The scheme of query invoked memory reorganization suggests that at storage time only possible ways of "binding" new input to old knowledge be recorded, as has been shown in the definition of conceptual memory (which records the possible connections). The actual realization of the knowledge integration is deferred to query time.

Figure 4.3 illustrates the different ways of realizing the "bindings". Four document stems δ_i are shown, $f(d_i) = \delta_i, i = 1, ..., 4$. The object O^* in document stem δ_4 is associated with the same object names O^* in document stems δ_1 and δ_2 through conceptual memory; and object O^{**} in document stem δ_2 is associated with δ_3 through the conceptual memory. With query Q_1, memory is reorganized in such a manner that δ_2, δ_3 and δ_4 are connected, and from the union of these three document stems, a fact can be constructed; for query Q_2, δ_1, δ_2 are connected and another fact can be constructed from the union these two document stems; for query Q_3, none of these document stems are actually connected, because they are irrelevant to this query. Therefore, although there is only one way to store the knowledge, there are potentially numerous ways to utilize it, depending on different queries.

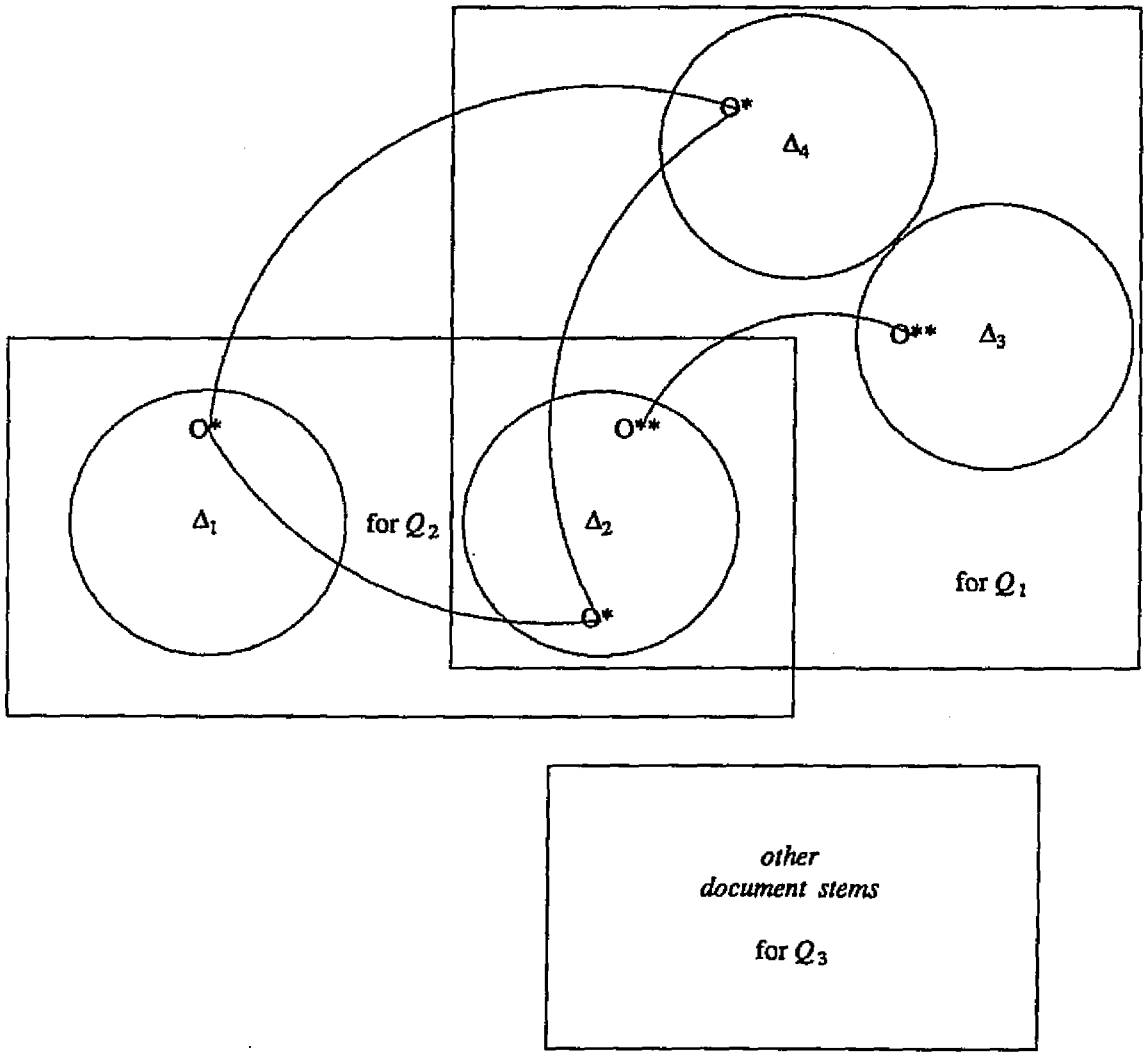


Figure 4.3 Different ways of reorganizing memory

Note: O's, O**'s are objects, Δ_i's are document stems*

The task of query invoked memory reorganization illustrated in Figure 4.3 can be more formally stated as follows:

Given query Q, retrieve text T from knowledge space **K** *by organizing the relevant contents on the demand of Q.*

In a sense, query invoked memory reorganization is similar to the "view" (external view) construction of the relational database. Using relational database terminology, a view is a virtual relation which is not part of the conceptual model that is made visible to a user. Views are not stored; a view must be recomputed from the actual relations for each query that refers to it. Different users of a shared database may benefit from the individualized views of the database [Kort85]. Therefore, a fact is similar to a view. Similar to join operations on a relational database, we may have join operations on the document stems. But due to the nature of COGMIR, the construction of a fact is far more flexible than the view construction in relational databases. Moreover, as we will see in a case study which to be given in the next two chapters, the structure used can be treated as a list-formed relational database. Nevertheless, essential differences exist between our approach and the database theory. Roughly speaking, we are only concerned with the top layer, the semantic layer of the relational database; further, we are not concerned with some topics such as functional dependency [Ullm82].

In addition, to use query invoked memory reorganization, COGMIR realizes knowledge integration using a kind of "dictionary based reminding" [Scha82] which is not suitable for episodic memory but is suitable for integration of scientific knowledge.

4.4 Different kinds of query invoked memory reorganization

Generally speaking, reorganization refers to utilization of the stored knowledge in an on-demand, flexible manner. The reorganization can be carried out in different ways.

(1) Identification and/or reconstruction. The simplest way is to identify a related area in knowledge space or to approximately reconstruct the original documents in a way easily understandable to the user. Memory is reorganized in the sense that a related area is highlighted. We will discuss this kind of query under the title "document retrieval" in subsection 4.4.1.

(2) Construction of a fact. At a more sophisticated level, the query is a guide to identify an area of knowledge space which comes from several documents. These documents form a D-cover of a fact to be retrieved. Memory is reorganized in the sense that the original implicitly recorded knowledge now becomes explicit. We will discuss this kind of query under the title "fact retrieval" in subsection 4.4.2.

(3) New structure generation. At the most sophisticated level, the query may act as a directory to generate new structure in the knowledge space, based on currently available knowledge. In this case, memory is reorganized in the sense that the knowledge space is enriched. We will discuss this kind of query under the title "pseudo fact retrieval" in subsection 4.4.3.

In any case, the query is answered by first searching some documents, which will be discussed separately.

4.4.1 Document retrieval

The task of document retrieval can be stated as follows:

Given type (i) query Q, get document d such that

$$R_c(B(f(d))) \supset Q,$$

where f is the mapping function from **D** to **K**, and B stands for the boundary of $f(d)$.

Document retrieval is carried out by manipulating the DRS. An example of document retrieval is illustrated in Figure 4.4, where a document d_2 is retrieved to answer the query $Q = (q_1, q_3)$.

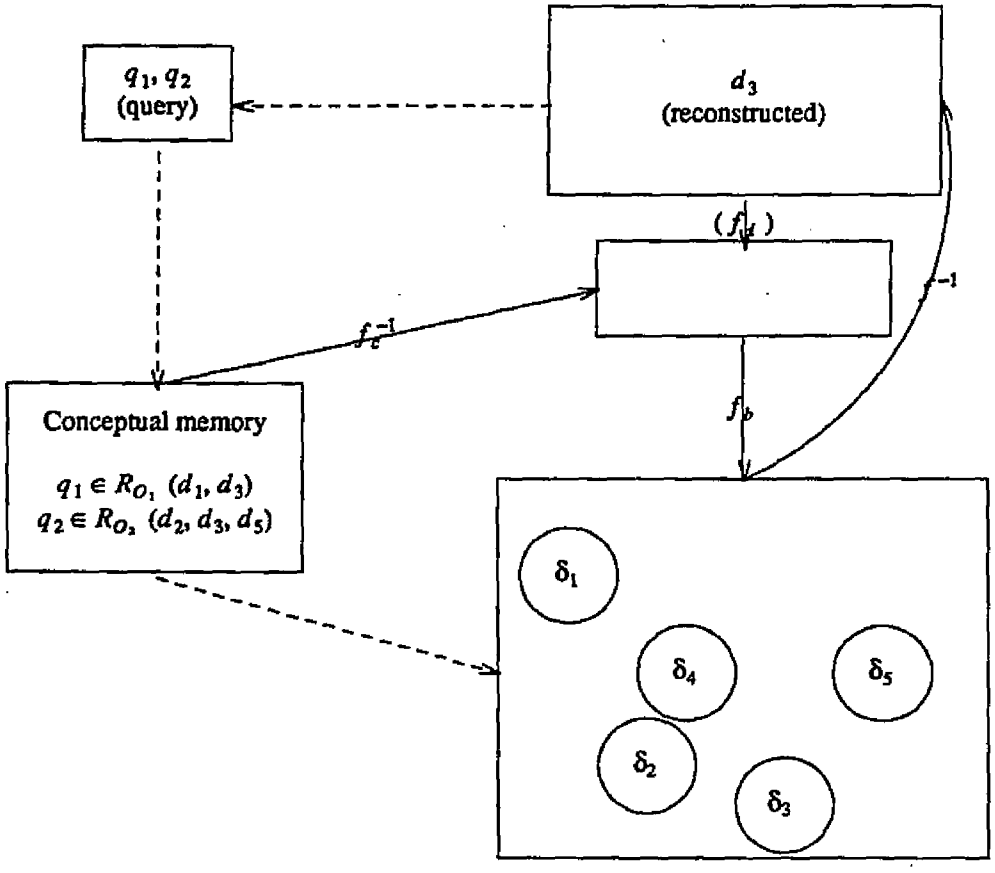


Figure 4.4 Example of document retrieval

4.4.2 Using documents as a clue to start heuristic search for fact retrieval

Since it is assumed that documents are mental models, they are potential candidates for starting a search. That is, the search always starts from some possible boundaries in the knowledge space.

In the framework of COGMIR, fact retrieval can be treated in a way similar to document retrieval, the only difference is that we first construct the boundary of the requested fact based on the boundary of the documents (a kind of memory reorganization process invoked by the query). The task of fact retrieval in COGMIR is to find some documents that constitute a fact so that the query description list will fall in the reference range of the boundary of the corresponding document stem. That is,

given the type (i) query Q, get a series of the documents

$$d_{Q_1}, d_{Q_2}, \ldots, d_{Q_n},$$

so that

$$R_c\,(B(\bigcup_{i=1,\ldots,n} f(d_{Q_i}))) \supset Q,$$

where B stands for the boundary and f stands for mapping function from **D** to **K**.

Similar to DRS and KRS in document retrieval, fact retrieval involves DRS and KRS, but there is an additional step. Unlike document retrieval where after the documents are retrieved, the task is done, fact retrieval involves an additional task to construct the fact from these retrieved documents. This is done by manipulating the boundaries of the documents that are in the D-cover. The formation of such a boundary is where document retrieval and fact retrieval differ. The process of fact retrieval with an example is shown in Figure 4.5, where documents d_1, d_2, d_3, d_4 were acquired

before. To answer a query, a fact with d_1, d_2, d_4 is constructed and presented to the user. Since the entire retrieval process is transparent to the user, he may think this fact is retrieved from the document space. In this sense, document retrieval and fact retrieval are unified.

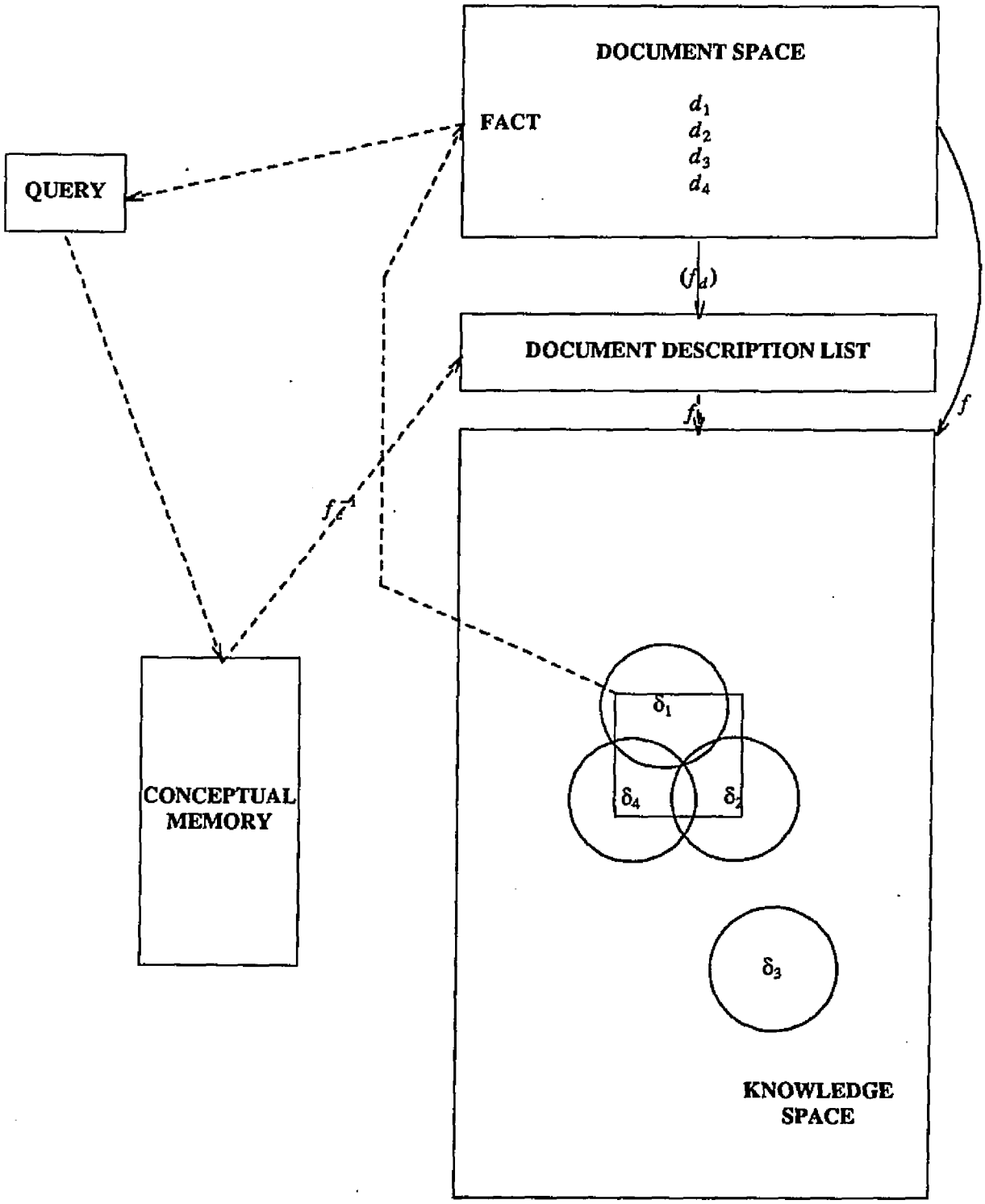


Figure 4.5 fact retrieval
(A fact covered by d_1, d_2, d_4)

* (Solid lines indicate operations related to storage, dashed lines indicate operations related to query)

4.4.3 Pseudo fact retrieval

In addition to retrieval by document and by fact, retrieval may be done in a more active manner. If the requested knowledge does not exist, new structures can be generated according to certain criteria. An example was shown in section 3.1. The generated text will be referred to as a *pseudo fact.* New structure generation is a way to actively utilize the stored knowledge invoked by a query. Structure generation is performed by structure comparison followed by structure mapping. Both structure comparison and structure mapping are processes based on the actual structure of the knowledge space.

In Chapter 3 (section 3.5) the definition of structure mapping was given. To incorporate the definition of type (ii) query given in section 4.1, we have:

Definition (structure generation, pseudo fact retrieval). By structure generation invoked by query Q with type (ii) we mean a document stem Ψ contains some objects and relationships which are not obtained from any document. This kind of retrieval is named *pseudo fact retrieval,* and $f^{-1}(\Psi)$ is called a *pseudo fact.*

Since the retrieval of a fact F is a result of fact retrieval, pseudo fact retrieval also concerns DRS and KRS. Unlike more conventional retrieval, however, the fact F itself is not retrieved to satisfy Q, but some other "untold" query Q'. The document stem of the fact for answering Q' is mapped to generate Ψ. Figure 4.6 illustrates the pseudo fact retrieval. Basically, it consists of two phases. At the first phase, the user enters an *incomplete document* which is similar to a document, but different from a regular document in that part of its content is to be filled. Similar to a regular document, the incomplete document is converted into its internal structure, but is mapped

into the temporary area rather than the knowledge space. At the second phase, from the document stem of this incomplete document, a complete document stem can be generated from some fact through structure mapping. (Some detail for a case study can be found in Chapter 6). Finally, a pseudo fact can be constructed from this resulting document stem. Therefore, an incomplete document plays the role of a query on one hand, and becomes part of the answer (i.e., the complete pseudo fact) on the other.

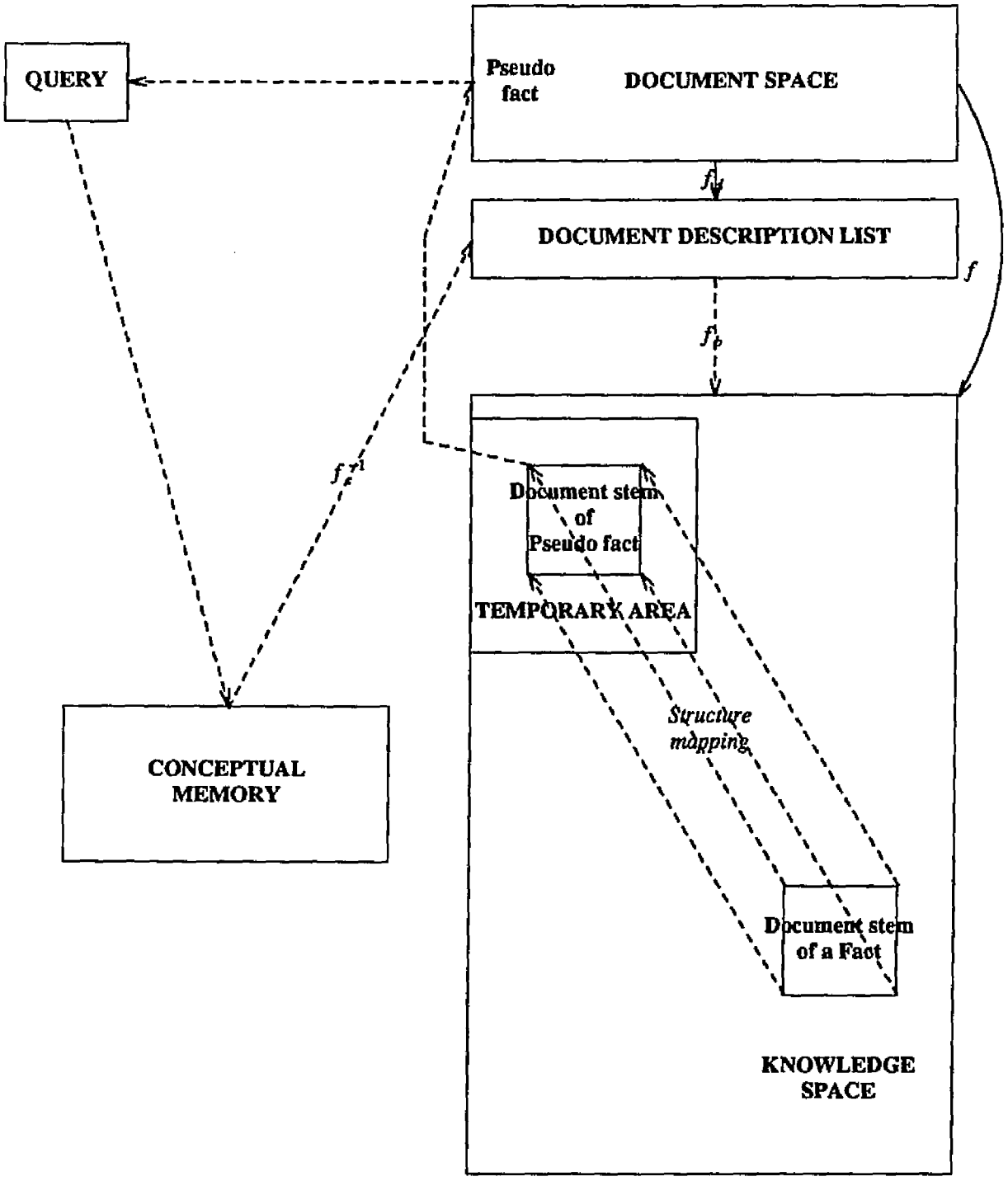


Figure 4.6 Pseudo fact retrieval by structure mapping

* (Solid lines indicate operations related to storage; dashed lines indicate operations related to query)

4.5 Summary: Query invoked memory reorganization implies inference

Now we are going to summarize the different types of memory reorganization. We will end this chapter by pointing out that query invoked memory reorganization implies inference, therefore, it implies a sense of intelligent retrieval.

First we point out that the scheme described in this chapter provides a unified way to view document retrieval and fact retrieval. Both document reconstruction and fact construction are the process of performing f^{-1}, the only difference is that in fact construction an additional step of constructing the boundary for the fact from the boundaries for the documents is required. Intuitively, both these two processes are realized by "expanding" from the boundary to the interior of the area.

There may be different algorithms to actually carry out this kind of (re)construction, depending on the actual form of the components. In the case study discussed in the following chapters, we will give an actual algorithm for document reconstruction by boundary expansion.

Both document retrieval and fact retrieval utilize the same functions such as f_c^{-1} and f^{-1}. The difference is that for fact retrieval there is an additional process of fact construction; moreover, inductive reasoning may be involved in the fact construction. But for the user, the underlying manipulations are transparent. Therefore, he works in a unified retrieval environment.

We have discussed different types of retrieval as different schemes to realize memory reorganization. They are document retrieval, fact retrieval and pseudo fact retrieval. The difference between several terms is summarized below. A *document* is

an input unit, and can be reconstructed (in the sense of document equivalence). A *fact* is constructed from document stems. Usually it is not an actual document (that is, it is not an input unit), but it looks like a document, and the contents contained in a fact come from the previously acquired documents. On the other hand, a *pseudo fact* is constructed to be similar to a fact, but it contains some objects and relationships that are not obtained from any document; it is a suggestion or a guess based on previously acquired knowledge. Finally, we have informally mentioned an *incomplete document* (to be further discussed in Chapter 6 through a case study), which is similar to a document in that it is typed in by a user, but different from the regular document in that part of its content is to be filled in by using existing knowledge. Therefore, an incomplete document plays the role of a query.

Underlying these different types of queries is the intelligent retrieval of knowledge. The scheme of query invoked memory reorganization implies inference; the capability of inference is built into the COGMIR model. This is obvious if we compare the behavior of COGMIR with human intelligence. For instance, to construct a fact from different documents requires intelligence; generating pseudo fact based on existing knowledge again requires intelligence. Furthermore, we point out that conceptual memory implies "if ... then" type rules. For instance, the use of the reference range can be explained as "if unable to retrieve a document relevant to object o_1, then retrieve a document relevant to object o_2, where o_2 falls in the same reference range of some concept in which o_1 falls." In addition, topics such as generalization and abstraction of the knowledge space in the COGMIR framework also concern inference, although we will not discuss them in detail.

In the next two chapters, we will give a detailed case study to show how the considerations in this chapter can be realized.

CHAPTER 5

REALIZING MEMORY INVOKED MEMORY REORGANIZATION IN COGMIR1: A CASE STUDY

5.1 A restricted version of COGMIR

In Chapter 3 we proposed a general model. From now on we consider a restricted version of this model so that more structural detail can be considered. This restricted version will be referred to as COGMIR1. It can be viewed as a case study of the general model. Although the general framework of COGMIR is related to classical IR model, this restricted version has more elements common with database systems.

The input documents are rewritten into English-like sentences. The purpose is to make our work self-contained. Furthermore, we assume the order that the documents enter the system forms a natural *time stamp*. This facilities the consideration of dealing with time and causality, as will be explained later. The time stamp is imposed through document description number, as well as the memory location number for objects and relationships. In order to avoid any possible confusion, in this case, we will assume document description number start at 1, and memory location number start at 101.

Under these restrictions, components of COGMIR take a frame-like structure, with slots standing for different fields. This kind of structure can be easily dealt with by list presentations.

In this case study, which consists of Chapters 5 and 6, we want to define the data structure for all of the actual components, and discuss the involved manipulations: how to map the input into internal structure, how to make knowledge tie together, how the knowledge space is constructed, and how the query-invoked memory reorganization is performed. Central to a possible realization of the query-invoked memory reorganization is to represent the knowledge in list-formed databases, upon which the considerations appearing at the end of Chapter 4 can be realized.

In COGMIR1, documents are written in particularly defined English-like sentences, the grammar for which will be referred to as G. It is used to represent a part of scientific knowledge through objects and simple relationships. The sample inputs in Chapter 1 can be rewritten using G. The represented knowledge does not necessarily fall in a particular knowledge domain. According to the syntax of G, sentences are restricted by assuming that each sentence consists of two objects associated through one relationship or three objects associated through two relationships. The other types of allowable sentences are "is/are" type sentences and questions. The purpose of defining this grammar is to simplify the mapping process, so that it can be handled by a non-sophisticated parser and we can focus our study on the internal structure of the documents. Using this grammar, documents can be rewritten. The grammar is given in Table 5.1, in which variables start with upper case letters while words starting with lower case letters indicate constants. Since we have assumed that the input documents are mental models, the sentences in the documents are components of these mental models; therefore, they are not only syntactic units, but also *semantic* structures.

Note that item in parentheses means optional; "*" denotes repetition; "|" means choice. [noun-list] is a pre-defined noun list. [verb-list] is a pre-defined verb list. [adj-list] is a pre-defined adjective list. [prop-list] is a pre-defined proposition list.

Sentence ::= <Object> <Relationship> <Object> <Stop> |

<Object> <Relationship> <Object> <Relationship> <Object> <Stop> |

<Object> <LinkVerb> <Adj> (and <Adj>)* <Stop> |

<Question>

Object ::= <Noun> | <Art> <Noun>

Relationship ::= <VerbGp>

Art ::= a | an | the

VerbGp ::= <Verb> | <Verb> <Prop> | <Prop>

LinkVerb ::= is | are

Noun ::= [noun-list]

Verb ::= [verb-list]

Adj ::= [adj-list]

Prop ::= [prop-list]

Question ::= how <Object> <Relationship> <Object> <Stop>

Stop ::= .

Table 5.1 Grammar G

A special type of sentence is the question, which is used in a user query. Each query may have more than one question. Questions are handled by using the reserved word "how." Therefore, in G, the period is the only punctuation.

Some general rules for writing sentences using G are listed in Table 5.2.

---

(1) To describe the ownership of the object, use the relationship "has".

(2) To describe an attribute of any object, use the relationship "is" or "are". The object must have appeared before in the same document. Therefore, these attributes may be used to describe this object.

(3) All the sentences in the same document are ordered in a time sense.

(4) A question is limited to be used one per document at the end of a document.

(5) A question mark "?" is used to indicate the end of any input.

Table 5.2 General rules for writing documents using G

Here are some examples of the sentences in different types:

the kite flies in air.

the arteries carry blood to heart.

the seeds are small and dark.

how doctor removes seeds.

In the introductory example of Chapter 3 some examples of documents were given. Some sample documents written in G can be found in Appendix A. On the other hand, a special type of document that ends with a question will be referred to as an *incomplete document,* and will be further discussed in Chapter 6.

In the process of encoding, each English-like sentence is mapped into an internal structure by updating or adding one or more items in the object list or the relationship list. Roughly speaking, this is done by converting nouns or noun groups into objects, adjectives and adverbs into attributes, and verbs and verb-clauses into relationships, respectively, with some exceptions. Consequently, each document has its own internal structure. At any time, the collection of used memory locations, which record the internal structures obtained from outside and inferred from itself, form the knowledge space of that instant.

G has been implemented in Prolog. In a sense, the memory of COGMIR1 can be viewed as a revised version of a Prolog database with all the relationships ordered. In order to easily capture the effect of time and causality, we assume events described in the sentences occur in order of their number in the document.

5.2 Representing knowledge using lists

The knowledge space, as defined in Chapter 3, can be built as a net-like structure. There are two basic constructs: objects and relationships. Relationships indicate how the objects are associated, and objects also indicate what relationships are associated with them.

All the data structures will be represented in terms of lists, using terminology of Prolog. Lists can be nested, and the contents of a list are put in square quotes, that is,

[]. The list representation for the introductory example used in section 3.2 is shown in Figure 5.1, and the meaning of the object list and relationship list stand for is already illustrated in Figure 3.1 graphically. These lists are the results of the functions f, f_b, f_c and f_d operated on the data structures of COGMIR1; for instance, the object list and relationship list are the result of function f, the document description list is the result of function f_d and f_b^{-1}, and the conceptual term list is the result of function f_c. In the following, we will give the definitions of the data structures for the system components. But instead of directly defining the various functions, we will give algorithms to describe how these lists can be constructed from the input documents.

INPUT DOCUMENTS:

1. the arteries carry blood from the heart. the veins carry blood to the heart.
2. bats emit sound. the sound is inaudible. an obstacle reflects the sound. the obstacle is invisible. bats detect the obstacle.
3. a scientist discovers the capillaries. the capillaries connect the arteries. the capillaries connect the veins.

Document description list:
[[1, [*], [[arteries, 101], [blood, 103], [heart, 104], [veins, 106]], [101, 103, 106, 103]],
[[2, [*], [[bat, 109], [sound, 111], [obstacle, 113]], [109, 113, 113, 109]],
[[3, [*], [[scientist, 115], [capillaries, 117], [arteries, 119], [veins, 121]], [115, 117, 117]]].

Conceptual term list:

[[arteries, [*], [1,3]],
[[blood_ vessels], [arteries, capillaries, veins], [1,3]],
[bat, [*], [2]],
[blood, [*], [1]],
[capillaries, [*], [3]],
[heart, [*], [1]],
[scientist, [*], [3]],
[sound, [*], [2]],
[obstacle, [*], [2]],
veins, [*], [1,3]],
...].

Object list:

[[101, [arteries], [], [102]],
[103, [blood], [], [102]],
[105, [heart], [], [104]],
[106, [veins], [], [107]],
[109, [bat], [], [110, 114]],
[111, [sound], [inaudible], [110, 112]],
[113, [obstacle], [invisible], [112, 114]],
[115, [scientist], [], [116]],
[117, [capillaries], [], [118, 120]],
[119, [arteries], [], [118]],
[121, [veins], [], [120]]].

Relationship list:

[[102, [carry], [], [101, 103]],
[104, [to], [], [103, 105]],
[107, [carry], [], [106, 103]],
[108, [from], [], [103, 105]],
[110, [emits], [], [109, 111]],
[112, [reflects], [], [113, 111]],
[114, [detects], [], [109, 113]],
[116, [discovers], [], [115, 117]],
[118, [connect], [], [117, 119]],
[120, [connect], [], [117, 121]]].

Knowledge space

Note: the symbol [*] stands for a list with contents not shown

Figure 5.1 List representation for the introductory example

Query: "heart, capillaries"

A fact constructed for the query "heart, capillaries":

arteries carry blood to heart. veins carry blood from heart. capillaries connect arteries. capillaries connect veins.

(1) Representing documents. All the short paragraph-like documents are written in G. Some examples are shown in Figure 3.1 as well as in Figure 5.1.

(2) Representing objects. Basically, the role of objects may fall in one of the following categories: an object is an actor of an action (i.e., the relationship), or an object is something worked on by some other object through some relationship. In order to distinguish these two types of objects, the first type of object will be referred to as *actor* while the second type will be referred to as *actee* (the term "actee" is coined to reflect its relationship to "actor"). The relationship between an actor and an actee is recorded in the associated object list in any relationship, which will be ordered as $[A_r, A_e]$, where A_r is an actor and A_e is an actee related to A_r through a relationship R. For instance, for "engineers design machine", if the location number for "engineers" (the actor of the relationship "design") is 201, and the location number for "machine" (the actee for the relationship "design") is 203, then the ordered list will be represented as [201, 203]. Therefore, an object O in COGMIR1 not only has its own memory location and attribute list, but is also the thing described by its related relationships.

An object name is a noun defined in grammer G. An object can be attached with an attribute list as well as other associate lists. Each object represented by a tuple written in list form. Each object tuple in COGMIR1 has the following format:

$$[L, [N], [A], [R]]$$

where

L -- location of the object

N -- name of the object

A -- attribute list

R -- location of related relationships

Examples of object tuples are shown in Figure 5.1, their graphical representations were previously shown in Figure 3.1.

(3) Representing relationships.

A relationship name is a verb or verb phase defined in G. Each relationship in COGMIR1 takes the form of a tuple, which has the following format:

$$[L, [N], [A], [A_r, A_e]]$$

where

L -- location of relationship

N -- name of the relationship

A -- attribute list

A_r -- location of the actor

A_e -- location of the actee

Examples of relationship tuples are given in the relationship list of Figure 5.1.

(4) Manipulating union of document stems.

A union of object tuples is a set of object tuples listed according to the increasing order of object locations. A union of relationship tuples is a set of relationship tuples listed according to the increasing order of relationship locations.

(5) Representing knowledge space.

The knowledge space **K** is a union of object tuples, O, and a union of relation-

ship tuples, R, where R involves only those objects that are appeared in O. The object list and the relationship list consisting the knowledge space of the previous example is shown in Figure 5.1.

(6) Representing document stems.

A document stem is a set of object tuples $(O_1, ..., O_n)$ along with some relationships among these objects. A document stem has the following format:

$$[[O], [R]],$$

where

O -- object list (consists of object tuples)

R -- relationship list (consists of relationship tuples)

The union of document stems in COGMIR1 is a document stem which takes all the object tuples and relationship tuples of those document stems. All the objects are listed in the increasing order of objects, so do all the relationships. Within the same document stem, two objects with same name appeared in the object tuples are treated as one object. The union of two document stems is to put the two document stems according to a certain order. This will be discussed later.

(7) Representing document description list.

The document description list in COGMIR1 is represented as

$$[D, [P], [O], [B]]$$

where

D -- document reference number,

P -- property list,

O -- object location list,

B -- boundary.

The document reference number is a serial number used to identify all the documents. This number is assigned to every newly acquired document by increasing 1. The property list may be used to record general information about the documents, such as year or author, as well as other information. The object location list consists of sublists, each sublist taking the is a pair of [name, location]. All the object locations appeared in the document stem are recorded. Finally, the document description list also records the important information of boundary. In COGMIR1 the boundary is a list of actors ordered according to their appearance in the sentences (may appear more than once if they appear in the same document more than once).

(8) Representing conceptual memory. In order to keep the structure simple, the conceptual memory in COGMIR1 is constructed as a *conceptual term list,* which consists of conceptual terms. To facilitate the search, the objects in the reference range can also be made directly referenced. Each conceptual term is represented as follows:

$$[C, [R], [D]]$$

where

C -- a conceptual term, which is an object name

R -- role list

D -- document list (document reference numbers of the documents that are related to C)

Objects may relate to concepts in many different ways. In this case study we just consider a special way, that is, the objects related to a concept play the "role" of this concept. These objects form the reference range of this concept. The role list may be

predefined, but may also be updated by hand. The document list part of the conceptual term list is updated by the system. The counterpart of this process in the classical IR system is automatic indexing [Salt84]. Of course, an automatically constructed conceptual term list will show greater intelligence, but even a pre-defined conceptual term list (with only its document list part is updated whenever a new document comes) will suffice to perform the major task of query invoked memory reorganization. An example of conceptual memory using the format defined here was already given in section 3.2.

5.3 Processing and storage of the input documents

From now on we describe some fundamental manipulations on COGMIR1. There are different types of operations for COGMIR1. There are operations related to mapping, locating the related area in knowledge space, and document and fact (re)construction. These operations concerns the integration of the documents, as well as the realization of query invoked memory reorganization.

The general idea behind these algorithms is to convert the input into the internal structure, which is a kind of linear list. Searching can therefore be simplified. As we will see, the time stamp as discussed in the beginning section of this chapter plays an important role in the controlling of these algorithms.

In our current case study, mapping input into the knowledge space is a process of parsing plus list handling.

(1) Parsing individual sentence.

The following are the general steps of the conversion for each sentence:

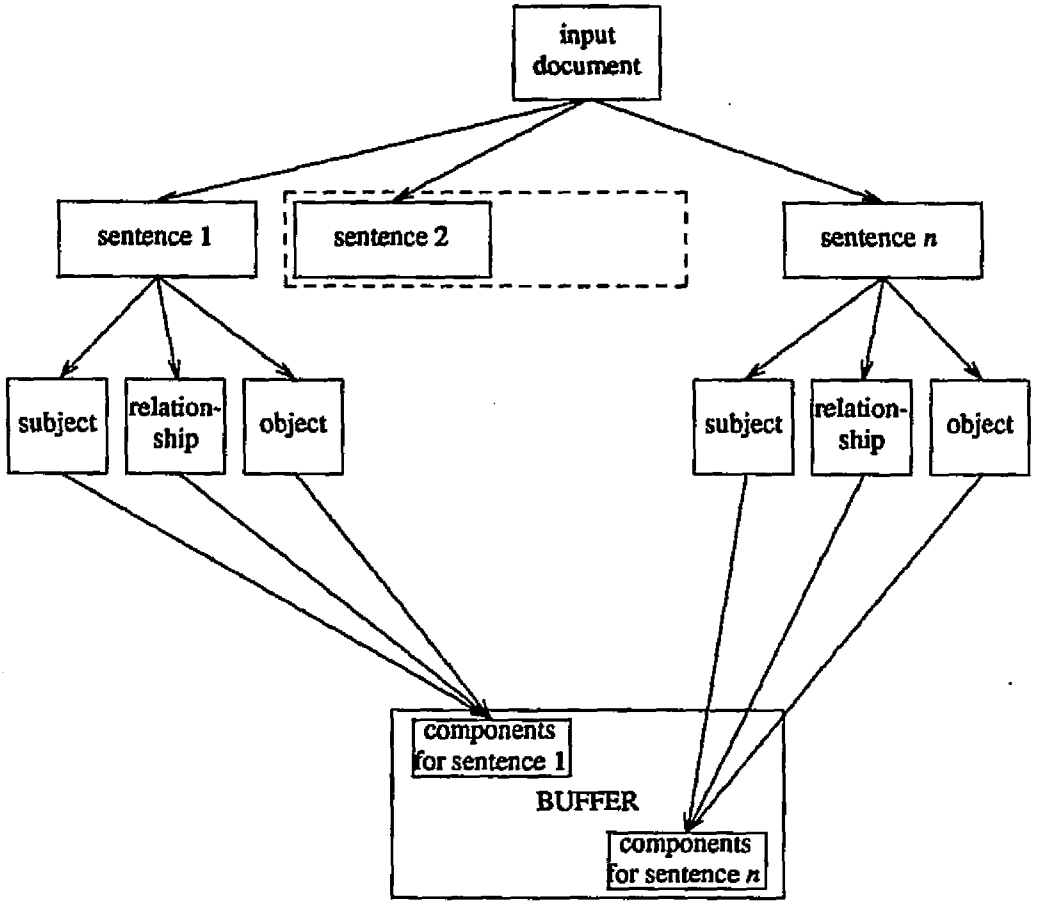
(i) Separating individual sentences. The punctuation "stop" (.) is the notation ending and separating sentences.

(ii) Parsing. The task of parsing is to check whether (1) each word in the sentence is a legal word (i.e., it is either in the noun list, verb list or proposition list); and whether (2) the sentence is in one of the three types as defined in Tables 5.1 and 5.2.

(iii) Identifying parts of the sentences. During parsing, the parts of the sentences (such as actors, relationships and the actees) are identified. Basically, each sentence is converted into the set of actor, relationship, and the actee, except those sentences in which more than two objects are involved (as defined in Table 5.1). In this latter case, a sentence is first decomposed into two sentences with each involves two objects (i.e., one actor, and one actee).

(iv) Store the parsed parts of the sentences in a temporary buffer. Except for the type of sentences in which more than two objects are involved, all the components of any sentence are stored in specified places in the temporary buffer. For the type of sentences in which more than two objects are involved, are first transformed into two sentences with each of them having only two objects, and stored separately.

The process of parsing can be illustrated in Figure 5.2.



Note: dashed box indicates posible decomposition of sentence

Figure 5.2 Parsing a document

(2) The mapping process

After all the sentences in a document are all parsed, the sentences in this document are converted into the internal structure and stored in the overall knowledge space. At any time, the collection of used memory locations, which records the internal structures obtained either from outside or inferred from itself, form the knowledge space of that instance. Moreover, all the related data structures, such as the document description list and the conceptual term list, are updated.

We now describe the process step by step.

(1) For each newly acquired document, form a new item for the document description list. Increase the document reference number by 1, and record the boundary of the document stem, which consists of the actors of all the sentences.

(2) For each document, update the conceptual term list by adding the new document reference number to the relevant concept.

(3) For each sentence in this document, construct or update object tuples and construct relationship tuples as follows. For each occurrence of a relationship, creat a new relationship tuple. For each occurrence of an object, if the object has not appeared before in this document, create a new tuple for this object; otherwise, update this object tuple (which was created by previous sentences in this document) by adding the relationship location to the relationship list.

In order to deal with error handling, in COGMIR1, the user actually work with a buffer. All the lists and tuples obtained from a new document are temporarily stored in the buffer. As soon as the user finished entering a valid document, these lists and

tuples will be appended to the proper lists, the conceptual term list and knowledge space are thus updated.

5.4 Operations on document stems

In this section, we define the following two operations of document stems in COGMIR1: union and chopping. Examples will be given in Figure 5.4.

In Chapter 3, we have defined the operations on document stems for the general COGMIR model. In this section, we give a more detailed definition of the union operation in terms of the specific structure of COGMIR1.

Definition (Union of document stems). The union of two document stems δ_i and δ_j in COGMIR1 is the appending of these two document stems according to the increasing order of their document reference numbers. The object tuples in one document stem are appended by the object tuples in another document stem, and the relationship tuples in one document stem is appended by the relationship tuples in another document stem, both according to the increasing order of their location numbers.

Due to the storage algorithms described in previous section, the location numbers for both object tuples and relationship tuples appear in a document stem in proper order, so this definition will not cause any conflict.

Definition (Chopping of document stem). The chopping of a document stem based on a set of location numbers (for a set of objects) refers to the following operation: get the lower bound min and upper bound max of these location numbers (of these objects), then remove all the tuples in this document stem whose location numbers do not fall in the range determined by max and min.

The rationale of having this operation is to remove those tuples which are not under current concern. The reason to keep all the tuples within the whole range (even if their location numbers are not in the specified set) is due to the assumption of time and causality (as discussed earlier). The chopping operation is useful in fact retrieval. When performing fact retrieval, the set of location numbers as mentioned before is usually specified in the fact retrieval plan (to be defined in section 5.5.3).

The use of these two operations will be further discussed in next section.

5.5 Document retrieval and fact retrieval

In this section we discuss how to perform retrieval in COGMIR1. Here we only consider type (i) query (as defined in Chapter 4), the other type of query will be discussed in Chapter 6.

5.5.1 Determining partial relevance of documents

The basic function of answering a query is to locate a specific boundary in order to retrieve the area (in the knowledge space) confined by this boundary and then actually construct (or reconstruct) the contents carried in this area.

To retrieve the relevant document, the process follows exactly the basic idea shown in Chapter 4 through the use of conceptual term list. After the document relevant to a query is determined, the process of document reconstruction from the document stem will be discussed in section 5.5.4. If there is more than one document relevant to the query, these documents will be ranked according to the reverse order of the document reference number.

5.5.2 Determining boundary and area for a fact

Since fact retrieval concerns more than one document stem, we need to find some way to indicate the relationship between different document stems. First, we give the following definition.

Definition (path). A path in the knowledge space K of COGMIR1 is an ordered sequence of objects (O_i's) and relationships (R_i's) taking the form

$$p_1, ..., p_n$$

(where p_i, $i = 1, ..., n$ is either an object or relationship) satisfying the following two conditions: (1) each relationship R_i associate the two objects which appear immediately before it and after it, and (2) any object can be immediately followed by at most one object, and this object must have the same name and larger location number.

The reason to restrict the consequent objects with the same name is to impose the time stamp. The following is an example is a path (objects and relationships are identified by their memory location numbers) in the introductory example of Chapter 3, which satisfies the requirements in the definition, and is illustrated in Figure 5.3:

106, 107, 103, 117, 120, 121

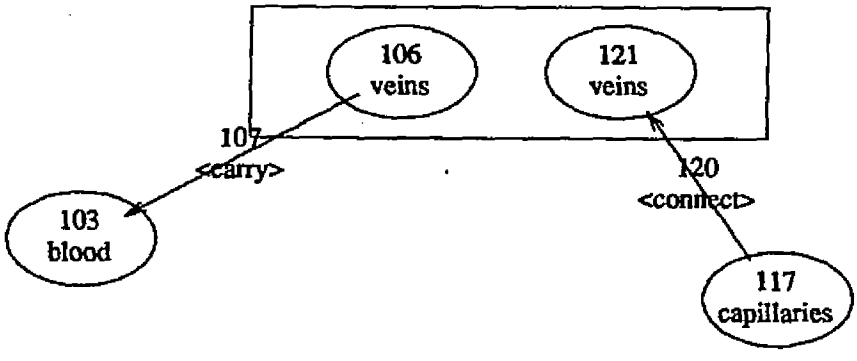


Figure 5.3 A path
(Attributes of objects not shown)

The purpose of introducing the concept of path is to define the the connection of document stem.

Definition (connected document stem). A document stem is connected if for any two objects in this document stem, there exists at least one path. Here different instances of same object are treated as same object.

Based on this definition a fact in COGMIR1 can be represented through the document stems in the following manner. A fact in COGMIR1 is a set of sentences written in G, and is constructed from a connected document stem in which the locations of objects appear in an increasing order.

5.5.3 Fact construction

To discuss the fact construction, we start from the following theorem.

Theorem 5.1 If QDL is the query description list for a fact retrieval, and after reordering this QDL, if (i) there is a way to divide it into several parts $Q_i, i = 1, ..., n$, each can be retrieved from a document d_{i_m} (satisfies $d_{i_m} < d_{i_n}$ if $m < n$), and (ii) the Q_i's are related in the following way: for any Q_i, there always exists some $q_{k_i} \in Q_i$, and some $q_{j_i} \in Q_j$ $(i \neq j)$, so that q_{k_i} and q_{j_i} have same object name, or they are associated with objects with same name (through some relationships), then the d_{i_j}'s form a D-Cover for the requested fact.

Proof. In order to prove the theorem, we notice that the fact to be constructed (as stated in the theorem) consists of several parts, each is a part of a document. Condition (ii) in the theorem implies that there is a kind of overlap of the boundary objects name, or the overlap of the object names that are associated through relationships; this

indicates that this fact consists of a path across the boundary. Moreover, the time stamp required by the union of document stems is satisfied by condition (i). Therefore, the union of these documents form the D-cover of this fact. []

In the following the fact retrieval is discussed step by step.

(i) Constructing the fact retrieval plan.

In order to construct a fact, we need to first to form a plan to indicate what documents should be retrieved, the order in which these documents are to be retrieved, and which parts of these documents will be used. From theorem 5.1 comes the following definition.

Definition (Fact retrieval plan). A fact retrieval plan is an ordered list with each item consisting of the following:

$$[D, Q]$$

where

D - document number, and

Q - the query description list for this document

Intuitively, a fact retrieval plan gives a process "from current existing document stems, how to construct an area so that its boundary contains the query description list." Such a plan should specify what documents are used and in what order these documents are used. The definition contains both these two information needs. The order is determined by time stamp on the documents, that is, the contents contained in the document that enters the system earlier are assumed to have occurred earlier.

The general steps for generating the fact retrieval plan can be stated as follows.

1. Use the method summarized in 5.5.1, search the KRS and convert to DRS to obtain a list of documents which are at least partially relevant to the QDL. This can be carried out by applying the basic idea described in section 4.4 to the data structures of COGMIR1.

2. Choose documents from the resulting list so that after arranging them according to the increasing order of their document reference numbers, they can jointly satisfy the following requirements: (1) any document stem must have some common boundary object names or common object names associated with boundary objects with those appeared in document stem directly before it and after it (this is to guarantee connection between document stems), (2) these objects must contain all the terms appearing in the QDL (not necessarily in the order appeared in the QDL).

3. List the abovementioned documents in increasing order, along with the boundary objects that appear in the QDL, taking the form specified in the definition of the fact retrieval plan.

For example, the fact retrieval plan for the query "heart, capillary" in the introductory example of Chapter 3 can be carried out as follows. From step 1 it can be determined that documents 1 and 3 are partly relevant to the query. Following step 2, these documents should be rearranged so document 1 is followed by document 3. Step 3 gives the following fact retrieval plan:

[[1, [heart, vein, artery]],

[3, [vein, artery, capillary]]].

Notice here that the documents are listed in order, and the objects "vein" and "artery" appear in both of these two items.

(ii) Performing the chopping operation in each document stem involved in the fact retrieval plan.

Each item in the fact retrieval plan specifies part of the boundary of a document stem that is relevant to the query. From the objects involved in this part of boundary, the lowest location number o_{low} and the highest location number o_{high} can then be determined. The object tuples and relationship tuples that are in this document stem but do not fall in a range determined by o_{low} and o_{high} can then be removed. Therefore, the chopping operation is performed by using information provided in the document description list and the fact retrieval plan.

(iii) Constructing the union of document stems from the fact retrieval plan.

The resulting document stems taken from different documents are then listed in order of the documents as specified in the fact retrieval plan. This enables us to perform the union operation on the document stems.

In the introductory example in Chapter 3, a fact about the query "heart, capillary" was constructed. The resulting document stem corresponding to this fact, which is the partial union of the document stems δ_1 and δ_2, which form the D-cover of this fact, is shown step by step in Figure 5.4 (in (a) and (b) blank lines are only used to distinguish different document stems. Notice that in step (c) the tuples related to the object "scientist" (which is not involved in this retrieval) is removed. This is to perform chopping operation. The union operation follows, which results in a new document

stem to be constructed into text form. Figure 5.5(a) is reproduced from Figure 3.1 for comparison, while Figure 5.5(b) graphically explains the process of query invoked memory reorganization, where objects 106 and 121 (with same object name "vein") are conceptually combined as a single object, and objects 101 and 119 (with same object name "arteries") are also conceptually combined as a single object. The new fact is confined in a big rectangle in Figure 5.5(c), the boundary of this fact formed by the objects "veins", "blood", "arteries" and "capillaries".

Object list:	Relationship list:
[[101, [arteries], [], [102]], [103, [blood], [], [102]], [105, [heart], [], [104]], [106, [veins], [], [107]], [109, [bat], [], [110, 114]], [111, [sound], [inaudible], [110, 112]], [113, [obstacle], [invisible], [112, 114]] [115, [scientist], [], [116]], [117, [capillaries], [], [118, 120]], [119, [arteries], [], [118]], [121, [veins], [], [120]]].	[[102, [carry], [], [101, 103]], [104, [to], [], [103, 105]], [107, [carry], [], [106, 103]], [108, [from], [], [103, 105]], [110, [emits], [], [109, 111]], [112, [reflects], [], [113, 111]], [114, [detects], [], [109, 113]], [116, [discovers], [], [115, 117]], [118, [connect], [], [117, 119]], [120, [connect], [], [117, 121]]].

Figure 5.4(a) Original document stems

Object list:	Relationship list:
[[101, [arteries], [], [102]], [103, [blood], [], [102]], [105, [heart], [], [104]], [106, [veins], [], [107]], [115, [scientist], [], [116]], [117, [capillaries], [], [118, 120]], [119, [arteries], [], [118]], [121, [veins], [], [120]]].	[[102, [carry], [], [101, 103]], [104, [to], [], [103, 105]], [107, [carry], [], [106, 103]], [108, [from], [], [103, 105]], [116, [discovers], [], [115, 117]], [118, [connect], [], [117, 119]], [120, [connect], [], [117, 121]]].

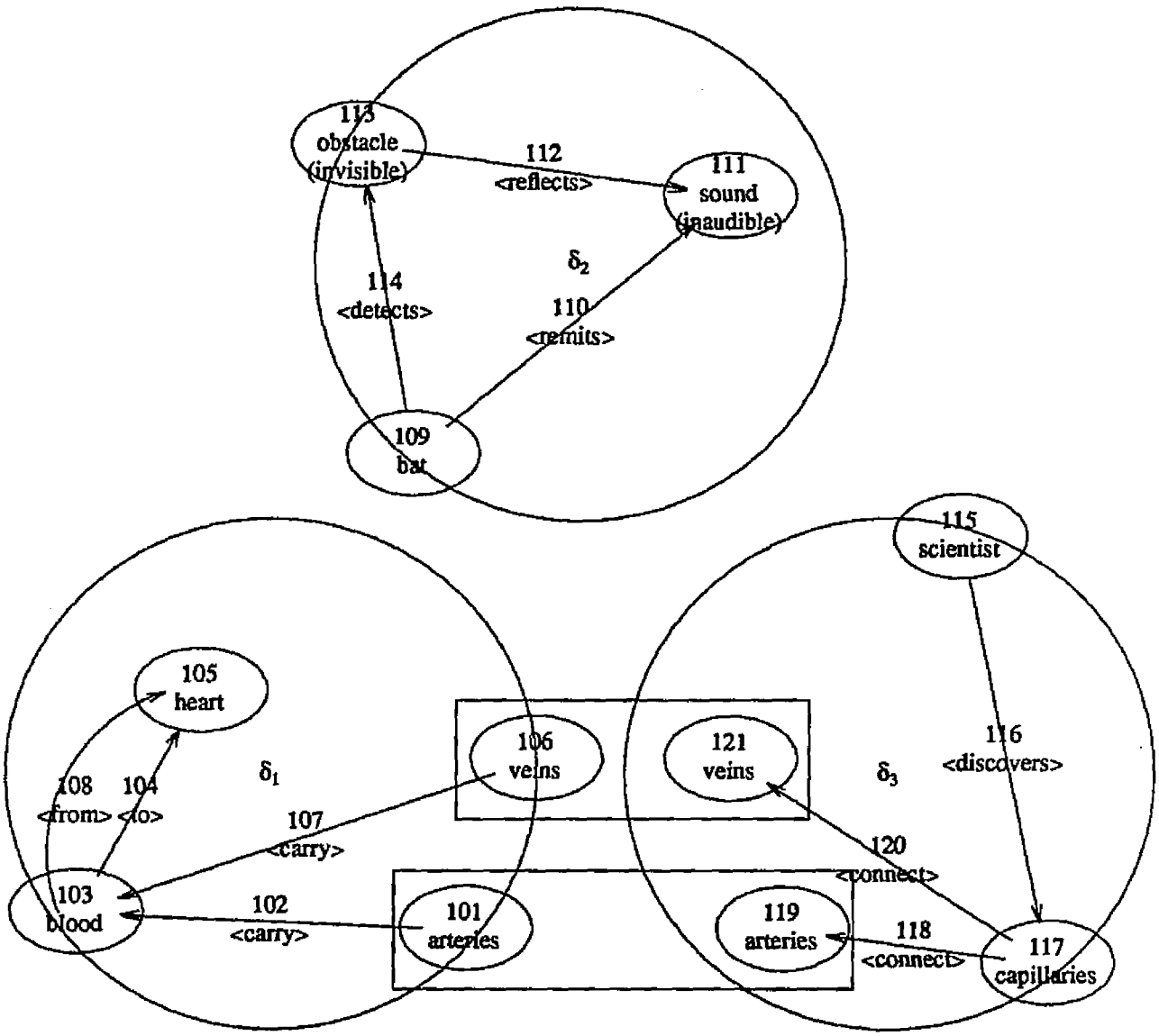
Figure 5.4(b) Relevant document stems (to be performed by chopping operation)

Object list:	Relationship list:
[[101, [arteries], [], [102]], [103, [blood], [], [102]], [105, [heart], [], [104]], [106, [veins], [], [107]], [117, [capillaries], [], [118, 120]], [119, [arteries], [], [118]], [121, [veins], [], [120]]].	[[102, [carry], [], [101, 103]], [104, [to], [], [103, 105]], [107, [carry], [], [106, 103]], [108, [from], [], [103, 105]], [118, [connect], [], [117, 119]], [120, [connect], [], [117, 121]]].

Figure 5.4(c) Union of document stems

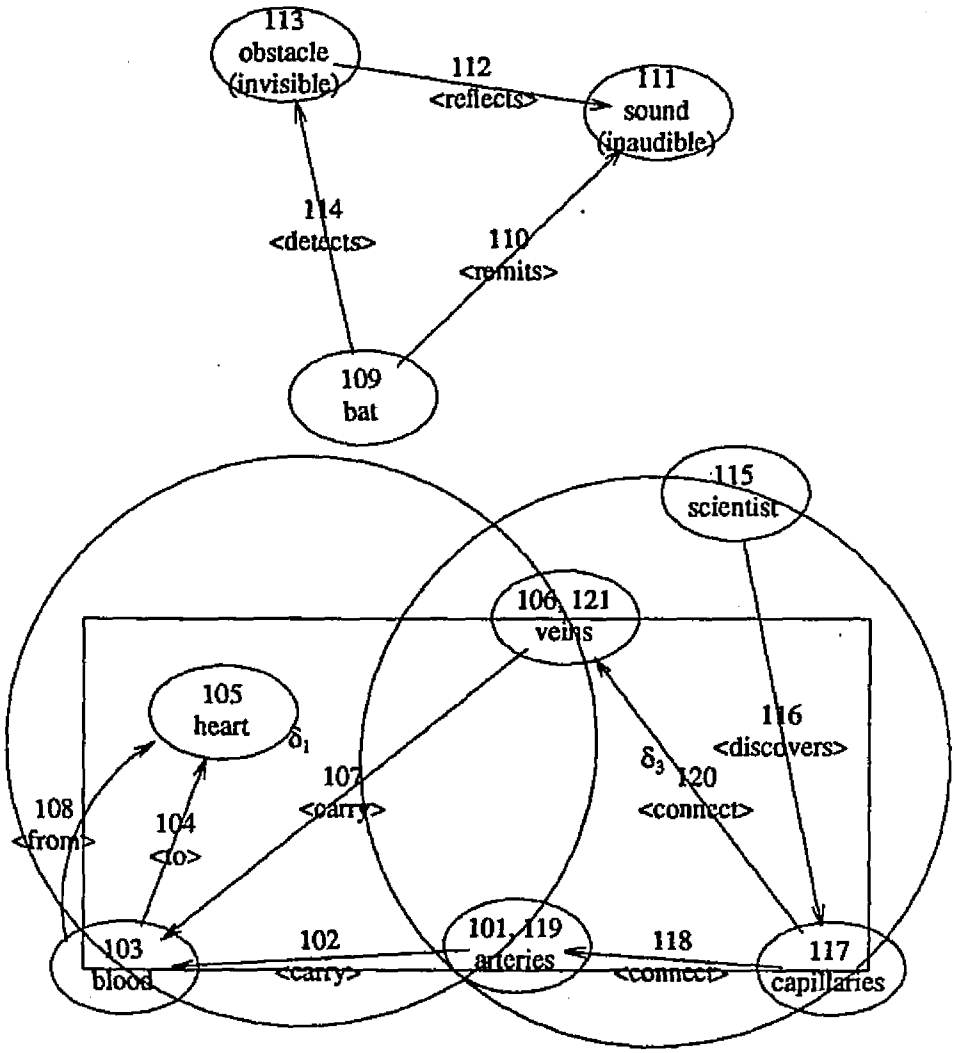
A fact constructed for query "heart, capillaries":

arteries carry blood to heart. veins carry blood from heart. capillaries connect arteries. capillaries connect veins.



Note: attributes for objects in parenthesis

Figure 5.5(a) Three document stems



Note: attributes for objects in parenthesis

Note: Two objects integrated by concept list have been merged together

Figure 5.5(b) Query invoked fact construction for query "capillaries, heart"
(Fact is indicated in the rectangular,
with d_1 and d_3 as its D-cover)

5.5.4 Text construction from document stems

The final step, going from the document stem(s) to a user readable form, is the same for both document retrieval and fact retrieval. Therefore, we will discuss them under the subtitle "text construction." The following steps describe the formation of document or fact from document stems.

The process of construction from a document stem is performed as follows:

1. For both document retrieval and fact retrieval, the user readable form is constructed sentence by sentence. For document retrieval, sentences are constructed according to the order of the object location number as specified in the boundary. For fact retrieval, more than one document stem (each is a part of a document stem converted from a document) is involved, and they are used according to the fact retrieval plan. For each document stem, the reconstruction process is the same for processing in document retrieval. Therefore, although for fact retrieval, we do not use a single "fact description list" (like the document description list in document retrieval) to provide information about its boundary, the required information is available from the document description list for the documents involved in the D-cover.

2. Each sentence is constructed in the following manner.

   (i) For each object (actor) location number in the boundary, get the name of the actor and find its object tuple in the object list;

   (ii) from this object tuple, find the related relationship location number;

   (iii) from the relationship location number, get the name of this relationship, and find the relationship tuple in the relationship list;

(iv) finally, from the relationship tuple, find the object (actee) location number, and from the object list find the tuple of this object to get the name of the actee.

3. Based on the information provided by the tuples obtained in step 2, write out sentences according to grammar G. Basically this is done by listing the names of the objects and relationships according to the following order:

(Art) actor_name relationship_name (Art) actee_name.

Where "Art" is an optional article which takes value "a/an" when the associated object (actor) appears in the document in the first time, or "the" elsewhere. "Is/are" type sentences are constructed after the first appearance of the actor, if the attribute list of this actor is not empty. Note that the reconstructed documents are not necessary identical with the input previously acquired by the system. For instance, the use of "article" may be different from the original form. But the reconstructed documents are equivalent to the original documents in the sense defined in Chapter 3.

As an example, following this algorithm, for the previous example, from the fact retrieval plan, then the fact constructed is

arteries carry blood to heart. veins carry blood from heart. capillaries connect arteries. capillaries connect veins.

5.6 Knowledge space and relational database

It is interesting to compare our work with other research in information systems. Using the terminology appearing in this chapter, there are two ways to view a document: each document is taken as a unit, or each document is a collection of object tuples or relationship tuples. We already discussed the first point. At another

extreme, as we already mentioned in the beginning of this chapter, since a document can be viewed as a collection of object tuples and relationship tuples, there are some interesting connections exist between our approach and the relational database model. It has been observed that relational databases have not been ideal for AI applications in the past because of their inefficiency in making large numbers of small inferences involving either very small relations or small parts of larger relations [Tani87]. By imposing the COGMIR framework at the system level, we have adopted some considerations from the relational model to meet the needs of knowledge integration.

(1) Frame-like lists resembles relational databases.

First of all, the frame-like lists as used in this case study can be roughly viewed as relational databases with fixed fields of attributes. For instance, the underlying structure of the knowledge space can be viewed as a relational database. All the object tuples form a relation *Object*, which has fixed fields L, N, A, R (using the symbols in section 5.2); and all the relationship tuples form a relation, *Relationship*, which has fixed fields L, N, A, A_r, A_e (again using symbols in section 5.2). Moreover, any document stem is also a relation. Similarly, the concept term list forms a relation, as does the document description list. But in our approach, instead of a tabular representation, a list representation is used. This is due to the different manipulation requirements between *data* and *knowledge*. But the main advantage of using a list representation over the tabular form is that it is more flexible in realizing the manipulations as described in this chapter. By mapping the input documents into a frame-like list representation which is much more regular than that in the original documents, the power of manipulating a regular, homogeneous structure, as that demonstrated in a

relational database, is adopted.

(2) Operations in knowledge space resemble those in relational databases.

The operations on these lists are similar to the operations on relational databases. For instance, the construction of a path in the knowledge space can be viewed as a kind of *join* operation, and the construction of a fact may be viewed as a result of a series of operations which resembles select and join operations in a relational database. The conceptual term memory resembles a data dictionary in a database, which is a database about a database i.e., a database that records information about the objects in the database [Gran87]. Moreover, to deal with conflicting information (a problem not addressed here) resembles the integrity problem in a conventional database. Nevertheless, an essential difference exists; for instance, with the document stems, there is nothing similar to "keys" in a relational database.

Since the knowledge space in COGMIR1 is a net-like structure, one may also ask whether there is any relationship between the network database model and COGMIR1 at all. In order not to digress from our current topic, we will not examine this in detail, although we want to point out that differences exist in the way of performing database navigation [Gran87], to which the COGMIR framework provides a way of searching through an intermediate conceptual memory (conceptual term list).

The relationship between a knowledge base and a database that is examined here can be viewed from several different levels. One may view the knowledge bases as differing from databases in that they "describe and operate on classes of objects rather than on individual objects" [Wide86]. We can view the problem from the "knowledge level" [Newe77] so that databases may be interpreted as large knowledge bases of a

certain limited form [Brac86]. Keeping this kind of relationship in mind, query invoked memory reorganization may be viewed as the "knowledge base version" of the same principle in database. But due to the essential difference between a database and a knowledge base, the actual algorithms involved are quite different.

(3) Integration in knowledge space resembles integration in multiple databases.

Other recent work in database systems also addresses similar problems. Recently integration has been a topic in multiple databases [Motr87, Litw84, Litw86]. In particular, Litwin's MALPHA [Litw84, Litw86] defines a kind of dynamic attribute which is pertinent to a user query and then it is "disappeared" when the query terminates. This shares a fundamental notion with our query invoked memory reorganization, although that research is not concerned with knowledge engineering at all.

5.7 Discussion and summary

In this chapter we have described the structure of COGMIR1, and defined the manipulations on this model. These manipulations mainly fall in two categories: those related to document storage and those related to retrieval. Document storage manipulations include constructing a new item in the document description list for a document, updating the knowledge space (which consists of an object list and a relationship list), and updating the conceptual term list. On the other hand, there are two problems to consider for the task of retrieval. First, since what are stored are not original documents, but are their correspondent document stems integrated into the knowledge space, the search is conducted in the knowledge space, while in order to reconstruct the documents, this kind of information must be converted back into documents. The other problem is to actually reconstruct these retrieved documents. In this

chapter, these two problems have been addressed. Particularly, we have emphasized the role of the conceptual term list and the document description list in the manipulations. What is more, we have examined the manipulations related to the fact retrieval process; through this process, memory has been reorganized. All these manipulations realize the principles as discussed in Chapter 3. Sample output can be found in Appendix B.

If we view each document stem as a structure pattern, then the algorithms in this chapter describe a process of constructing and manipulating patterns which take documents as basic units to work with. Moreover, there is an inference ability associated with theses patterns, as will be discussed in the next chapter.

Some traditional issues are not discussed in this chapter. This includes updating and deleting. However, more issues related to inference on the COGMIR1 structure will be discussed in the next chapter.

CHAPTER 6

QUERY INVOKED STRUCTURE GENERATION

6.1 Introduction

The fundamental idea of query invoked memory reorganization in COGMIR and its restricted version COGMIR1 has been examined in previous chapters. But knowledge integration not only means tying things together, but also implies utilization of acquired knowledge in a holistic manner, usually requiring some kind of inference. An information system without an inference ability is very restricted. One indication of inference for a retrieval system is that it can provide information which was not previously stored. In this chapter we will show how COGMIR1 can be used to generate new structure based on available knowledge, an inference method which is usually referred to as analogical reasoning. Analogical reasoning is closely related to, but not identical to, inductive inference [Holl86, Jant87]. Efforts have been made to provide a formal framework for analogical reasoning [Klin71, Chou83, Indu87]; however, many aspects of analogical reasoning are still to be explored. What we want to do in this chapter is to explore the nature of analogical reasoning from the perspective of an integrated cognitive process.

We will treat analogical reasoning as as a natural extension of information retrieval. Although it is generally agreed that analogical reasoning concerns the *reminding* of current situation to some previous one [Carb85], the key point of *how to find these previous situations* (i.e., the analogs) is far from being solved: so far as this aspect is concerned, it has been simply artificially assumed that they were already

there. However, locating the analogs is the most important aspect which uncovers the underlying cognitive process of analogical reasoning, and this problem cannot be solved without considering the environment of intelligent retrieval. The COGMIR environment has made this possible. Due to the large amount of work involved in analogical reasoning, instead of a developing a comprehensive theory of analogical reasoning, we will only focus on the facet of analogical reasoning as extended knowledge retrieval.

We start our discussion with structural mapping in COGMIR1. We will consider the queries which take the following form: they look like regular inputs in COGMIR1, except that they end with a question. These inputs, except the last sentences (i.e., the questions), consist of some objects and relationships describing known condition, therefore, they provide structure information for part of the area to be retrieved. Moreover, following the algorithm given in the next section, a partial boundary can also be obtained from this input. Therefore, this kind of input is a type (ii) query as defined in Chapter 4. The following is an example of such a query:

the enemies dispatch a plane. the plane is invisible. how people detect the plane.

Suppose the answer is not available from the knowledge space, but a document about "bat" was previously acquired by the system, then COGMIR may perform a structure mapping which results in a *suggestion* to the user for consideration. Notice that this kind of reasoning never "guarantees" to provide a "correct" solution. The generated suggestion is stored in the temporary area reserved in the knowledge space as defined in Chapter 3. It is the responsibility of the user to determine whether to store the solution or not. If the user wants to store the suggestion just as if it were

acquired knowledge, this generated knowledge is treated as a document stem, and is assigned a document reference number. The conceptual term list is also updated. By this way, the knowledge space is self-updated.

The entire process of generating suggestion is to be performed as two steps: construct a document stem $\Delta(Q')$ using trial retrieval and test (to be discussed in this chapter), followed by Φ ($\Delta(Q')$, Ψ), where Q is a list of query terms (directly available or obtained from the problem statments), Q' is another list which related to Q, where Φ ($\Delta(Q')$, $\Psi(Q)$) is the process of generating new structure $\Psi(Q)$ based on retrieval result of $\Delta(Q')$. Using the conventional terminology of analogical reasoning, $\Delta(Q')$ can be called as an *analog* (consequently, the problem implied in the query can be called a *target).* Unlike the case considered in Chapter 5, in the current case *what to be retrieved* is not precisely provided by the query (as in the conventional case). The query Q' itself is something unknown, and usually the analog $\Delta(Q')$ is obtained through a series of "trial retrieval and testing." For instance, in the previous "plane" example, Q is a query concerns the detecting of a "plane", and Q' is not specified in the original query, through a process of "trial retrieval and testing", finally, the document related to "bats" is retrieved.

To summarize, the tasks of this chapter are: (1) process the query (i.e., incomplete document) into internal structure, (2) revise the algorithms given in Chapter 5 for document retrieval and fact retrieval to the case where a query Q' is not specified as illustrated above, and (3) develop the algorithms for structure comparison and generation. However, these tasks are intertwined, so we will not discuss them separately.

6.2 Mapping incomplete document into temporary area

Using COGMIR1 to provide suggestions for the problems as discussed before requires additional data structures and operations. These data structures and operations parallel those of taking documents as input as described in the previous chapters. The difference is that the query (taking the form of an "incomplete" document) is mapped into the temporary area in the knowledge space.

First, we want to define what is an incomplete document. In Chapter 3 we have defined the temporary area in COGMIR, and in this section we want to show how to map the incomplete document into this temporary area.

Definition (Incomplete object/relationship). An incomplete object/relationship is an object or a relationship with name "?".

For instance, the following is an incomplete object:

[-4, [?], [], [-2]],

which indicates an object with unknown name (at temporary location -4) is associated through relationship (at temporary location -2); while the following is an incomplete relationship:

[-2, [?], [], [-4, -1]],

which indicates a relationship with unknown name (at temporary location -2) associates two objects (actor at temporary location -4 and actee at temporary location -1). The reason of having a negative number as temporary location number will be explained before the end of this section.

Definition (Incomplete document stem, or IDS). An incomplete document stem is a

collection of objects and relationships with some of them being incomplete.

Incomplete document d and incomplete document stem δ are denoted $d-$ and $\delta-$, respectively.

Parsing the incomplete document is similar to parsing a regular document. (1) A description list for this incomplete document which will be referred to as a *query description list*, is constructed in the same way as for a document description list. (2) If a sentence is not a question, we use the method described in Chapter 5. If a sentence is a question, on the other hand, we change the sentence from the original form into a form in which an object with unknown name "?" and a relationship with unknown name "?" are involved, and then use the abovementioned method. The document stem converted from the incomplete document is stored in the temporary area separated from the knowledge space. There will be a sequential location number used to impose a time stamp, but here the location number is assigned in a different way. For each different incomplete document stem, we start from 0, and decrease by 1 (instead of starting at 101 and increasing by 1 in the rest of the knowledge space). This is why the temporary location number will always be negative.

6.3 Steps of new structure generation

Now we discuss the general process of using analogical reasoning to generated suggestions. In order to discuss the general process of generating suggestions, an incomplete document stem $\delta-$ (i.e., the internal structure of the incomplete document $d-$ provided by the user) will be denoted

$$\delta- = \delta_0 \cup \psi,$$

where δ_0 is a document stem not involved in the structure mapping, and ψ is the

document stem involved in the structure mapping. δ_0 and ψ are not interchangeable, because of the time stamp.) In fact, δ_0 is the unmatchable part of a document stem to be compared, and it varies for different document stems. For instance, in the "plane" example, "enemies dispatch plane" is not involved in structure mapping, supposing that the knowledge space still hold the three document stems as shown in the example of Chapter 3.

6.3.1 Retrieval of structure from single document

The basic idea of structure generation in COGMIR1 can be shown in the case where the structure to be mapped is from a single document. In this case, the pseudo fact is generated by mapping a document to the incomplete document.

Since structure comparison is time consuming, in COGMIR1 it is restricted to a small number of most recently acquired documents. For document retrieval this process can always start from the last document; if it is not successful, go back to a less recent one. This process will stop when an analog is found or none of the documents to be checked is qualified to be an analog. The range of documents to be checked will be referred to as the **checking range.** Retrieval of an analog from a single document is a process of "trial retrieval and testing" which consists of the following steps:

1. Construct an original checking queue which contains documents in the checking range; these documents are ordered according to the document reference number, from highest to lowest. The process of checking will follow the strategy of "last-in-first-check" (LIFC).

2. If the checking queue is not empty, take the first document in this queue and go to

step 3. Otherwise, report "structure generation cannot be performed based on currently available knowledge," and quit.

3. Use the boundary information of this document stored in the document description list (incorporate conceptual term list, if necessary), perform structure comparison (to be discussed in section 6.4).

4. If this document is similar (or partially similar) to the incomplete document (using the structure comparison method described in section 6.4), then the "trial retrieval and testing" process is successfully done and quit. Otherwise, remove this document from the checking queue and go to step 1.

In the previous "plane" example, since the number of document stems stored is small, all the three document stems can be used as checking range. Following this algorithm, we start with document d_3. Since it is not similar to the query, we go back to document d_2. The similarity can be determined, and d_2 is the document that should be retrieved.

Comparing the algorithm with the algorithm for document retrieval appearing in Chapter 5, the difference is in the way of determining the relevance of a document. Unlike the algorithm in Chapter 5 where the relevance of a document is determined by searching the conceptual term list, here to determine relevance, a testing process for structure similarity is required.

6.3.2 Retrieval of structure from several documents

The basic idea described in the last subsection can be generalized to the case where the structure to be mapped is constructed from several documents from which a

fact can be constructed. Since our purpose is to illustrate our general idea, we will consider the case in which only two documents are involved in constructing the analog. The basic idea discussed below can be generalized to a more general case in which more documents are needed.

Generally speaking, the need for a second document is due to the fact that the first document can form only part of the analog. In this case, the task is to find d_1, d_2 so that $\Phi(\delta_1 \cup \delta_2) = \psi$, where ψ is the document stem of the pseudo fact. Basically, the method described for the single document (in section 6.3.1) can be used recursively, that is, the first document is chosen as before, and the second document is chosen so that it can be used to perform $\Phi(\delta_2) = \psi_2$, where $\Phi(\delta_1) = \psi_1$, $\psi_1 \cup \psi_2 = \Psi$, and $d_1 < d_2$ (to ensure the document reference number must follow in increasing order to impose time stamp). An important difference here is that the LIFC strategy cannot be used. Instead, within the checking range, checking is performed from the lowest reference number to the highest. This is from considerations of time and causality.

Comparing the algorithm for fact retrieval as appearing in Chapter 5, the main difference is that in the current algorithm, no explicit fact retrieval plan is needed; instead, a testing process is used, and structure comparison is required. However, the basic operations described in Chapter 5, such as chopping and union on the document stems, are still important in constructing a fact.

To summarize the case in which more than one document is involved in forming an analog, the basic difference is that the fact retrieval plan cannot be constructed in advance. This is because that in the current case, the query Q' is not specified. Nevertheless, a fact is still constructed through checking the conceptual term list and

the document description list, although structure comparison incorporating with "trial retrieval and test" is needed. Using the symbols in section 6.1, retrieval through analogical reasoning can be summarized in Figure 6.1.

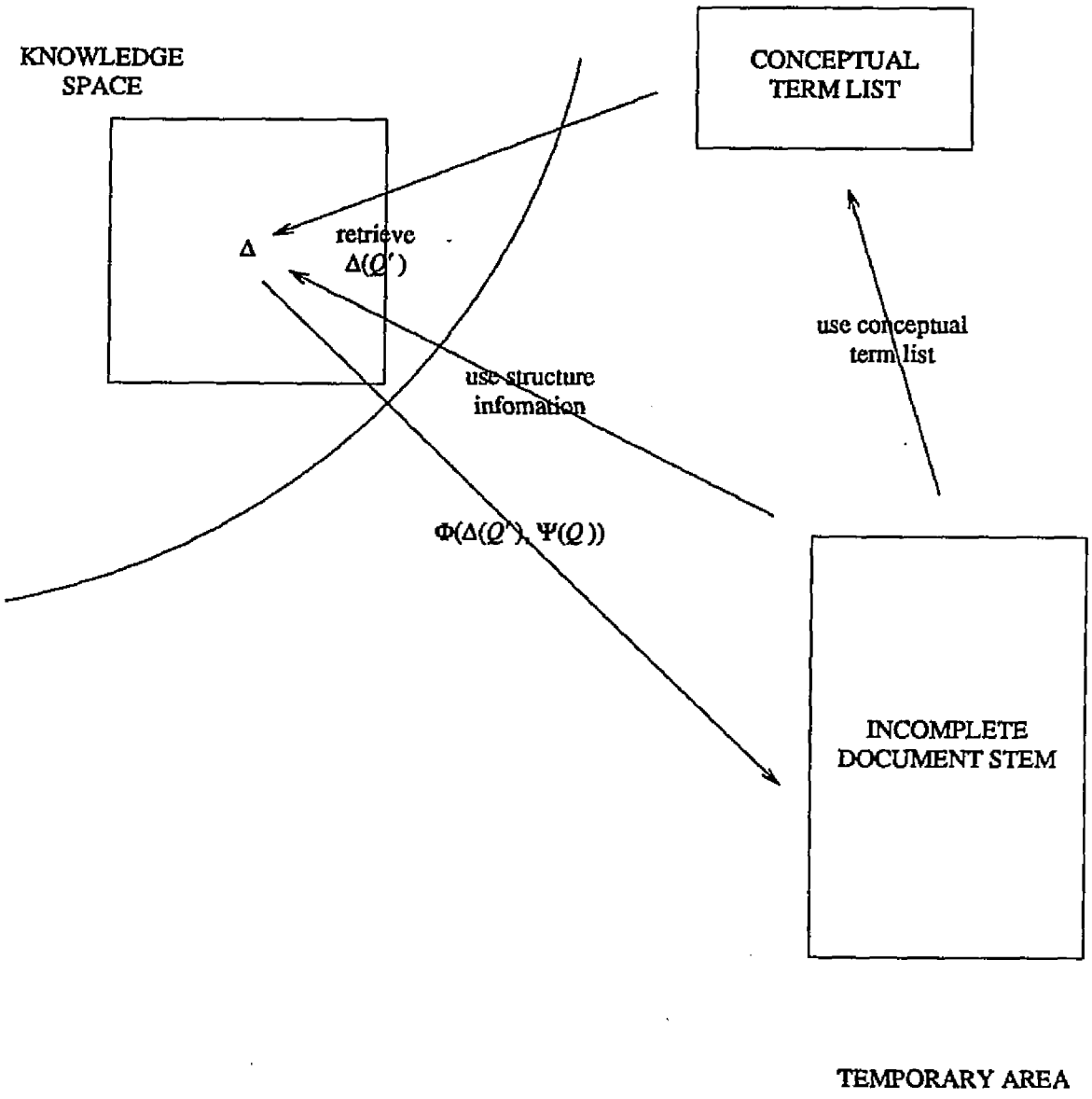


Figure 6.1 Analogical reasoning as extended retrieval

6.4 Structure comparison

In this section, we discuss the actual process of structure comparison mentioned in the previous section. But first we will define the actual form of the mapping function as first discussed in Chapter 3. A **structure mapping function** specifies how the structure of a document stem is mapped to another.

Definition (Structure mapping function). A structure mapping function Φ specifies an area mapping between areas A and A*, if it is able to construct a one-to-one mapping between the objects and relationships in A and A*. That is,

$$\Phi(o_i) = o_i^*,$$

where objects $o_i \in \Delta$ and $o_i^* \in \Delta^*$, and

$$\Phi(r_i) = r_i^*,$$

where relationships $r_i \in \Delta$ and $r_i^* \in \Delta^*$, and Δ and Δ^* are document stems in target domain and the related domain, respectively.

Obviously, this definition meets the basic requirements given in the definition of structure mapping in Chapter 3.

6.4.1 Deciding object similarity

A general scheme of deciding object similarity *sim* can be stated as follows:

1. If two objects have same name, then they are treated as the instances of a same object; assign sim = 1.

2. If two objects are in a same reference range of a concept, assign sim = W ($0 \leq W \leq 1$).

3. Compute similarity determined by attribute lists sim* (see below), and if sim* $\geq \Theta$ (where Θ is a pre-determined threshold), assign sim = Θ; otherwise assign sim = ε (a small number close to 0), where $0 \leq \varepsilon < \Theta \leq W$. The sim* between two objects A and B is defined as

$$\text{sim*}(A, B) = \min(c / a, c / b),$$

where c is the number of common attributes of the two objects, and a and b are the number of total attributes of objects A and B, respectively.

4. Otherwise assign sim = U (unknown). The purpose of assigning U for some pairs is to give a chance to the pairs whose similarity cannot be decided immediately. Its use will be discussed in section 6.4.2.

This strategy assigns highest similarity measure to object pairs with the same names, smaller measure with less similar object pairs.

6.4.2 Similarity between two parts of document stems

Now that we have discussed similarity between pairs, we now can describe how to determine similarity between two document stems:

(1) The order of checking similarity between objects and relationships follows the same process of document reconstruction. That is, for each document stem, we start from some item in the boundary.

(2) The similarity between relationships in these two documents is of fundamental importance. Two relationships in these two document stems (also called *relationship pairs)* are considered similar if they have exactly the same name, or if they can be treated as similar by using the rules provided in the previous subsection. Since in

COGMIR1 we do not include a rule base, we will limit our consideration to the same exact names. The similarity between relationships will be treated as ε if the similarity does not hold.

(3) The similarity between two objects in two document stems (or *object pairs)* are determined through the scheme stated in section 6.4.1. Those objects with unknown similarity U but are associated with relationship pairs that are similar will be considered as similar.

(4) The overall similarity between two document stems are the minimum similarity among all the involved object pairs and relationship pairs as stated above.

(5) Check the overall similarity after every relationship pair comparison. Abort the comparison as soon as the similarity is determined under a certain threshold. Repeat the same process by starting to compare different objects in the boundaries of the two document stems, or choosing some other document stems to compare. The threshold Θ used here to control the similarity is usually a pre-determined number $\Theta \in (0, 1]$ and usually close to 1.

Choosing the proper parts of two document stems to compare follows the objects appearing in the boundary. If the first object to be compared in the boundary of a document description list is not similar to the object to be compared in the incomplete document stem, and their associated relationship pair is not similar either, then the next object in the boundary of the document description list is selected to try. For instance, suppose the query description list Q is

$$Q = q_1, \ldots, q_m,$$

while the item in the document description list for a given document is

$$D_R = d_{R_1}, \ldots, d_{R_n},$$

then q_1 will be compared with d_{R_1}, and if the similarity cannot be determined, d_{R_2} will be used to compare with q_1, and so on, until the similarity is detected or the effort of comparing Q with D_R is abandoned. Notice here that although boundary objects are used to start the structure comparison, objects on the boundary as well as those not on the boundary are used for structure comparison. This process is either ended with a similarity between two document stems, or the comparison fails and the next document (with a higher document reference number) is selected to try.

6.4.3 Constructing mapping function

If we denote a document stem to be constructed as $\Psi = \psi^* \cup \psi^{**}$, where ψ^* is the incomplete document stem, ψ^{**} is the remaining part of Ψ, and a document stem for the analog is $\Delta = \delta^* \cup \delta^{**}$. There is a necessity to distinguish ϕ (which indicates an existing correspondence between δ^* and ψ^*) and Φ (which indicates an extended correspondence between Δ and Ψ). Since the mapping is one to one, the inverse of function ϕ exists, and will be denoted ϕ^{-1}. To retrieve a proper δ^* from Ψ^* is the process of finding this ϕ^{-1}.

The process of constructing the mapping function Φ takes the following steps.

(1) Starting from $\Phi = \{ \varnothing \}$ (empty mapping).

(2) For those object pairs and relationship pairs p_i, r_i (where p_i is an object or relationship in δ and r_i is an object or relationship in ψ^*) that are determined similar, construct $\phi(p_i) = r_i$.

(3) Extend ϕ in Δ to Φ in Ψ.

After the structure similarity is determined, we are able to generate a new structure. The generated structure is an extension of the original structure.

To illustrate the basic idea of this process, let us go back to the example appeared in the beginning of this chapter. The query is

the enemies dispatch a plane. the plane is invisible. how people detect plane.

And suppose the following document was previously acquired by the system:

a bat emits sound. the sound is inaudible. obstacle reflects sound. the obstacle is invisible. the bat detects obstacle.

In this example, the original function as well as the extension of this function are shown below.

$$\Phi \text{ (detects)} = \phi \text{ (detects)} = \text{detects}$$

$$\Phi \text{ (obstacle)} = \phi \text{ (obstacle)} = \text{plane (reason: sim* = 1)}$$

$$\Phi \text{ (bat)} = \text{people}$$

$$\Phi \text{ (emit)} = \text{emit}$$

$$\Phi \text{ (reflects)} = \text{reflects}$$

$$\Phi \text{ (sound)} = \text{sound-like}$$

Notice here that new relationship names are generated through mapping by assigning them the same names from which they are mapped, while new object names are assigned by object names from which they are mapped and concatenated with a postfix "-like".

A picture that illustrates the involved structure mapping is shown in Figure 6.2,

where two document stems d_2 and Ψ are indicated in two circles, relationships in Ψ represented by dashed arrows that indicate generated relationships. The arrows between two document stems (associated with ϕ or Φ) indicate the structure mapping.

In COGMIR1, the information about the function Φ is contained in a data structure called *NQ* list.

Incomplete document (query):

enemy dispatch plane. plane is invisible. how people detect plane.

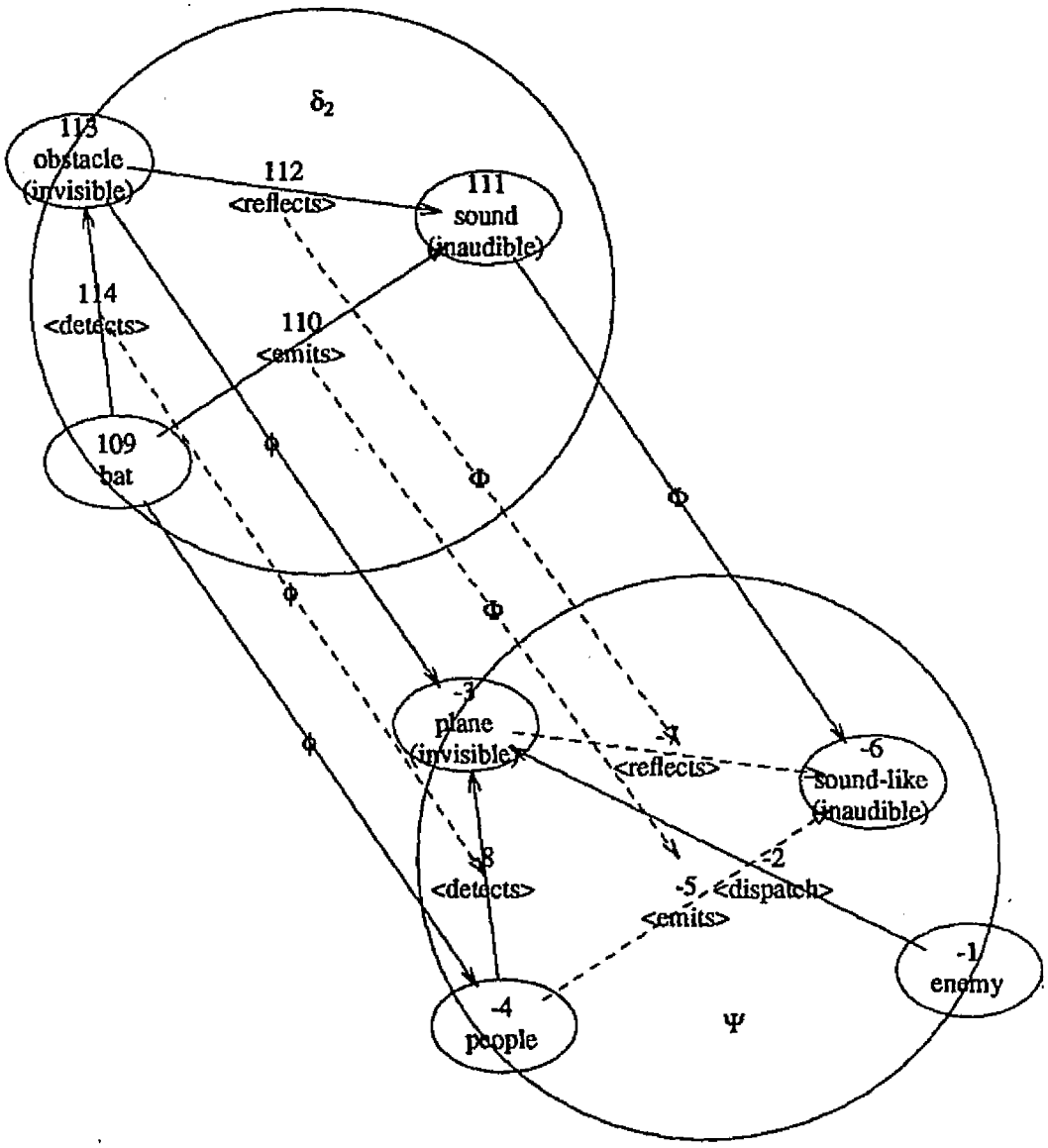


Figure 6.2 Structure mapping in "bat" example

Pseudo fact (generated suggestion):

enemy dispatch plane. plane is invisible. **people emits sound-like. sound-like is inaudible. plane reflects sound-like.** *people detect plane.*

This example can be generalized to the following definition.

Definition (Extension of function ϕ, restriction of function Φ). A structure mapping function Φ is said to be an extension of the structure corresponding function ϕ, if ϕ is defined on IDS $\psi^* \in \mathbf{K}$, Φ is defined on $\Psi \in \mathbf{K}$, $\Psi \supset \psi^*$, Φ agrees with ϕ on ψ^*. We may also say ϕ is a restriction of Φ on ψ^*, denoted $\phi = \Phi|\psi^*$. Here "Φ agrees with ϕ on ψ^*" means following:

$$\Phi(x) = \begin{cases} \phi(x) & \text{if } (x \in \psi^*) \\ [x\text{–}like] & \text{if } (x \notin \psi^*) \cap (x \textit{ is an object}) \\ x & \text{if } (x \notin \psi^*) \cap (x \textit{ is a relationship}) \end{cases}$$

The algorithm discussed in this section is summarized in Figure 6.3.

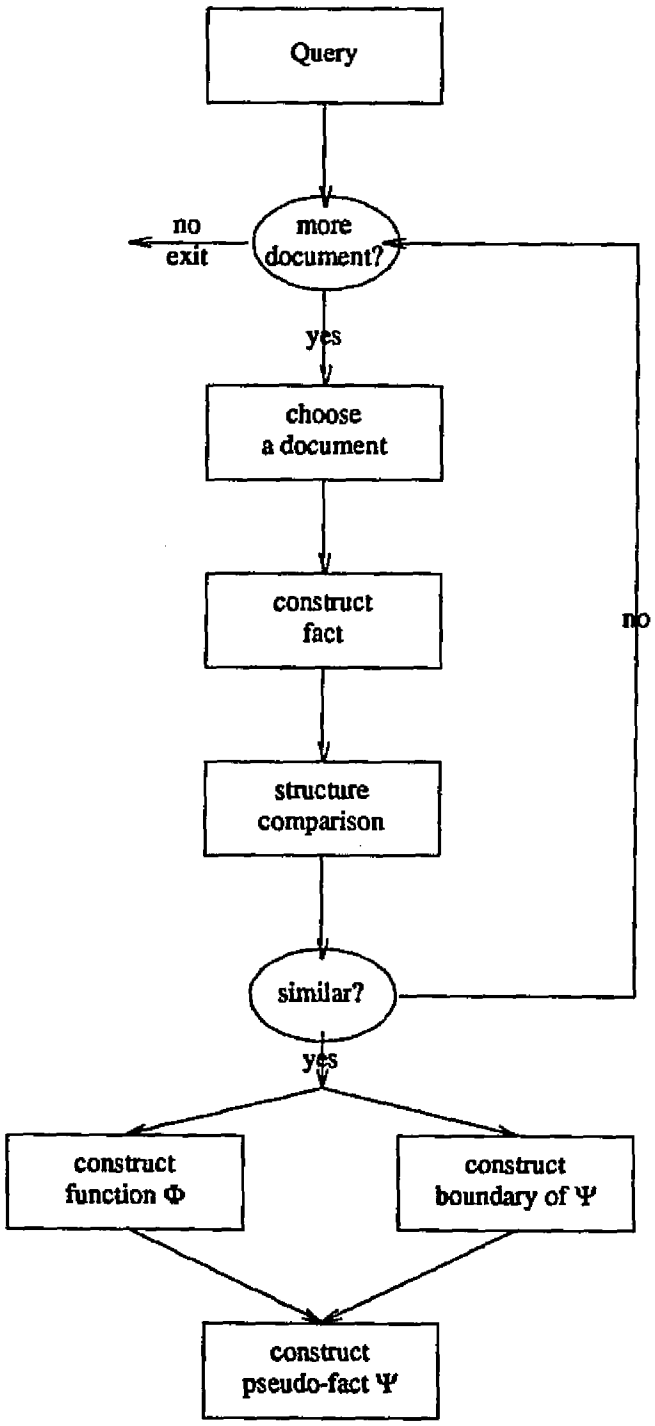


Figure 6.3 Steps for generating suggestions

6.5 Discussion and Evaluation

To summarize what has been discussed so far in this chapter, the process of "trial retrieval and testing" realizes the idea of eliminating blind searching for an analog through the framework of intelligent retrieval as described in previous chapters. The retrieval schemes in COGMIR1 are extended by incorporating the idea of performing structure mapping on the structures of the retrieved document stems.

We must admit that the example used in this chapter is simple, and it is easy to generate new structures (i.e., suggestions) which do not have any meaning in the real world. However, if the documents can be made to contain more detailed information, more meaningful insight about the structure can be captured; consequently, more sophisticated structure comparison and generation can be performed. This will make structure generation closer to real world situations when analogical reasoning is actually used. In addition, our scheme can be used to incorporate considerations of incremental analogical reasoning [Burs85] to improve the quality of analogical reasoning. Therefore, although the algorithms given in this chapter are simple, they can be refined and can be used as a starting point for a new direction of research in analogical reasoning.

An important note to analogical reasoning follows. In Chapter 5 we pointed out that manipulating the document stems, in a sense, is manipulating the patterns; analogical reasoning on document stems can thus be viewed as generating new patterns through inference which requires analogical reasoning.

In the rest of this section we consider the following two questions which can be viewed as an evaluation of the algorithms appearing in this chapter.

(1) Under what condition new structure can be generated? We answer this question through the following theorem.

Theorem 6.1 If a query takes the form of an incomplete document and can be rewritten (using grammar G) so that structure similarity exists between the incomplete document stem and a portion in the knowledge structure, then a new structure can be generated.

Proof. The purpose of rewriting the input is to make the internal structure of the input match part of the existing structure. From the portion of the knowledge space as stated in the theorem, a fact D can be constructed. Using D as analog and following the algorithm given in the previous section, a suggestion can be generated. []

There are two important notes that should be made here. First, this indicates one short coming of our current approach: analogical reasoning is somewhat syntax-driven. The other point is that, unlike what some people might guess, Theorem 6.1 indicates that it is fairly easy to generate analogies in the COGMIR framework. This echoes what happens in natural intelligence: analogical reasoning is prevalent in people's daily thinking [Slee84]. According to current algorithms, COGMIR1 may easily generate unwanted analogy, but this can be controlled by replacing the current structure comparison algorithm with a more advanced one.

(2) Analogical reasoning is slippery. There is a need for evaluate the quality of the suggestion generated through analogical reasoning. How bad the result may be? The following theorem gives us some insight.

Theorem 6.2 The generated structure Ψ does not exceed the D-cover of the fact D used as an analog.

Proof. Suppose Ψ exceeds the D-cover, that is, there exists at least one object or relationship, say $e \in \Psi$, which is not mapped from the Δ (the document stem which correspond to D). Consequently, $\Phi^{-1}(e) \notin D$. But this is contradictory to the definition of the D-cover. []

This theorem states that the width of the structure generation is under control.

An important final remark is that, no matter what suggestions are generated, it is the user who should determine whether need to keep them or not. That is, unlike other situations in which no new structure is generated, here, the relevance is determined by the user.

6.6 Concluding remarks

In this chapter we have shown how analogical reasoning can be treated as a natural extension of information retrieval. Analogical reasoning is prevalent in human intelligence. Our work indicates how our COGMIR framework can deal with this in a manner similar to that done by a human being. No matter whether the result of analogical reasoning is good or bad, the ability for doing this this kind of reasoning is a symbol of intelligence, and this is done in COGMIR framework. Similar to [Lark88], the work in this chapter has shown that knowledge stored in the knowledge space can be used flexibly, with relatively easy generalization or transfer to new domains. But unlike [Lark88] where "general knowledge" is needed to deal with domain related knowledge, in COGMIR this kind of control is built into the structure of the model.

Since COGMIR is extended from a model of classical information retrieval, we have the following comments regarding to the relationship between our work and classical IR. Although there was some previous work using analogical reasoning in

retrieval in classical IR systems (eg. [Naka82]), that kind of analogical reasoning does not concern the generation of new structure. Nevertheless, there is still some relationship between our work and traditional IR. To investigate this kind of relationship, it is helpful to understand the nature of inference in COGMIR1 (and its more general version, COGMIR).

In the classical IR model, RSV (retrieval status value) is used to measure the degree of the similarity between a query and a document. Particularly in the vector space model, the RSV can be viewed as calculated from the query vector and a given document vector. But classical IR is not at all concerned with generating a new vector to stand for new text. In the context of COGMIR, what is done in the vector space model is to restrict comparison only between the boundaries. While in the structure mapping, as discussed in this chapter, the structure is compared through the whole "interior" of the document stems. The method of comparison is also more complex.

To summarize, in this chapter we have demonstrated how to use the COGMIR framework to solve query implied problems by using analogical reasoning in COGMIR1. Here are some of the features:

(1) We have examined the process of query invoked structure generation, or analogical reasoning, in the framework of COGMIR1. We found out that COGMIR1 provides a proper framework to perform structure comparison and new structure generation.

(2) Analogical reasoning is treated as inductive reasoning in the COGMIR framework. In this framework, analogical reasoning can be viewed as an extension of intelligent retrieval. Although it has long been recognized that retrieval of some familiar knowledge is a prerequisite for performing analogical reasoning, the research in

analogical reasoning does not take the process of retrieval into serious consideration. Usually it is assumed that the analogs are already there, while actually they are not, and they must be constructed first. These analogs are constructed in the COGMIR through revised algorithms for document reconstruction or fact construction.

CHAPTER 7

DESIGN ISSUES

7.1 Basic considerations

In Chapters 5 and 6 we described a case study. As mentioned in Chapter 1, the purpose of this dissertation is to explore some problems in knowledge engineering. Basically, we are not aiming for a complete implementation or a commercializable end product. However, algorithms described in previous chapters (along with some other less important algorithms) have been implemented, written in Prolog. This kind of implementation was helpful in fully developing these algorithms.

In Chapters 5 and 6 we only discussed separate algorithms. The purpose of this chapter is to give some additional description for implementing the system as a whole. In this section we consider some general issues that should be considered in an implementation.

An experimental system for implementing the algorithms should provide the following working functions: (1) storage of documents, (2) retrieval of single document as well as fact, and (3) construction of pseudo-facts based on currently available knowledge.

The fundamental task of implementation is to meet the system functions at one end and the data structure at the other. Some desirable features are:

1. Top-down design. At the top level are the system modes. At the lowest level are the most fundamental modules directly dealing with the data structures.

2. Extendable. The structure of the entire program should facilitate later extension.

The result of these considerations is a 3-layer structure. Each component in the top layer stands for a system mode, and each component has several components in the middle layer. The lowest layer consists of some fundamental modules dealing with the manipulation of the lists; these modules can be accessed by all the modules at the middle level. The lowest layer directly deals with the data structure. The middle level modules may deal with the data structures through the lowest level, but may also deal with the data structures directly. The three layers are shown in Figure 7.1 (The modes at the highest layer will be explained soon).

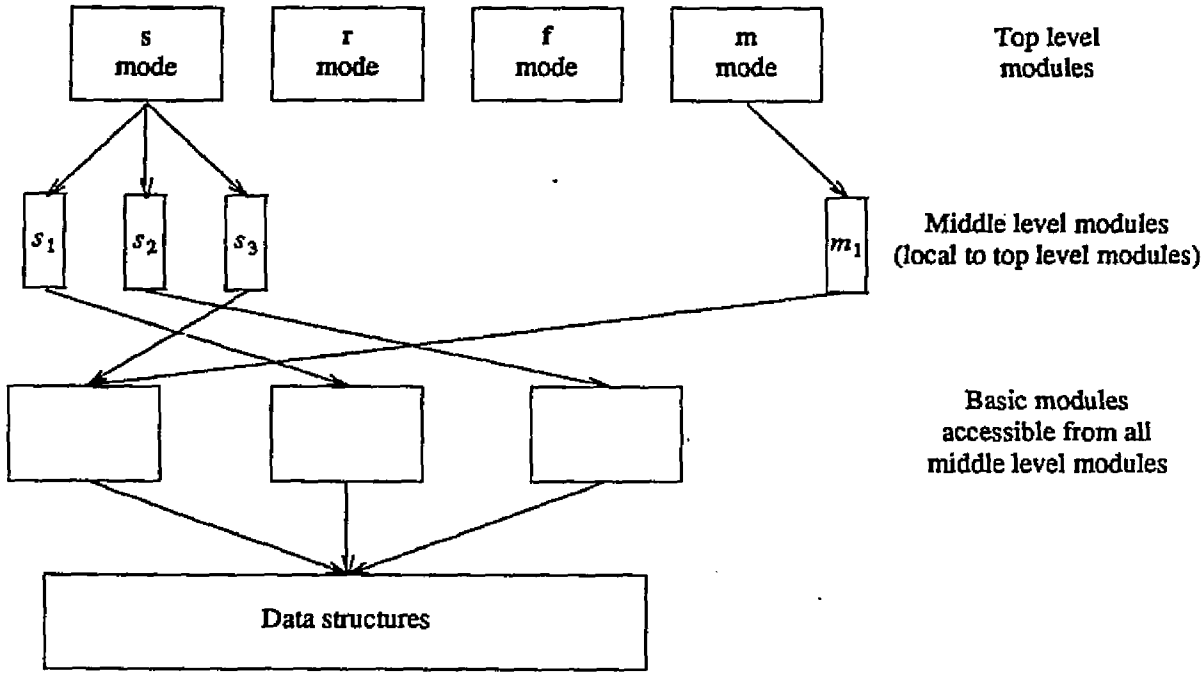


Figure 7.1 Relationship beteeen the modules of the system

7.2 System modes

From a user's view, the system works in hybrid manner: storage and retrieval (including pseudo retrieval). The following are the various modes provided by the system.

(1) s: storage document. In this mode, documents written in grammar G and typed in front of the terminal by the user will be converted into internal form (object tuples and relationship tuples), and these new tuples are attached to the overall object list and relationship list which are stored in outfiles. Other related lists, including the conceptual term list and the document description list, which are stored in some other out files, will also be updated. After these lists are updated, they will be sent back to the files from which they originated.

(2) r: retrieve a single document by query description list. In this mode the user may retrieve a document by specifying query description list. The various lists (such as the object list, relationship list, conceptual term list, or document description list) are consulted by the main program and the retrieved document is displayed on the screen.

(3) f: retrieve a fact by query description list. In this mode, contents from different documents will jointly form a fact to answer the query. Facts are constructed according to the algorithm described in Chapter 5. In the current version, if there is more than one possible way to construct facts, only one fact will be constructed. (This is because the current version is only used to demonstrate our scheme does work. It is not used for "discovery design").

(4) m: new structure generation based on structure mapping. The mode realizes analogical reasoning. The user types in the incomplete document to describe the

situation of the problem, and COGMIR1 will construct a query invoked new structure as a suggestion. However, storing the generated structure into the knowledge space has not yet been implemented. Since analogical reasoning is the final resort when no "real" knowledge is available, the user is encouraged to try more conventional modes first.

7.3 The bare system

This simple system (implemented in Vax 11/780 running on Unix) was developed to carry out some fundamental ideas described in previous chapters. Since the system does not necessarily provide initial knowledge space, that is, it has an empty knowledge space, this kind of system may be referred to as the "bare" system.

The "bare" system includes the following:

(1) The core procedures. These core procedures control the execution of the system, but is extendable.

(2) Some fundamental lists. To facilitate use, in addition to these core procedures, the "bare" system also provides the following:

(a) a pre-defined dictionary, and vocabularies of different parts of words, that is, noun list, verb list and adjective list;

(b) a pre-defined conceptual term list which provides some related terms for some terms which are very likely to be used as "concepts;"

(c) user utilities to modify the noun list, verb list and adjective list.

All these lists are stored as files and are updated whenever a valid document is stored.

When a user enters COGMIR1, he can type in short documents (written in grammar G). Among other things, the system will process these documents into their internal structure in terms of object tuples and relationship tuples. The knowledge space can thus be built gradually.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this concluding chapter we summarize our work, examine our major contributions, mention problems, and discuss some possible future works.

8.1 Contributions

As already mentioned in Chapter 1, this dissertation is an explorative work. To examine the contribution of our work, we first point out that there have been many articles to study the use of AI techniques in information retrieval. This work shows an inverse direction, i.e., using the model of information retrieval in knowledge engineering. Our work is also the first to investigate analogical reasoning from the perspective of extended information retrieval. We have made the following three contributions in this work:

(1) A model. We point out the need for a cognitive model to extend the current IR model into knowledge engeneering and propose a general model which can serve as a framework for integrating and utilizing knowledge. This model serves as an intermediate level between acquired knowledge and the actual data storage. Moreover, in a case study, we further define data structures for the components of the model, and discuss the operations on these data structures.

(2) A scheme. We have developed a new idea of query invoked memory reorganization, which provides a useful scheme for utilizing knowledge in a flexible and holistic manner. That is, although there is only one way to store knowledge, there are potentially numerous ways to utilize it.

(3) Combination of research in intelligent retrieval and analogical reasoning. We have proposed a way to combine the task of analogical reasoning with the task of information retrieval, two areas that were previously separate. COGMIR provides a suitable framework for structure mapping which is crucial for analogical reasoning.

8.2 Summary of the methods

We have assumed that computer science is an empirical inquiry [Newe77]. This includes the following steps.

(1) Formation of assumptions and hypothesis in the subject area. We take the viewpoint that computer science is an empirical science. Based on recent results of cognitive science and some of our own observations, we have formed a set of assumptions used in exploration. The assumptions include considerations from cognitive science, such as limited capacity assumption, biased comparison assumption and integrated processing assumption as discussed in Chapter 2. We also considered restrictions of our model, such as order assumption as given Chapter 5.

(2) Extension of property. We start from the model of information retrieval, extend some of its desired properties to form a more general system, and then demonstrate it in a more detailed prototype system.

(3) Formation of the model. Based on a set of fundamental assumptions and related knowledge, it is possible to establish our framework. A computer model is developed in Chapter 3, which includes the specification and the function of its components, and the fundamental ideas of manipulating this model. According to the criteria given in [Arch75], this type of model can be classified as a theoretical model, because it demonstrates some basic criteria.

8.3 COGMIR as a theoretical model

We examine the features of COGMIR, from the modeling point of view. COGMIR falls in the category of theoretical model [Achi75]. [Achi75] also gives five features to characterize a theoretical model. In the following, we point out how our suggested model meets these features by examing them one by one.

(1) COGMIR is a set of assumptions about describing the process of intelligent information retrieval. (as described in previous section).

(2) COGMIR describes a type of information retrieval (IR) system by attributing to it an inner structure, composition, or mechanism, reference to which is to explain various properties exhibited by this kind of IR system.

(3) COGMIR is treated as a simplified approximation, useful in that it includes actually carrying out some important principles of human information retrieval.

(4) COGMIR is proposed within the broader framework of the theory of information retrieval, and it can be easily extended for further research.

(5) COGMIR is formulated, developed, and employs some terms (eg. conceptual memory) used, on the basis of an analogy between the system described in the model (a computerized information retrieval system) and a different system (human intelligent system).

8.4 Query and memory reorganization

There are some advantages related to query invoked memory reorganization. First of all, knowledge obtained from different documents can be organized together to meet the request represented by the query. This realizes the integration of

knowledge. Requested knowledge is not restricted to any single input document, although the original documents still provide clues to the search of memory. Moreover, knowledge is viewed in a more active manner, which realizes the idea of knowledge engineering. Input documents are not the only source of knowledge accumulation; knowledge can also update itself. Another advantage is that a query serves as a hint to make memory updating possible. Memory without such an updating ability is too passive; on the other hand, updating in a fixed, pre-determined manner is of only very restricted use. A fourth advantage of query invoked memory reorganization is that analogical reasoning, which has been a somewhat isolated research issue, is now treated as a natural extension of knowledge retrieval, i. e., , a kind of "intelligent" retrieval in which that which was "retrieved" was not stored before. Query invoked memory reorganization provides an opportunity of memory updating along the desired direction specified by the query. Finally, new structure can be generated through query invoked memory reorganization.

8.5 Relationship with research topics in AI

In the previous sections we have examined what is new in our work. In Chapter 1 we pointed out that our interest is in the use of IR methods to AI. In the following, we will summarize several aspects of our contributions along the direction of traditional AI research interests.

(1) Knowledge representation. We have introduced an intermediate level for knowledge representation; we have also developed the method of using conceptual memory to organize knowledge stored in the knowledge space into various domains.

(2) Pattern manipulation. It has been noticed that one of problems of Prolog is that pattern matching can become extremely inefficient when the database is large and there are many pattern-directed modules in the program; this kind of efficiency can be improved by a more sophisticated organization of the database, including the indexing of the information in the database, or partition of the information into sub-bases, or partition of the set of pattern-directed modules into subsets [Brat86]. But so far no actual approach has ever been proposed. The COGMIR framework provides a feasible scheme for both indexing and partition; instead of dealing with low level patterns directly, the use of documents (or what we called facts) provides an intermediate level of manipulating patterns. When COGMIR is implemented in Prolog, this kind of indexing and partition can be viewed as a kind of enhancement of Prolog.

(3) Space searching. As a direct consequence of improved pattern manipulation, space searching is no longer restricted to low level such as searching for a single object or a single sentence. Rather, searching is carried out at a higher level.

8.6 Relationship with the core of computer science

The research described in this thesis, although new in concept, is closely related to the core of computer science. In fact, COGMIR is a suitable project to fully utilize computer science knowledge. We already discussed the relationship with classical IR and database systems. Below, we consider the core of computer science.

(i) Relationship with computer architecture. At the highest level is the virtual machine of knowledge engineering extended from the classical model of information retrieval. The basic idea is indicated in the COGMIR model.

(ii) Relationship with language issues. This dissertation project is related to language issues in many aspects, particularly those related to the processing of input documents and generation of new text.

(iii) Relationship with database theory. We have already pointed out the relationship of COGMIR with database theory.

(iv) Relationship with foundations of computer science. This work also concerns some mathematical concepts which are fundamental to computer science; for instance, the set theoretical approach as well as the methodology mentioned in the previous sections.

8.7 Some general observations

(1) More about knowledge integration. In this dissertation, only a few aspects of knowledge integration are investigated; i. e., those that are related to knowledge accumulation. There are many other aspects of knowledge integration, for instance, dealing with conflicting information, that were not addressed here. It is possible to deal with this by simply using a time stamp, ignoring previous acquired knowledge and keeping only the most recently acquired information.

(2) More issues about traditional IR. We have mentioned the relationship with the traditional IR many times. In this dissertation we have only concentrated on the issues that have not been addressed in IR; on the other hand, features closely related to classical IR have been simplified. More issues can be discussed along the direction of IR, for instance, the forming of ranked output for document retrieval as well as fact retrieval. Many useful techniques, such as automatic indexing and constructing of thesaurus, can be incorporated into COGMIR (particularly its restricted version

COGMIR1). In addition, some restrictions which are used for simplify the implementation can also be removed. For instance, in COGMIR1 we have assumed that the documents entered the system according to their document description number. This restriction can be easily removed by using an additional field in each item of the document description list, to indicate the actual order these documents entered.

(3) About parallelism. We agree to view that thought and problem solving have a sequential character when viewed over a time frame of minutes or hours, although each step in the sequence may be the result of the simultaneous activity of a large number of simple computational elements, each influencing others and being influenced by them [Rume87]. Since our model deals with problem solving at a level "higher enough," it is handled in a sequential manner.

(4) The fuzzy nature of information processing.

The research of analogy is closely related to approximate reasoning, which may be viewed as a way of fuzzy reasoning. There are some more considerations related to fuzzy concepts in COGMIR1. For instance, we may consider a fuzzy boundary of a document stem. Moreover, the process of structure comparison and structure generation can be incorporated with a fuzzy function.

The importance of fuzzy set theory in classical information retrieval has been emphasized in [Kraf83, Radi83]. On the other hand, the relationship between analogy and fuzzy set theory or fuzzy logic was discussed in [Bour83, Reis82, delC82].

When the classical IR model is extended to knowledge engineering in which the internal structure of documents is considered, the issue of uncertainty becomes an ever important consideration. Some reasons are: (1) a major consideration is how truthful

the representation represents the contents of the documents; (2) analogical reasoning performed on the internal structure of the documents involves a large degree of uncertainty; and (3) sometimes it is desirable to access the original documents, and since documents are represented in their internal form, they must be approximately reconstructed from these documents; this again involves uncertainty.

(5) The role of the rule base.

The research of rules in AI is flourishing, and the limitation of using the rules has also been discussed by many authors [John83, Gold86, Stan86a, Stan86b]. Although in this dissertation we do not fully explore the use of the rule base in the COGMIR framework, it can be incorporated along the notes in section 3.7.

8.8 Future work

Some considerations related to this dissertation work have been described in [Chen87a, Chen87b, Chen88]. In addition, there are some different directions for future research.

(1) Better language processing ability. If we do not strive to "full" natural language processing, then it is almost "always" possible to improve the ability of handling the restricted English (or English-like sentences).

(2) Text understanding and graphic input. The current version of COGMIR is syntax oriented (although mixed with semantic considerations). Better "understanding" in the COGMIR framework means a lot of additional work, including means to better utilize a rule base. An alternative to text input is to consider graphic input.

(3) More theoretical investigation. Just like some other AI work, in COGMIR1,

efficiency has not been a major concern. Further research around computational problems may result in better data structures and more efficient algorithms. Another aspect is to further formalize our result (see point (5) below).

(4) Consideration related to distributed knowledge sources. As indicated in Chapter 2, we have been interested in the process of integration in the sense that it is related to knowledge accumulation. On the other hand, since the documents may be viewed as knowledge sources, it is reasonable to assume that under certain conditions, the knowledge needed to solve a problem can be treated as the result of a kind of integration on some knowledge sources, and these knowledge sources can be treated as documents. Starting from this viewpoint, we may consider deliberately distributing a problem solving task in a kind of distributed environment. This would mean that processing could be done in a parallel manner.

(5) More explorations on analogical reasoning. There is still much work to be done related to analogical reasoning, especially incremental analogical reasoning. In fact, the structure of the knowledge space in the case study, provides a suitable framework for structure comparison at different levels. In addition, we can consider the relationship of analogical reasoning and logic in the framework of COGMIR, although we recognize the inherent limitation of logic [Pent83].

(6) Knowledge transferring between different knowledge domains. We have allowed documents may from more than one domain. In practice, these documents may from several specified domains. This kind of consideration may lead to the topic of knowledge transference between different knowledge domains (e.g., transferring knowledge from physics to chemistry).

APPENDIX A: SAMPLE INPUT

The following are some short paragraphs, each stands for a possible input (i.e., document) to COGMIR1. Each document is processed into object tuples and relationship tuples and then stored in the knowledge space. The original documents are not stored.

a person has a heart. the heart pumps blood into arteries. the arteries carry blood to heart. the veins pumps blood from heart.

a bat remits sound. the sound is inaudible. the bat approaches an obstacle. the obstacle is invisible. the obstacle reflects the sound. the bat detects the obstacle.

people put stones on treetrunk. stones are heavy. treetrunk rolles on ground. treetrunk carries stones. people move stones.

people use string. string pulls kite. kite is light and flat. airstream hit kite at angle. kite flies through air.

a water clock has a reservoir. the reservoir generates overflowoutlet. the reservoir drips into tank. tank has float. the float carries pointer. the water level lifts a pointer. the pointer indicates time.

a melon has seeds. the melon is round and edible. the seeds are small and hard. engineers design a machine. the machine opens the melon. the machine removes the seeds.

APPENDIX B: SAMPLE OUTPUT

The following is the edited printout of COGMIR1. Input after the symbol "|: " or "| ?" was typed by the user. The response of COGMIR is appeared in upper case letters. Only the comments appeared in { } are edited. The purpose of this editing is to help the reader understand what has happened in COGMIR1 system.

****

WELCOME TO COGMIR SYSTEM

****

IN ORDER TO ENTER A NEW SESSION OF COGMIR,

PLEASE RE-ENTER PROLOG

THEN TYPE [cog].

COGMIR IS NOW READY TO STORE

NEW DOCUMENTS AND INTEGRATE THEM

INTO CURRENT KNOWLEDGE SPACE OR RETRIEVE

INFORMATION STORED IN

THE KNOWLEDGE SPACE.

PLEASE TYPE YOUR OPTION. THE VALID OPTIONS ARE:

d. FOR DOCUMENT RETRIEVAL

h. FOR HELP

m. FOR STRUCTURE MAPPING

r. FOR DOCUMENT RETRIEVAL

f. FOR FACT RETRIEVAL

s. FOR STORAGE

u. FOR UTILITIES

x. TO END A SESSION

yes

| ?- s.

NOW YOU ARE IN STORAGE MODE.

YOUR DOCUMENT WILL BE REFERRED TO AS NUMBER 5

PLEASE INPUT DOCUMENT AS BELOW.

|: people use water. water drives wheels. wheels grind grain.?

PROCESSING . . .

DOCUMENT IS STORED AND INTEGRATED INTO KNOWLEDGE SPACE.

UPDATED LISTS WILL BE SENT TO RELATED FILES.

IN ORDER TO DO SO, YOU HAVE TO LEAVE COGMIR TEMPORARILY.

THANK YOU FOR HAVING CONSULTED COGMIR.

PLEASE REENTER PROLOG FOR A NEXT COGMIR SESSION.

BYE-BYE. SEE YOU SOON.

[Prolog execution halted]

{ The following is a new session of COGMIR invoked from PROLOG.

The "WELCOME" screen is same as before, and is not duplicated below. }

| ?- r.

PLEASE TYPE IN QUERY LIST:

|: tumor doctor?

COGMIR IS NOW FINDING REQUESTED DOCUMENT(S) . . .

RETRIEVED DOCUMENT(S) IS (ARE):

[]

THE RECONSTRUCTED TEXT IS AFTER THE FIRST DOT.

.

{ Empty posting. No document stored is related to the query }

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- r.

PLEASE TYPE IN QUERY LIST:

|: veins heart?

COGMIR IS NOW FINDING REQUESTED DOCUMENT(S) . . .

RETRIEVED DOCUMENT(S) IS (ARE):

[[3,[veins, heart],[]]]

{ Document #3 has all the requested objects, and is constructed below. }

THE RECONSTRUCTED TEXT IS AFTER THE FIRST DOT.

.

person has heart.

heart pumps blood into arteries.

arteries carry blood to heart.

veins pumps blood from heart.

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- r.

PLEASE TYPE IN QUERY LIST:

|: engineer story?

COGMIR IS NOW FINDING REQUESTED DOCUMENT(S) . . .

RETRIEVED DOCUMENT(S) IS (ARE):

[[2,[engineer],[story]]]

{ Document #2 is the only one partially satisfied, which includes

object "engineer" but does not have "story". }

THE RECONSTRUCTED TEXT IS AFTER THE FIRST DOT.

.

engineer design machine.

machine open melon.

machine remove seeds.

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- f.

COGMIR IS NOW PERFORMING FACT RETRIEVAL.

PLEASE TYPE IN QUERY LIST.

|: melon machine?

COGMIR IS NOW TRYING TO FIND A FACT . . .

RELATED DOCUMENTS ARE:

[[1, [melon], [machine]], [2, [melon, machine], []]]

THE CONSTRUCTED FACT IS AFTER THE FIRST DOT.

.

melon has seeds.

melon is watery and round.

seeds are small and hard.

engineers design machine.

machine opens melon.

machine removes seeds.

{Notice that the fact comes from two documents instead of one}

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- m.

COGMIR NOW PERFORMING STRUCTURE MAPPING FOR ANALOGICAL REASONING.

PLEASE INPUT A FEW SENTENCES BELOW USING LEGAL FORMAT.

COGMIR WILL TRY TO ANSWER YOUR QUERY ANALOGICALLY

BASED ON CURRENTLY AVAILABLE KNOWLEDGE.

|: brain has tumor. brain is watery and round.

tumor is hard and small. how doctor remove tumor.?

{ The user types several sentences to describe the

situation. COGMIR will form a fact by performing

query-invoked memory reorganization and generate

new structure to provide a suggestion. }

THE GENERATED TEXT IS AFTER THE FIRST DOT.

.

brain has tumor.

brain is watery and round.

tumor is hard and small.

doctor design [machine,-like].

[machine,-like] open brain.

[machine,-like] remove tumor.

yes

| ?- r.

PLEASE TYPE IN QUERY LIST:

I: people, plane?

COGMIR IS NOW FINDING REQUESTED DOCUMENT(S) . . .

RETRIEVED DOCUMENT(S) IS (ARE):

[]

THE RECONSTRUCTED TEXT IS AFTER THE FIRST DOT.

.

{ Again empty posting. No document stored is related to the query }

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- r.

PLEASE TYPE IN QUERY LIST:

I: bat?

COGMIR IS NOW FINDING REQUESTED DOCUMENT(S) . . .

RETRIEVED DOCUMENT(S) IS (ARE):

[[6,[bat],[]]]

THE RECONSTRUCTED TEXT IS AFTER THE FIRST DOT.

bat emits sound.

sound is inaudible.

obstacle reflects sound.

obstacle is invisible.

bat detects obstacle.

COGMIR IS READY FOR A NEW TASK.

ENTER YOUR OPTION:

yes

| ?- m.

COGMIR NOW PERFORMING STRUCTURE MAPPING FOR ANALOGICAL REASONING.

PLEASE INPUT A FEW SENTENCES BELOW USING LEGAL FORMAT.
COGMIR WILL TRY TO ANSWER YOUR QUERY ANALOGICALLY
BASED ON CURRENTLY AVAILABLE KNOWLEDGE.

|: the enemies dispatch a plane. the plane is invisible. how people detect plane.?

{Similar to the "tumor" example,
the user types several sentences to describe the

situation. COGMIR will form a fact by performing
query-invoked memory reorganization and generate
new structure to provide a suggestion. }

THE GENERATED TEXT IS AFTER THE FIRST DOT.

.

enemies dispatch plane.

plane is invisible.

people emit [sound-like].

[sound-like] is inaudible.

plane reflects [sound-like].

people detect plane.

| ?- x.

END OF A SESSION.

BYE-BYE.

REFERENCES

[Alte85] Alterman, R., "A dictionary based on concept coherence," *Artificial Intelligence*, vol. 25, pp. 153-186, 1985.

[Ande83] Anderson, J.R., *The Architecture of Cognition*, Harvard, 1983.

[Ande84] J. R. Anderson and S. M.Kosslyn (eds.), *Tutorials in Learning and Memory*, W.H.Freeman and Co.,1984.

[Abar87] Abarbanel, R. M. and M. D. Williams, "A relational representation for knowledge bases," in [Kers87], pp. 191-206.

[Bour82] Bourne, L. E. et al., *Psychology: Its Principles and Meanings*, (4th ed.), Holt, Rinehat and Winston, 1982.

[Bour83] Bourrelly, L., E. Chouraqui and M. Ricard, "Formalisation of an approximate reasoning: the analogical reasoning," *IFAC Fuzzy information*, 1983, pp. 135-141.

Bratko, I., *Prolog Programming for Artificial Intelligence*, Workingham, UK: Addison-Wesley, 1986.

[Brod86] Brodie, M. L. and J. Mylopoulos (eds.), *On Knowledge Base Management Systems*, New York: Springer-Verlag, 1986.

[Burk78] Burke, J., *Connections*, Little Brown and Co., Boston, 1978.

[Burs85] Burstein, M. H., "Concept formation by incremental analogical reasoning and debugging," pp. 351-369 in [Mich85].

[Carb82] Carbonell, J. G., *Subjective Understanding: Computer Models of Belief Systems,* Ann Arbor, MI: UMI Research Press, 1981.

[Carb83] Carbonell, J. G., "Learning by analogy: formulating and generalizing plans from past experience,", in [Mich83].

[Carb85] Carbonell, J. G., "Derivational analogy in problem solving and knowledge acquisition," pp. 371-392 in [Mich85].

[Cate87] Cater, S. C., "TIRS: A topological information retrieval system satisfying the requirements of the Waller-Kraft wish list," *Proc. 10th ACM SIGIR Conf.,* pp. 171-180, 1987.

[Chen76] Chen, P. P. S., "Entity-Relationship model: toward a *ACM Transactions on Database Systems,* pp. 1-34, 1976.

[Chen87a] Chen, Z., "Some aspects of a cognitive model for information retrieval," *Proc. Modeling and Simulation,* pp. 905-911, 1987.

[Chen87b] Chen, Z., "A language for modeling users' virtual machine for information retrieval," *Proc. IEEE Languages for Automation,* pp. 74-77, 1987.

[Chen88] Chen, Z., "Building expert systems through the integration of mental models," *Proc. 1st International Conf. AI and Expert Systems,* pp. 751-758, 1988.

[Chou83] Chouraqui, E., "Construction of a model for reasoning by analogy," *Proceedings,* 1983, pp. 169-183.

[Cook86] Cooke, N. M. and J. E. McDonald, "A formal methodology for acquiring and representing expert knowledge," *Proceedings of the IEEE,* pp. 1422-1429, vol. 74, No. 10, Oct. 1986.

[Coom84] Coomb, M. J. (ed.), *Developments in Expert Systems,* London: Academic Press, 1984.

[Coop84] Cooper, W.S., "Bridging the gap between AI and IR," in: van Rijsbergen (ed.), *Research and Development in Information Retrieval (Proceedings of the 3rd BCS and ACM symposium),* 1984, pp. 259-265.

[Crof87] Croft, W. B. and D. D. Lewis, "An approach to natural language processing for document retrieval," *Proc. 10th ACM SIGIR Conf.,* pp. 26-32, 1987.

[Cull81] Cullingford, R. E., "Integrating knowledge sources for computer 'understanding' tasks," *IEEE Transactions on Systems, Man and Cybernetics,* Vol. SMC-11, No. 1, 1981, pp. 52-60.

[Dard87] Darden, L., Viewing history of science as compiled hindsight, *AI Magazine,* Summer 1987, pp. 33-41. (Short version in *Proceedings AAAI-86,* pp. 1146-1147.)

[Date86] Date, C. J., *Introduction to Database Systems,* (4th ed.), Addison-Wesley, 1986.

[Davi86] Davies, R. (ed.), *Intelligent Information Systems: Progress and Prospects,* Ellis Horwood Ltd., New York, NY: 1986.

[Defu84] Defude, B., "Knowledge based systems versus thesaurus: an architecture problem about expert system design," *Proc. ACM SIGIR,* pp. 267-279, 1984.

[Defu85] Defude, E., "Different levels of expertise for an expert system in information retrieval," *Proceedings, 8th ACM SIGIR Conf.,* pp.147-153, 1985.

[Ders85] Dershowitz, N., "Programming by analogy," pp. 395-423 in [Mich85].

[DeJa86] De Jaco, D. and G.Garbolino, "An information retrieval system based on artificial intelligence techniques," Proceedings, ACM SIGIR, pp.214-220, 1986.

[DeJo79] DeJong, G. F., *Skimming Stories in Real Time: An Experiment in Integrated Understanding*, Department of Computer Science, Yale University, New Haven, CT, 1979.

[DeJo83] DeJong, G., "Artificial intelligence implications for information retrieval," *Proceedings, ACM SIGIR 1983*, pp. 10-17, 1983.

[delC82] del Cerro, L. F., "Valid reasoning / classical logic = analogical reasoning / no-classical logic," *Deontic Logic, Computational Linguistics and Legal Information Systems*, vol. II, (Martino, A. A. ed.), pp. 161-165, 1982.

[Ders85] Dershowitz, N., "Programming by analogy," in [Mich85], pp. 169-183, 1983.

[Elio86] Eliot, L. B., "Analogical problem-solving and expert systems," *IEEE Expert*, Summer 1986, pp. 17-26.

[Fahl79] Fahlman, S.E., *NETL: A system for Representing and Using Real-world knowledge*, The MIT Press, 1979.

[Falk86] Falkenhainer, B., K.D. Forbus and D. Gentner, "The structure-mapping engine," *Proceedings, AAAI-86*, pp. 272-277.

[Fike85] Fikes, R. and T. Kehler, "The role of frame-based representation in reasoning," *CACM*, Sept. 1985, Vol. 28, pp. 905-920.

[Find79] Findler, N.V., "A heuristic information retrieval system based on associative networks," in: Findler, N.V. (ed.), *Associative Networks*, pp. 305-326, 1979.

[Fox 47] Fox, R., *Great Men of Medicine,* Random House, New York, 1947.

[Fox 87] Fox, M. S., "Beyond the knowledge level," in [Kers87], pp. 455-463.

[Gent83a] Gentner, D., "Structure-mapping: a theoretical framework for analogy," *Cognitive Science,* vol. 7, pp. 155-170, 1983.

[Gent83b] Gentner, D. and Stevens (eds.), *Mental Models,* Erlbaum, 1983.

[Gera68] Gerardin, L., *Bionics,* (Translated from French), McGraw-Hill, New York, 1968.

[Gick80] Gick, M. L. and K. J. Holyoak, "Analogical problem solving," *Cognitive Psychology,* vol. 12, pp. 306-355, 1980.

[Gick83] Gick, M. L. and K. J. Holyoak, "Schema induction and analogical transfer," *Cognitive Psychology,* vol. 15, pp. 1-38, 1983.

[Gold86] Golden, M., R. W. Siemens and D. C. Ferguson, "What's wrong with rules?" *Proc. IEEE WESTEX,* pp. 162-165, 1986.

[Gran87] Grant, J., *Logical Introduction to Databases,* Harcourt Brace Jovanovich, 1987.

[Greg63] Gregor, A. S., *A Short History of Science: Man's Conquest of Nature from Ancient Times to the Atomic Age,* MacMillan, New York, 1963.

[Hill85] Hillis, W.D., *The Connection Macnine,* MIT Press, 1985.

[Holl79] Holland, "Text retrieval computers," *IEEE Computer,* pp. 40-50, March 1979.

[Holl86] Holland, J. H., K. J. Holyoak, R. E. Nisbett, P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery* MIT Press, 1986.

[Indu87] Indurkhya, B., "Approximate semantic transference: a computational theory of metaphors and analogies," *Cognitive Science*, vol. 11, pp. 445-180, 1987.

[Jant87] Jantke, ed. *Analogical and Inductive Inference*, Berlin: Spring-Verlag, 1987.

[John83] Johnson-Laird, P.N., *Mental Models*, Harvard Univ. Press, 1983.

[Jone83] Jones, B., "General systems theory and algorithm theory," *Int. J. General Systems*, Vol. 9, pp. 157-160, 1983.

[Jone85] Jones, B., "The cognitive contents of system substates," *Proc. IEEE Conf. Languages for Automation*, pp. 11-13, 1985.

[Kamo86] Kamouri, A. L., J. Kamouri and K. H. Smith, "Training by exploration: facilitating the transfer of procedural knowledge through analogical reasoning," *In. J. Man-Machine Studies*, vol. 24, pp. 171-192, 1986.

[Kers87] Kerschberg, L., (ed.), *Expert Database Systems: Proceedings from the First International Conference*, Benjamin/Cummings Publishing Company, 1987.

[Kidd87] Kidd, A. L., *Knowledge Acquisition for Expert Systems: A Practical Handbook*, Plenum Press, New York, 1987.

[Klah88] Klahr, D. and K. Dunbar, "Dual space search during scientific reasoning," *Cognitive Science*, vol. 12, pp. 1-48, 1988.

[Klin71] Kling, R.E., "A paradigm for reasoning by analogy," *Artificial Intelligence*, vol. 2, 1971, pp. 147-178.

[Klir85] Klir, G. J., *Architecture for System Problem Solving*, Plenum, 1985.

[Kolo83] Kolodner, J.L., "Indexing and retrieval strategies for natural language fact retrieval," *ACM Trans.Database Sys.*, vol.8, Sep. 1983, pp. 434-464.

[Kobs84] Alfred Kobsa, "Knowledge Representation: A Survey of its mechanisms, a sketch of its semantics", *Cybernetics and Systems: An International Journal,* vol. 41-89, 1984.

[Koch84] Kochen, M., "Toward a paradigm for information science: the influence of Derek de Solla Price," *JASIS,* vol. 35, No. 3, pp. 147-148, 1984.

[Korf85] Korfhage, R. R. and C. Hamphill, "Retrieval linguistics," *Proc. IEEE Languages for Automation,* pp. 173-177, 1984.

[Kraf83] Kraft, D. H. and D. A. Buell, "Fuzzy sets and generalized Boolean retrieval systems," *Int. J. Man-Machine Studies,* vol.19, 1983, pp. 45-56.

[Kraf85] Kraft, D. H., "Advances in information retrieval: what is that /#*&@c record?" *Advances in Computers,* vol.24, pp.277-318, 1985.

[Kwok86] Kwok, K.L., "An interpretation of index term weighting schemes based on document components," *Proc. ACM SIGIR,* pp.275-283, 1986.

[Lang87] Langley, P., H. A. Simon, G. L. Bradshaw and J. M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes,* MIT Press, Cambridge, MA: 1987.

[Lark88] Larkin, J. H., F. Reif, J. Carbonell and A. Gugliotta, "FERMI: A flexible expert reasoner with multi-domain inferencing," *Cognitive Science,* vol. 12, pp. 101-138, 1988.

[Lebo83] Lebowitz, M., "Intelligent information systems," *Proceedings, ACM SIGIR Conf.,* pp. 25-30, 1983.

[Lebo85] Lebowitz, M., "RESEARCHER: an experimental intelligent information system," *Proc. IJCAI*, pp. 858-862, 1985..

[Leve84] H.J.Levesque, "Foundations of a Functional Approach to Knowledge Representation", *Artificial Intelligence*, vol. 23, 155-212, 1984.

[Litw84] Litwin, W., "MALPHA: A relational multidatabase manipulation language," *Proc. 1st IEEE COMDEC*, pp. 234-242, 1984.

[Litw86] Litwin, W. and Ph. Vigier, "Dynamic attributes in the multidatabase system MRDSM," *Proc. 2nd IEEE COMDEC*, pp. 103-110, 1986.

[Maud87] Maudlin, M. L., "Better information retrieval through linguistic sophistication," pp. 189-192, in [Mich85].

[McGr81] McGregor, D.R. and J.R.Malone, "The fact database: a system based on inferential methods," *Research and Development in Information Retrieval*, (Oddy, R. N. et al. eds.), pp.203 -217, 1981.

[McKe85] McKeown, K. R., *Text Generation: Using discourse strategies and focus constraints to generate natural language text*, Cambridge University Press, Cambridge, UK: 1985.

[Mich83] Michalski, R.S. et al. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983.

[Mich85] Michalski, R.S. et al. (eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. II, Morgan Kaufmann, 1985.

[Mitc87] Mitchell, T. M., Carbonell and R. S. Michalski, *Machine Learning: A Guide to Current Research*, Boston: Kluwer Academic Publishers, 1986.

[Motr87] Motro, A., "Superviews: virtual integration of multiple databases," *IEEE Transactions on Software Engineering*, vol. SE-13, No.7, July 1987.

[Naka82] Nakamura, K. and S.Iwai, "A representation of analogical inference by fuzzy sets and its application to information retrieval system," in: Gupta, M.M. and E.Sanchez (eds.), *Fuzzy Information and Decision Processes*, pp. 373-386, 1982.

[Newe76] Newell A. and H. A. Simon, "Computer science as empirical inquiry: symbols and search," *Communications of the ACM*, March 1976, Vol. 19, No. 3, pp. 113-126.

[Nils71] Nilsson, N. J., *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New York.

[Norm81] Norman, D. A. (ed.), *Perspective on Cognitive Science.* ABLEX publishing co., 1981.

[Oppe56] Oppenheimer, R., "Analogy in science," *Science*, pp. 127-135, 1956.

[Pent83] Pentland, A. P. and M. A. Fischler, "A more rational view of logic, or, up aganist the wall, logic imperialists!" *AI Magazine*, pp. 15-18, Winter 1983.

[Petk83] Petkoff, B., "Artificial intelligence and computer simulation of scientific discoveries," in: Bibel, W. and B. Petkoff (eds.), *Artificial Intelligence: Methodology, Systems, Applications*, pp. 19-34.

[Pric63] Price, D. de Solla, *Little Science, Big Science*, Columbia University Press, New York and London, 1963.

[Pric78] Price, D. de Solla, "The evolution of invention," Part 6 in R. Bourne (ed.), *Smithsonian Book of Invention*, New York: W. W. Norton & Co., 1978, pp. 184-190.

[Quil69] Quillian, M. R., "The teachable language comprehender: a simulation program and theory of language," *Communications of the ACM*, vol. 12, No.8, pp. 459-479, 1969.

[Rend83] Rendell, L.A., "Toward a unified approach for conceptural knowledge acquisition," *AI Magazine*, pp. 19-27.

[Rich83] Rich, E., *Artificial Intelligence*, McGraw-Hill, 1983.

[Rick85] Rickheit, G., W. Schnotz and Hans Strohner, "The concept of inference in discourse comprehension," in G. Rickheit and H. Strohner (eds.), *Inferences in Text Processing*, pp. 3-50, 1985.

[Ries84] Riesbeck, C. K., "Knowledge reorganization and reasoning style," *Int. J. Man-Machine Studies*, pp. 45-62, Vol. 20, No. 1, 1984.

[Roth85] Rothfeder, J., *Minds over Matter*, Simon & Schuster, Inc., New York, NY: 1985.

[Rout85] Routh, R. L., "Cortical thought theory: a working model of the humanhuman gestalt mechanism"(Abstract of a 1985 dissertation), *Dissertation Abstract International*, p. 3518, Vol. 46, No. 10, 1986. 1985.

[Rume87] Rumelhart, D. E., J. L. McCeelland and PDP, *Parallel Distributed Processing*, vol. 1 & 2.

[Russ86] Russell, S.J., "A quanatitative analysis of analogy by similarity," *Proceedings, AAAI-86*, 1986, pp.284-288.

[Salt83] Salton, G. and M.J.McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[Scha82] Schank, R. *Dynamic Memory,* 1982.

[Scha83] Schank, R., J. L. Kolodner and G. DeJong, "Conceptual information retrieval," *Proceedings, SIGIR,* 1983, pp. 94-116.

[Ship49] Shippen, K. B., *The Brght Design,* The Vikin Press, New York, 1949.

[Silv83] Silverman, B. G., "Analogy in systems management: a theoretical inquiry," *IEEE Transactions on Systems, Man and Cybernetics,* SMC-13, no.6, Noc./Dec., pp. 1049-1075, 1983.

[Smit76] Smith, L. C., "Artificial intelligence in information retrieval systems," *Inf. Process. Manage.,* vol. 15, pp. 189-222, 1976.

[Stan86a] Stanfill, C. and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM,* Dec. 86, vol. 29, No. 12, pp. 1213-1228.

[Stan86b] Stanfill, C. and B. Kahle, "Parallel free-text search on the connection machine system," *CACM,* Dec. 1986, pp. 1229-1239.

[Ster77] Sternberg, R.J., *Intelligence, Information Processing, and Analogical Reasoning: The Componential Analysis of Human Abilities,* Lawrence Erlbaum Associates, 1977.

[Ster86] Sternberg, R.J., "Toward a unified theory of human reasoning," *Intelligence,* vol. 10, pp. 281-314, 1986.

[Stol84] Stolfo, S.J., "Learning control of production systems," *Cognition and Brain Theory,* vol.7, No.1, pp. 61-88, 1984.

[Ston87] Stone, H. S., "Parallel querying of large databases: a case study," *IEEE Computer,* pp. 11-21, Oct. 1987.

[Tani87] Tanimoto, S. L., *The Elements of Artificial Intelligence: An Introduction Using LISP*, Computer Science Press, 1987.

[Tesk87] Teske, F. N., "Enriched knowledge representations for information retrieval," *Proc. 10th ACM SIGIR Conf.*, pp. 43a-43g, 1987.

[Ullm82] Ullman, J. D., *Principles of Database Systems*, 2nd ed., Computer Science Press, 1982.

[Ullm88] Ullman, J. D., *Principles of Database and Knowledge-Base Systems*, vol. 1, Computer Science Press, 1988. (2nd ed.), Computer Science Press, 1982.

[Utgo86] Utgoff, P.E., *Machine Learning of Inductive Bias*, Kluwer Academic Publishers, 1986.

[vanR79] van Rijsbergan, C. J., *Information Retrieval*, 2nd. ed., Butterworths, 1986.

[Wass85] Wasserman, K., "Physical object representation and generalization: a survey of programs for semantic-based natural language processing," *AI Magazine*, Winter 1985.

[Wata85] Watanabe, S. *Pattern Recognition: Human and Mechanical*, John Weley and Sons, 1985.

[Wate78] Waterman, D.A. and F.Hayes-Roth (eds.), *"Pattern-directed Inference systems"*, Academic Press, 1978.

[Walt87] Waltz, D.L., "Applications of the connection machine," *Computer*, Jan. 1987, pp. 85-97.

[Wins80] Winston, P.H., "Learning and reasoning by analogy," *CACM*, 1980, vol.23, pp. 689-703.

[Wins84] Winston, P. H., *Artificial Intelligence,* (2nd ed.), Addison-Willey, 1984.

[Wong86] Wong, S. K. M. and W. Ziarko, "A machine learning approach to information retrieval," *Proc. ACM SIGIR,* pp. 228-233, 1986.

[Wood86] Woods, W. A., "Important issues in knowledge representation," *Proceedings of the IEEE,* pp. 1322-1334, vol. 74, No. 10, Oct. 1986.

[Zarr82] Zarri, G. P., "Relations between artificial intelligence and information science: expert systems and factual data banks," in: Ciampi,C. (ed.), *Artificial Intelligence and Legal Information Systems,* vol.1, 1982, pp. 389-405.

VITA

Zheng Xin Chen (Zhengxin Chen), the son of Zhi Chu Chen and Grace Fang Chen, was born on October 2, 1947 in Nanjing (Nanking), Jiangsu (Kiangsu) Province, China. He was raised in Shanghai, China, and graduated from high school in 1966. The chaos years then came, during 1966-1977, he worked in Shanghai 15th Pharmacy Plant. He entered East China Normal University in 1978 and received his B.S. Degree in mathematics in 1982. During 1982-1983 he was a mathematics instructor in Shanghai Pharmacy Workers' University. In 1983, he enrolled in Computer Science of Louisiana State University for graduate study, and received M.S. in System Science degree in 1985. He joins the faculty of Department of Mathematics and Computer Science, University of Nebraska at Omaha, in August 1988.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Zheng Xin Chen

Major Field: Computer Science

Title of Dissertation: COGMIR: A Computer Model for Knowledge Integration

Approved:

J. Burk Jones
Major Professor and Chairman

F. Glen Kemberg
Dean of the Graduate School

EXAMINING COMMITTEE:

Donald Hoff

Doris L Carver

Leslie P Jones

A. Elamary

Robert Walther

William A. Gale

Date of Examination:

July 25, 1988