# Spontaneous Retrieval in a Conceptual Information System

Lisa F. Rau
Artificial Intelligence Branch
Corporate Research and Development
GE Company
Schenectady, NY 12301 USA

**Abstract**

A traditional paradigm for retrieval from a conceptual knowledge base is to gather up indices or features used to discriminate among or locate items in memory, and then perform a retrieval operation to obtain matching items. These items may then be evaluated for their degree of match against the input. This type of approach to retrieval has some problems. It requires one to look explicitly for items in memory whenever the possibility exists that there might be something of interest there. Also, this approach does not easily tolerate discrepancies or omissions in the input features or indices. In a question-answering system, a user may make incorrect assumptions about the contents of the knowledge base. This makes a tolerant retrieval method even more necessary.

An alternative, two-stage model of conceptual information retrieval is proposed. The first stage is a spontaneous retrieval that operates by a simple marker-passing scheme. It is spontaneous because items are retrieved as a by-product of the input understanding process. The second stage is a graph matching process that filters or evaluates items retrieved by the first stage. This scheme has been implemented in the SCISOR information retrieval system. It is successful in overcoming problems of retrieval failure due to omitted indices, and also facilitates the construction of appropriate responses to a broader range of inputs.

## 1 Introduction

The System for Conceptual Information Summarization, Organization and Retrieval (SCISOR) is an information retrieval system designed to analyze, answer questions about, and summarize short newspaper stories in natural language. Operating in the domain of corporate takeovers and finance, SCISOR is unique in its approach to retrieval of the complex conceptual events stored in its knowledge base.

Most approaches to conceptual information retrieval can be broken down into the following four phases:

1. **Retrieval decision:** The system comes to a point in its processing when it desires some information from memory.

2. **Retrieval setup:** Indices or features are collected and put into a correct format for retrieval.

3. **Retrieval:** The retrieval process is performed.

4. **Post-processing / Matching:** The outcome of the retrieval process is examined and conditional actions may be taken.

The conceptual information retrieval performed in FRUMP (DeJong, 1979), CYRUS (Kolodner, 1984), COREL (DiBenigno, Cross & DeBessonet, 1986), (which uses the PEARL AI package (Deering, Faletti &Wilensky, 1981), and IPP (Lebowitz, 1983) can all be put into this framework, as could any system that performs deductive information retrieval in the style of Charniak, Riesbeck and McDermott (1980). An exact match restriction may be relaxed after the initial fetch returns a negative result. The model of finding items in memory here is an iterative one of generate and test. This model has certain problems when we consider how it could be used to perform certain desirable functions in an intelligent information retrieval system.

In contrast to this model, in SCISOR, items are retrieved from memory automatically as a result of the instantiation of new input instances. When a user's question is instantiated, potential answers appear in a short-term buffer. When a new story is instantiated, any previously existing context for that story appears in the buffer. This automatic retrieval is implemented with a constrained form of marker-passing. Items spontaneously retrieved in this manner are then run through a more computation intensive matching process.

Three problems with the model of retrieval initially described will be given, followed by a description of the SCISOR system and its solution to these problems.

# 2 Problem Description

## 2.1 Find things without looking

One capability the SCISOR system has is to retrieve automatically a user's previously posed but unanswered question when an answer to that question becomes known or refined. For example, consider the following scenario:

**User:** How much did the ACE company offer to take over ACME?

**System:** *Figures for the ACE-ACME takeover have not yet been disclosed.*

**Intervening time...**

**System:** *BEEP! Figures for the ACE-ACME takeover have just been released. ACE has offered $40 / share for all outstanding shares of ACME.*

In order to provide this capability to a system that performs retrieval as previously described, the system would have to keep a list of unanswered questions present. When new stories were input to the system, it would poll the unanswered questions, asking "does this answer you?". A better approach would be to set up demons on each unanswered question that look for certain input features that might relate to the content of the question. In either case, however, the system is *always looking* for answers to its questions. In SCISOR, if input features happen to relate to an unanswered question in memory, that question is spontaneously brought up for consideration. If the input does not relate to any unanswered questions, nothing happens.

## 2.2 Retrieval with incorrect or misleading inputs

SCISOR has another capability that would be awkward to implement in the model of retrieval previously described: to find events in memory even when a user's question contains only partial, or even incorrect information. For example, consider the following question, along with some independent potential states of the world that might be true at the time the user asked the question.

- Did ACE food company take over ACME hardware company?

  1. The ACME hardware company took over ACE food company.
  2. The ACE food company took over the BIG-ACME company, which *owns* the ACME hardware company.
  3. The ACE food company made an offer to the ACME hardware company, but has not yet succeeded in taking over the ACME company.
  4. ACE is a conglomerate and not simply a food company.

In each of these cases, the question asked cannot be well answered simply "yes" or "no". Consider what a deductive information retrieval mechanism such as that described in Charniak, *et. al.* (1980). might do to find the answers to the questions above. One possible method would be to retrieve all takeover events in which the company taking over another company was the ACE food company. The episodes found would then be checked to find ones in which ACE took over another company. The resulting events are examined to see whether the object of the takeover was the ACME hardware company. Such a procedure is incapable of detecting any of the scenarios described above without further augmentation.

There are many possible augmentations one could make. For example, one could try switching the arguments, looking up and down an "isa" hierarchy, or looking with all subsets of the set of features. One could also generate plausible indices through reconstruction of what was likely to be present in the situation, as is done in the CYRUS program (Kolodner, 1984). Although this procedure has a certain cognitive appeal, it is not guaranteed to find events present in the memory. In fact, none of the augmentations described is guaranteed to find relevant situations; some are not particularily principled, and all involve substantial additional computation.

In SCISOR, finding partially matching scenarios is a by-product of the retrieval process because the first pass of the two-stage retrieval process SCISOR uses simply finds events with features that are the same as, or similar to, features in the input question. The validity of the relationships between the features is not considered until the second pass of the two-stage process is performed. Thus, any of the scenarios above would be retrieved given the input question as stated. The evaluation mechanism then determines what is the same as the user's question and what the differences are. These differences may then be expressed to the user. Note that when nothing closely matches what the user asked, no events in the system's knowledge base will have enough activation to exceed the threshold, and the system will respond that it doesn't know.

Finally, the SCISOR system can find previous articles stored in memory when a new article is input that deals with the same situation. In the corporate takeover domain, events happen over time. For example, ACE may make an offer to ACME on Monday, and ACME may respond to the offer on Friday. The initial offer should be retrieved from memory when the response to the offer is input, so that this new piece of information can be properly integrated. The way that this could be done with the traditional method of retrieval is similar to the case of retrieving unanswered questions when the answers are input. Checking after each new story to see if it is a

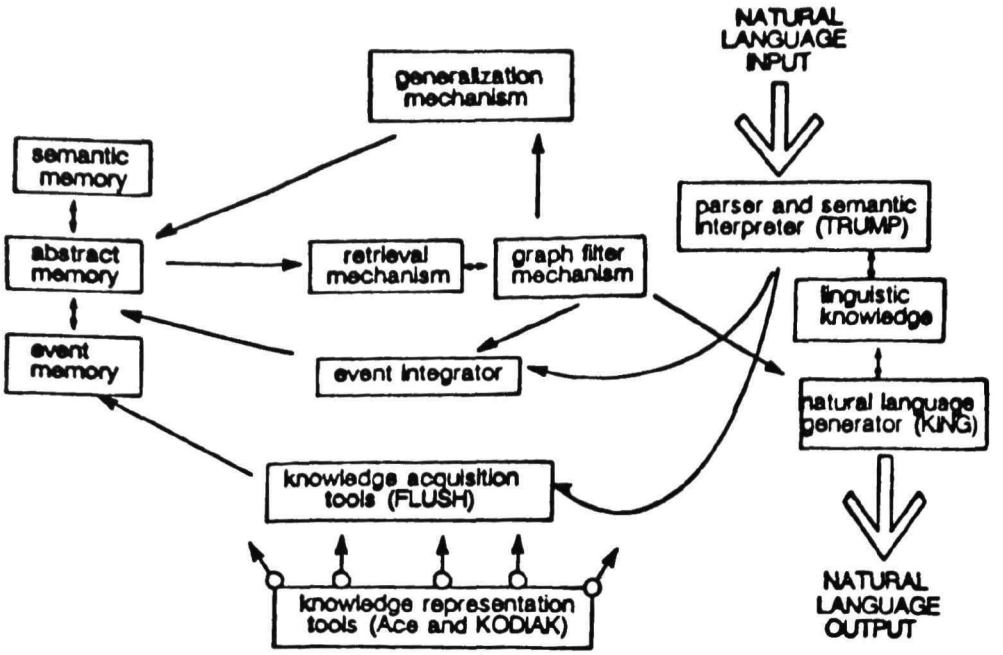


Figure 1: SCISOR System Architecture

continuation of a known story would result in substantial additional overhead. Also, dealing with the features present in a story continuation could have the same problem as dealing with missing or incorrect features in a user's input questions. For example, it would be difficult to find out that ACE was trying to take over ACME if subsequently ACME announced it was trying to take over ACE. Both of these concerns are elegantly addressed with SCISOR's method of retrieval.

This section has described some of the problems with traditional methods of conceptual information retrieval. The next section describes the SCISOR system and in more detail how its approach to retrieval addresses the problems.

# 3 System Description

## 3.1 Implementation

SCISOR takes input in natural language, integrates new information in memory, and answers natural language questions in natural language. The natural language input is processed with the TRUMP parser and semantic interpreter (Jacobs, 1986). New events are integrated as a continuation of an ongoing story (if present) by the event integrator, which also stores new events for retrieval. The FLUSH acquisition tools (Besemer, 1986) are aids to the acquisition of vocabulary and phrases in the system's phrasal lexicon. The events in SCISOR are represented using the KODIAK knowledge representation language (Wilensky, 1986), augmented with some scriptal knowledge (Schank & Abelson, 1977) of typical events in the domain. Linguistic knowledge is represented using the Ace linguistic knowledge representation framework (Jacobs & Rau, 1985). Responses to the user are to be generated with the KING (Jacobs, 1985) natural language generator. Figure 1 illustrates the architecture of SCISOR.

The following exchange illustrates the kind of input-output behavior SCISOR was designed to exhibit:

**User:** Did ACE food company take over the ACME hardware company?

**System:** *Yes they did. The ACE food company completed the $13.2 million friendly takeover last Friday.*

**User:** What was the sequence of events in the ACE-ACME takeover deal?

**System:** *Rumors that ACME was to be taken over started May 13, 1985. The stock rose $5 a share to $65. On May 16, the ACE company announced that it had made an offer to ACME valued at $66 a share. The offer was a cash offer to purchase all 2 million outstanding shares of ACME. After a month of deliberation involving some potential conflict of interest matters, the ACE-ACME deal was completed last Friday.*

**User:** What happened yesterday in the HAL Corporation bid for BIG?

**System:** *HAL Corporation increased its bid for the BIG company yesterday, up $4 to $70 a share. BIG has not commented on the new offer.*

## 3.2 Memory Organization

SCISOR manipulates conceptual structures represented in the KODIAK knowledge representation language. KODIAK can be viewed as a hybrid frame and semantic net-based language, similar in spirit to KRYPTON (Brachman, Fikes & Levesque, 1983) or KL-ONE (Brachman & Schmolze, 1985). In SCISOR, knowledge stored is either specific, abstract, or semantic. An example of a specific memory is the memory of the ACE food company acquiring the ACME hardware company. A traditional episodic memory (Tulving, 1972) is composed of specific memories. An example of an abstract memory is the generalization across specific experiences we might have heard about where companies have acquired other companies. Abstract (or generalized episodic) memories are on the border between specific memories and semantic memory. Semantic memory is the memory or knowledge of what "companies" are and what "acquiring" is. Semantic memory is used in understanding and making inferences about the input to the system. Figure 2 illustrates the structure of long-term memory with associated examples.

In addition to this tripartite division, another level of organization is superimposed on memory. Groups of related concepts in episodic or abstract memory are linked together through a common node, called a TAG. Each TAG has a numerical threshold value, currently equal to a fraction of the number of concepts in the episode, currently one-third. Long articles may consist of TAGs that have TAGs as components. Figure 3 illustrates the kind of structure the integrator superimposes on memory.

## 3.3 The Retrieval Mechanism

Retrieval in SCISOR is a two-stage process. The first stage is a coarse search that finds events in memory likely to be relevant. Relevance is determined by the number of features present in an event in memory related to features in the input. After the most likely candidates have been isolated, a matching process is performed.

The first stage of the retrieval process is performed by a process of priming or constrained spreading activation. As concepts are instantiated in the system, instances of concepts that are related via category membership links are marked (*i.e.*, primed or activated). When a certain subset of the concepts in an event or episode is marked, the entire episode is put into the system's short-term memory buffer. This is the spontaneous retrieval phase. The events that have been spontaneously retrieved can then be run through the match filter to check the nature of the match between the input and the events retrieved. Periodically, all marks are deleted from the system.

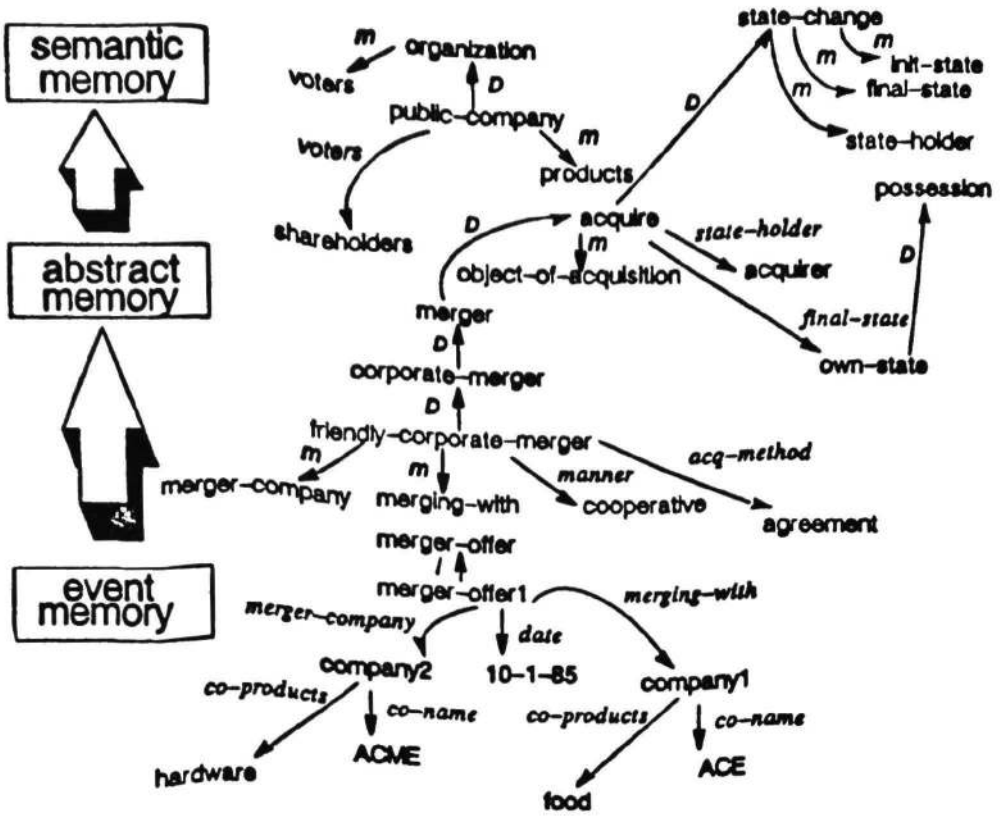


Figure 2: Structure of long-term memory

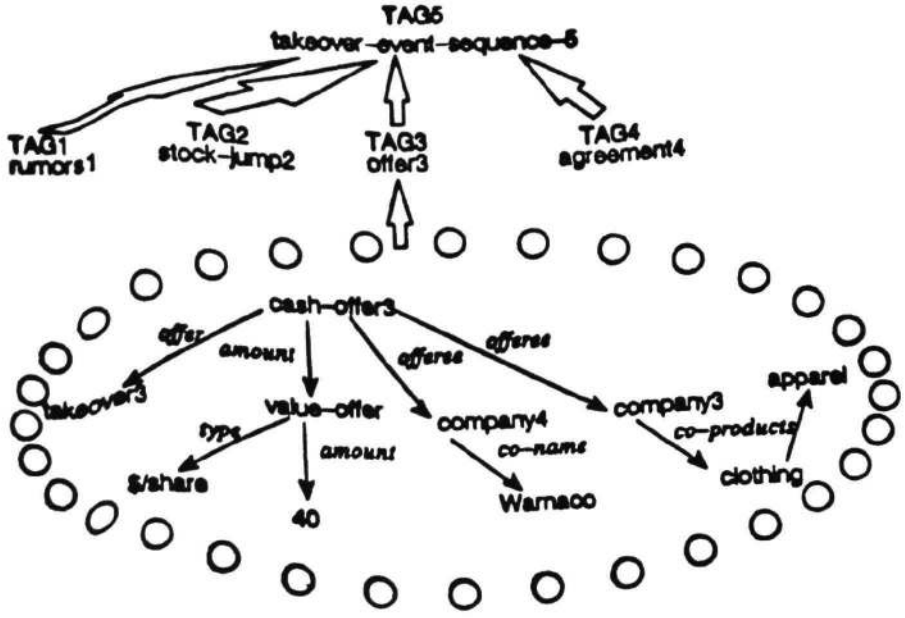


Figure 3: Superimposed Structure

Marker-passing "waves" are propagated continuously as new concepts are instantiated. This mechanism retrieves answers to input questions, old unanswered questions to new input answers, and stories given related input stories in exactly the same manner.

In more detail, the retrieval mechanism operates in the following manner:

1. At the time of concept instantiation, related concepts are marked according to the "priming rules" (see next section). Every concept that is marked causes its containing episode or episodes (its TAGs) to have increased activation. The current and threshold levels of activation are simple properties of the TAG.

2. TAGs that have had their values increased check themselves to see whether their current activation levels exceed their threshold.

3. If the TAG threshold has been exceeded, an *intersection* has been detected, and the entire episode is put into the short-term memory buffer.

4. In the current implementation, all concepts in the memory are unmarked after every one or two sentences and after every question are input.

Components of experiences in the input cause other concepts to be marked, but if those other concepts are not components of relevant experiences, the marking does not contribute to retrieval.

The rules that guide how levels of activation or marks are passed through the conceptual hierarchy are given below. These rules were formulated to decrease the possibility of retrieving memories that have limited predictive capacity or relevance to the current situation.

### 3.3.1 Priming Rules

The effect of the following rules is that all instances of concepts that are components of episodes and are children of the incoming parent of the parent of the incoming instance in the conceptual hierarchy, are marked. Also, all instances of concepts that are components of memories, and are *direct* instances of concepts that are parents of the incoming instance, are marked. This rule prevents everything in the hierarchy from being marked, and limits what is marked to a level of conceptual abstraction supported by the presence of a direct instance at that level.

For example if a user asked "what happened to the ACE company?", the event being asked about is at a fairly general level of conceptual abstraction. Any event more specific than this general "event" instance would be a valid answer. If no event were known, the unanswered question would be stored. In the future, *any* input events involving the ACE company (i.e., offers, rumors) would cause the unanswered question to be activated.

Figure 4 shows a simple example of where components of memories would cross-index each other and when components of memories would not. Double arrows between instances (concepts with numbers following them) signify cross-indexing. Single-headed arrows signify where the instantiation of one concept will activate the concept pointed to, but not vice versa. Note that all concepts as they are instantiated cause any related instances to be marked, which allows any feature in the input to be a potentially useful index key into memory. Also, note that each instance is part of a specific memory that is not shown in Figure 4. For example, "material-company1" may be part of the memory of the unspecified materials company that rumors were circulating about yesterday.

**Rules:**

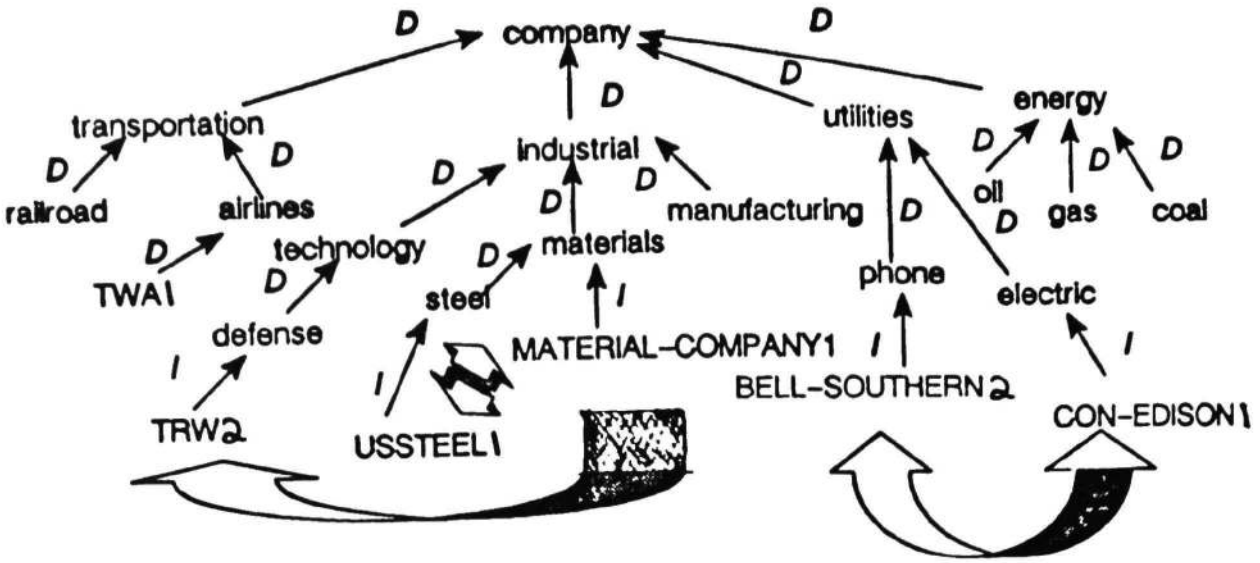


Figure 4: Example of Implicit Indexing

1. Mark only concepts that are components of specific or abstract memories. These concepts are marked with TAGs.
2. Determine the categories to which the incoming instance belongs (Categories-of A).
3. Determine the concepts in the reflexive, transitive closure along category membership links of Categories-of (Categories-of A).
4. Mark the direct instances of concepts in this reflexive, transitive closure. Each marked concept increases the current activation value of each of its TAGs.
5. Check to determine whether the current activation level of the episodes that just had their components marked exceeds the threshold level.

Additional refinements to this algorithm have been made to increase the system's efficiency. One such refinement has been to check the number of episodes containing a certain concept before passing markers to all those episodes. This check can be made easily by maintaining a count of the number of category members in each conceptual category. For example, if the system had read about events involving thousands of companies, this number would be stored at the parent COMPANY node. When this number is very large, the system suppresses the marker-passing operation. Events that are retrieved through the activation of more unique concepts then incorporate the high-frequency concept information. This simple refinement suppresses waves of activation unlikely to cause significant differentiation among events in memory, while still taking all features of the input into consideration.

After a set of events has been spontaneously retrieved, a match filtering operation is performed to ensure that the correct *relationship* exists between concepts within the events.

### 3.3.2 The Match Filter

The match filter iteratively checks to see that the relationships between marked concepts are correct. For example, consider the example illustrated in Figs. 5 and 6.

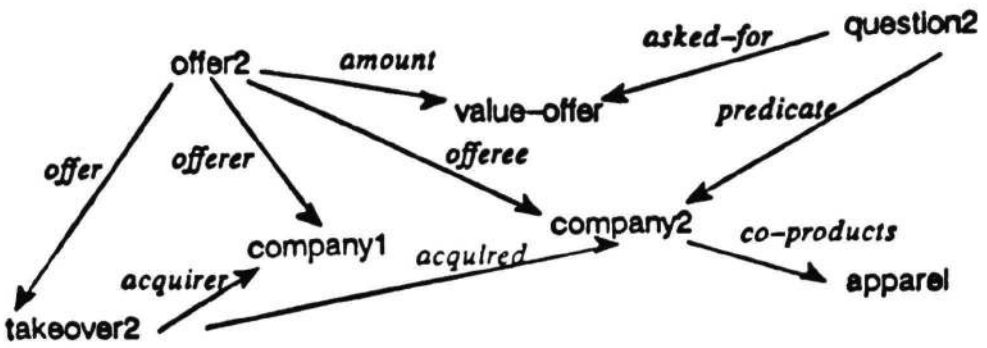


Figure 5: How much are takeover offers for apparel companies?

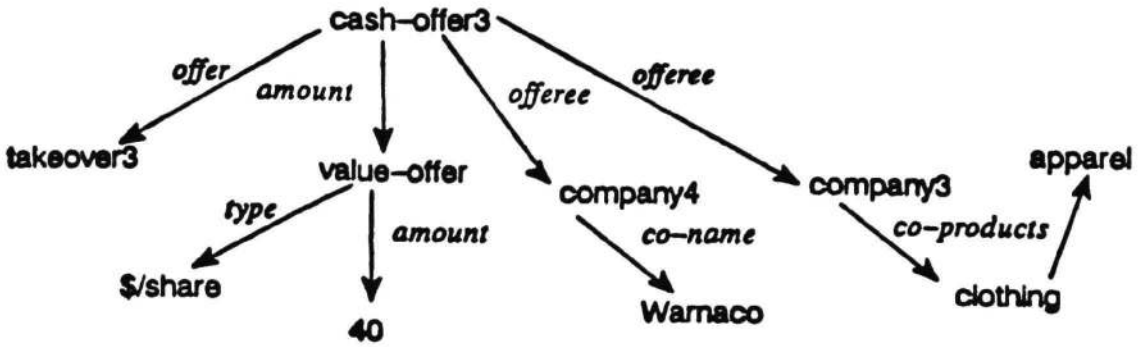


Figure 6: Part of a Story Episode

The story episode contains information about a kind of offer, a CASH-OFFER, made by Warnaco for a clothing company. The question episode requests the value of any offers made by a company in a takeover attempt on an apparel company.

Due to the marker-passing, the instantiation of OFFER in the question episode causes CASH-OFFER to be marked. The match filter begins at a node that is marked only by one node from the input. OFFER2 is such a node, but note that COMPANY2 and COMPANY3 are both marked by COMPANY4 and COMPANY5.

The filter proceeds to check that the OFFER of CASH-OFFER3 (TAKEOVER3) is marked by the OFFER of OFFER2 (TAKEOVER2). This process is repeated for every node in the input. In the case of answering a question, those nodes that do not correspond are added to a list of presuppositions to be expressed to the user. Also, any node that matched but was more general than the input may be expressed to the user. For example, if the user was interested in what pet-food companies were being taken over, and the system only knew about food companies, this generalization is pointed out.

# 4 Related Research

## 4.1 Question Answering

In SCISOR, the processes that find the approximate location of an answer to a user's question and the processes that determine what the answer should be are separate. A great deal of work has been done on the second problem, most notably by Lehnert (1978). Determining an appropriate answer to a user's question, given that the context in which the user's question was posed is already known, is a separate process from the initial retrieval of a context in which to search for an answer. This initial retrieval of a context is the spontaneous retrieval this paper describes.

## 4.2 Conceptual Information Retrieval

Kolodner (1984) has a well-developed theory of conceptual information retrieval. However, the system is not guaranteed to answer correctly. As a cognitive model, its memory failures are understandable and interesting, but when accuracy and reliability of information are important, such a model is not viable. Her theory of retrieval developed can be viewed as one attempt to overcome problems and limitations with existing methods of conceptual information retrieval, such as those described in Charniak, *et. al.*

## 4.3 Distributed Representation

As Hinton points out (Hinton, 1984) a localist knowledge representation scheme combined with a spreading activation or priming mechanism is hard to distinguish from a distributed representation in terms of functionality. Two main properties of distributed representations are that the knowledge base is contents-addressable and that generalization is automatic. Because the SCISOR knowledge base is represented using the KODIAK localist knowledge representation language and performs retrieval of items in the knowledge base by a constrained form of marker-passing (Charniak, 1983), SCISOR representations form an effective distributed representation, and therefore the structures in the knowledge base are contents-addressable. Thus, the content of input questions or stories "address" events in memory with similar features or content. The automatic generalization that SCISOR also exhibits will be discussed in another paper.

## 4.4 Cognitive Effects

In systems such as SCISOR that use the same parsing mechanism for both question parsing and story parsing, interesting effects occur. For example, in BORIS (Dyer, 1983), the system would occasionally take as true information present in a user's question not previously known to the system. Although this has been shown to be a cognitively valid effect (Loftus, 1975), and does increase the system's opportunities to learn, it is not a desirable effect for an information retrieval system. In the SCISOR system, a strict difference is maintained between information read in articles, and information present in user's questions in that all information present in questions and not previously known to the system is flagged as potentially unreliable. Thus the system believes nothing it is told, but everything it reads in the paper.

The model of retrieval discussed has a certain cognitive appeal. It exhibits the same kind of spontaneous recall as people do, as is discussed in Schank's work on reminding (Schank, 1982). Also, the answer to a user's question may be retrieved before the user has finished asking the question, an interesting effect also first achieved in Dyer's BORIS system (1983).

# 5 Summary

SCISOR performs retrieval in a two-stage process. The first step, a spontaneous and coarse search, operates as follows:

1. **New inputs are instantiated**
2. **Marker passing occurs**
3. **Items are spontaneously retrieved**
4. **Items are evaluated**

The second stage is a graph matching process that performs a syntactic matching function on the likely candidates retrieved by the first process. This spontaneous method of retrieval elegantly solves some retrieval problems found in other systems:

1. SCISOR addresses a user's previously unanswered questions if an answer becomes known without always looking for an answer.
2. SCISOR can locate relevant information in response to a user's questions, even when that question contains misleading or partial information.
3. SCISOR retrieves previous events when updates of those events are read. This is done in the same manner as retrieving an unanswered question, and with the same tolerance for partial or contradictory input as in the question-answering case.

# 6 System Status

SCISOR is implemented in Common Lisp; it is used on VAX computers and Symbolics and SUN workstations. The TRUMP parser and semantic interpreter has not yet been tested with a large grammar or vocabulary but, in these early stages, it has been relatively easy to customize. On the SUN-3 it processes input at the rate of a few seconds per sentence, including the selection of candidate parse and semantic

interpretation. The KING natural language generator was implemented in Franz Lisp, and at this writing has not yet been converted to Common Lisp to run with TRUMP.

The system has now been tested with a dozen or so stories stored in the knowledge base. Hundreds of semantic concepts and domain vocabulary are also present. About a dozen questions are answered by the system.

# 7 Problems

One problem with SCISOR is in the kind of questions it can answer. Currently the SCISOR system is capable of answering only questions about information explicitly stored in the knowledge base. Any information that potentially could be *reconstructed* or *inferred* from information stored in the knowledge base is not available. The line between what is explicit in a story and what can be deduced from that story is not sharp, because some amount of "figuring" must go on to obtain any reasonable understanding of the story. To obtain this understanding, SCISOR computes something similar to a maximally complete inference set (Cullingford, 1986) as the set of information present explicitly in articles and inferred from the context and other world knowledge. Anything in that understanding can be directly retrieved.

For example, SCISOR is able to answer the question "What company was sold for $3 billion?" without pre-indexing a story containing that information by AMOUNT-OF-SALE. However the system cannot answer the question "Which companies have been taken over more times than they have taken over other companies?" for example, because an answer would require counting all the times a company has been taken over and has taken over other companies, comparing these two numbers, and repeating the process for every other company in the knowledge base.

Another problem that ultimately must be addressed is the speed and memory requirements necessary in a viable information system. A simple calculation based on the number of relevant articles per year (10,000) and the average size of the memory requirements per conceptual representation of the articles yields a conservative memory requirement estimate of 25 Mbytes of storage. Such a large knowledge base may effect the speed and accuracy of the retrieval mechanism.

# 8 Conclusions

SCISOR is an experiment in the usefulness of a spontaneous, marker-passing approach to conceptual information retrieval. In its current implementation, it has demonstrated some promising results. The marker-passing scheme, combined with the knowledge representation used, produces an effective contents-addressable, distributed representation. Retrieval occurs spontaneously when features in the input are related to features of events in memory. SCISOR can find answers to input questions even in light of missing or misleading input information.

The system has not yet been tested on a large number of documents. However so far, the tests that have been performed are quite promising. Before any definitive claims can be made about the ultimate usefulness of this type of system, it must be tested with a large sample of documents in real IR tasks. The next stage of the project will include such tests.

# References


[1] D. Besemer. *FLUSH: Beyond the Phrasal Lexicon.* Technical Report 86CRD181, General Electric Corporate Research and Development, 1986.

[2] R. Brachman, R. Fikes, and H. Levesque. Krypton: integrating terminology and assertion. In *Proceedings of the National Conference on Artificial Intelligence*, Washington, D. C., 1983.

[3] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 1985.

[4] E. Charniak. Passing markers: a theory of contextual influence in language comprehension. *Cognitive Science*, 7(3), 1983.

[5] E. Charniak, C. Riesbeck, and D. McDermott. *Artificial Intelligence programming.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[6] R. E. Cullingford. *Natural Language Processing: A Knowledge-Engineering Approach.* Rowman and Littlefield, Totowa, NJ, 1986.

[7] M. Deering, J. Faletti, and R. Wilensky. PEARL: an efficient language for artificial intelligence programming. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, 1981.

[8] G. DeJong. *Skimming Stories in Real Time: An Experiment in Integrated Understanding.* Research Report 158, Department of Computer Science, Yale University, 1979.

[9] DiBenigno, M.Kathryn, Cross, George R. and DeBessonet, Cary G. *COREL - A Conceptual Retrieval System.* Technical Report CS-86-147, Computer Science Department, Washington State University, 1986.

[10] M.G. Dyer. *In-Depth Understanding.* MIT Press, Cambridge, MA, 1983.

[11] G. Hinton. *Distributed Representations.* Computer Science Technical Report CMU-CS-84-157, Carnegie-Mellon University, 1984.

[12] P. Jacobs. *A knowledge-based approach to language production.* PhD thesis, University of California, Berkeley, 1985. Computer Science Division Report UCB/CSD86/254.

[13] P. Jacobs. Language analysis in not-so-limited domains. In *Proceedings of the Fall Joint Computer Conference*, Dallas, Texas, 1986.

[14] P. Jacobs and L. Rau. Ace: associating language with meaning. In T. O'Shea, editor, *Advances in Artificial Intelligence*, North Holland, Amsterdam, 1985.

[15] J. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[16] M. Lebowitz. Generalization from natural language text. *Cognitive Science*, 7(1), 1983.

[17] W. G. Lehnert. *The Process of Question Answering: Computer Simulation of Cognition.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[18] E. F. Loftus. Leading questions and the eyewitness report. *Cognitive Psychology*, 7, 1975.

[19] R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates, Halsted, NJ, 1977.

[20] R.C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, 1982.

[21] E. Tulving. Episodic and semantic memory. In E. Tulving and W. Donaldson, editors, *Organization and Memory*, Academic Press, New York, 1972.

[22] R. Wilensky. Knowledge Representation - A Critique and a Proposal. In J. Kolodner and C. Riesbeck, editors, *Experience, Memory, and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.