University of Colorado at Boulder

Department of Computer Science

ECOT 7-7 Engineering Center
Campus Box 430
Boulder, Colorado 80309-0430
(303) 492-7514

IN THE PROCEEDINGS OF THE ACM CONFERENCE ON
HUMAN FACTORS IN COMPUTING SYSTEMS (CHI '89), AUSTIN, TX, APRIL 30 - MAY 4, 1989

# HELGON:
# Extending the Retrieval by Reformulation Paradigm

*Gerhard Fischer and Helga Nieper-Lemke*

Department of Computer Science and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430

## Abstract

People who attempt to use a complex information store on a computer encounter a number of problems: They do not know what information exists or how to find information, they get no support in articulating a question, and they are unable to phrase their question in terms that the system understands. HELGON, an intelligent environment that supports limited cooperative problem solving, helps people deal with complex information stores. HELGON supports retrieval and editing by reformulation with multiple specification techniques, and it acquaints the user with the system model of the information store. Within the current HELGON system, a number of different information stores have been implemented. Empirical evaluations have shown that HELGON supports effective communication. In addition, the evaluations have shown interesting extensions for future work.

**KEYWORDS:** Complex information stores, information retrieval, retrieval by reformulation, editing by reformulation, cooperative problem solving systems, visualization.

# HELGON:
# Extending the Retrieval by Reformulation Paradigm

*Gerhard Fischer and Helga Nieper-Lemke*

Department of Computer Science and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430

**ABSTRACT**
People who attempt to use a complex information store on a computer encounter a number of problems: They do not know what information exists or how to find information, they get no support in articulating a question, and they are unable to phrase their question in terms that the system understands. HELGON, an intelligent environment that supports limited cooperative problem solving, helps people deal with complex information stores. HELGON supports retrieval and editing by reformulation with multiple specification techniques, and it acquaints the user with the system model of the information store. Within the current HELGON system, a number of different information stores have been implemented. Empirical evaluations have shown that HELGON supports effective communication. In addition, the evaluations have shown interesting extensions for future work.

**KEYWORDS:** Complex information stores, information retrieval, retrieval by reformulation, editing by reformulation, cooperative problem solving systems, visualization.

## INTRODUCTION

Retrieval by reformulation as exemplified by systems such as RABBIT [16, 15] and ARGON [12] has proven to be an innovative and effective way to deal with complex information stores. Informal evaluation of the ARGON system and a formal evaluation of an earlier version of the HELGON system uncovered some limitations that we have tried to overcome in the recent version of the HELGON system. In HELGON, the retrieval by reformulation paradigm is extended in two ways: Users have additional possibilities to specify a query to the knowledge base; and they can use the same framework for the creation of new knowledge base items.

In this paper, we describe complex information stores and illustrate the importance of cooperative problem solving in dealing with them. We present a conceptual framework for our system-building efforts and describe the HELGON system, which is a first step towards supporting cooperative problem solving between a user and the system in dealing with complex information stores. We conclude by discussing the evaluation of our current system and outlining plans for the future.

## COMPLEX INFORMATION STORES

Complex information stores (e.g., high-functionality computer systems such as LISP machines, knowledge bases about literature, and personal information environments) are different from databases. Their complexity arises not from the number of items in the system but from their heterogeneity, and the abstraction space to characterize them is not well developed. The interaction paradigms for dealing with complex information stores (as well as databases) have often been based on the unfounded assumption that people using these systems approach them with a precisely described task. But in most problem-solving and information retrieval tasks, the articulation of a precise task is the most difficult problem. HELGON is a system that tries to overcome this difficulty.

### Problems with Complex Information Stores

In ill-structured problem domains, there are no precise goals or specifications for problem solving and information retrieval. Consequently, for example, users of high-functionality computer systems, which contain tens of thousands of objects and tools [2], suffer from a lack of knowledge about the interdependencies between problem articulation and specification, and of knowledge about the tools that exist for solving these problems. (In fact, empirical studies have shown that there are no experts with complete mastery of high-functionality computer systems [1]). Ignorant of these mappings, users concentrate too quickly on implementation and often overlook alternative solutions. Users are unable to generate a mental model [3] of these systems, and without adequate support tools, these systems are unusable.

### The Importance of Cooperative Problem Solving

When humans (e.g., a novice and an expert) communicate, much more happens than just the request for factual information. A question can be phrased in a variety of ways. Novices cannot ask questions about knowledge that they do not know exists, and they may not be able to articulate their questions without the help of the expert. They ask many questions initially at a very general level, and a good deal of dialogue must occur before the communicators attain sufficient level of specificity.

These dynamics of cooperative problem solving indicate why *natural language interfaces* to databases do not solve the critical problem of access to information. Studies (e.g., [14]) have provided evidence that a natural language front-end is a fallacy. Current natural language interfaces support the translation of a fully articulated query from natural language into a formal query language, but they do not assist users who are unable to describe precisely what they want at the beginning of an information-seeking process.

## THE CONCEPTUAL FRAMEWORK BEHIND HELGON

### Retrieval by Reformulation

HELGON is based on the paradigm of retrieval by reformulation [16, 15], which was derived from a theory of human remembering [10]. This theory postulates that people naturally think about categories of things not in terms of formal attributes but in terms of examples. HELGON supports the incremental description of a desired object with multiple specification techniques. Systems that support retrieval by reformulation are cooperative in the sense that after users give an initial description, the system responds with an indication of its understanding by displaying example items from the knowledge base that match this description. Users then refine their description based on this feedback until a suitable item is found or until the absence of such an item is established.

### Situation Model and System Model

Many current information stores (e.g., help systems) are oriented toward the system rather than toward the user. That is, information is structured around a description of the system, not around an analysis of the problems users address when using the system. For example, a shortcoming of many existing information stores is that access is by *implementation unit* (e.g., LISP function, UNIX command) rather than by *application goal* on the task level.

In our evolving framework for user-centered system design [4], we define the *situation model* as a mental representation of the situation as the user sees it, including the problems motivating a task, general ideas for finding a solution, and a characterization of the desired goal state. A crucial stage of problem solving is to map this imprecise situation model onto a formal *system model*. The system model consists of a set of operations that, when invoked, will result in the desired solution. These operations must all be within the repertory of the system; that is, for each operation there must exist one or more commands, depending upon context, to execute it. At the level of the situation model, goals refer to actions and states in the user's problem space. Goals may be precise or imprecise, but the important point is that they are not necessarily structured or named according to the system [6]. The successful use of a complex information store requires that the goals expressed in the situation model be translated in terms of the system model (see Figure 1). HELGON's approach to solving this problem is to make the system model more obvious to the user by providing support tools for easy access to the "world" of the system.

**Query as represented in a specific user's situation model:**

"Find the reference for the final ONR project report from CU."

**A corresponding query in system-understandable terms:**

```
TECHREPORT
  INSTITUTION: Department of Computer Science,
               University of Colorado
  YEAR:        1988
```

**Figure 1:** Situation Model versus System Model

### Switching the Role of Speaker and Listener

Early systems, such as RABBIT and ARGON, and earlier versions of HELGON accomplished retrieval by reformulation by assigning the role of the speaker exclusively to the system. Users could not describe parts of the query directly; they could only criticize examples. This design feature was intentional, because these systems were based on the assumption that users in most cases do not know what they want and therefore need to be guided in the formulation of their queries. This type of interaction avoided the well-known problems that occur when users do not know names for objects and attributes in a heterogeneous and complex information store and therefore do not share an understanding of the information store with the system [6].

The evaluation of an earlier version of HELGON [5] showed that this strict assignment of roles has major drawbacks. If users know what they want, they should be able to say so directly. Being restricted to the critiquing of examples can lead to cumbersome specifications. The trade-offs for different role assignments are shown in Figure 2.

**User in Listener Role:**

- *Specification of Information*: Clicking at information displayed on the screen.
- *Advantage*: Only terms that the system knows can be used.
- *Disadvantage*: The information has to be on the screen.

**User in Speaker Role:**

- *Specification of Information*: Keyboard input.
- *Advantage*: Users can type in values they know right from the beginning.
- *Disadvantage*: Users may use terms the system does not know.

**Figure 2:** Speaker versus Listener Role

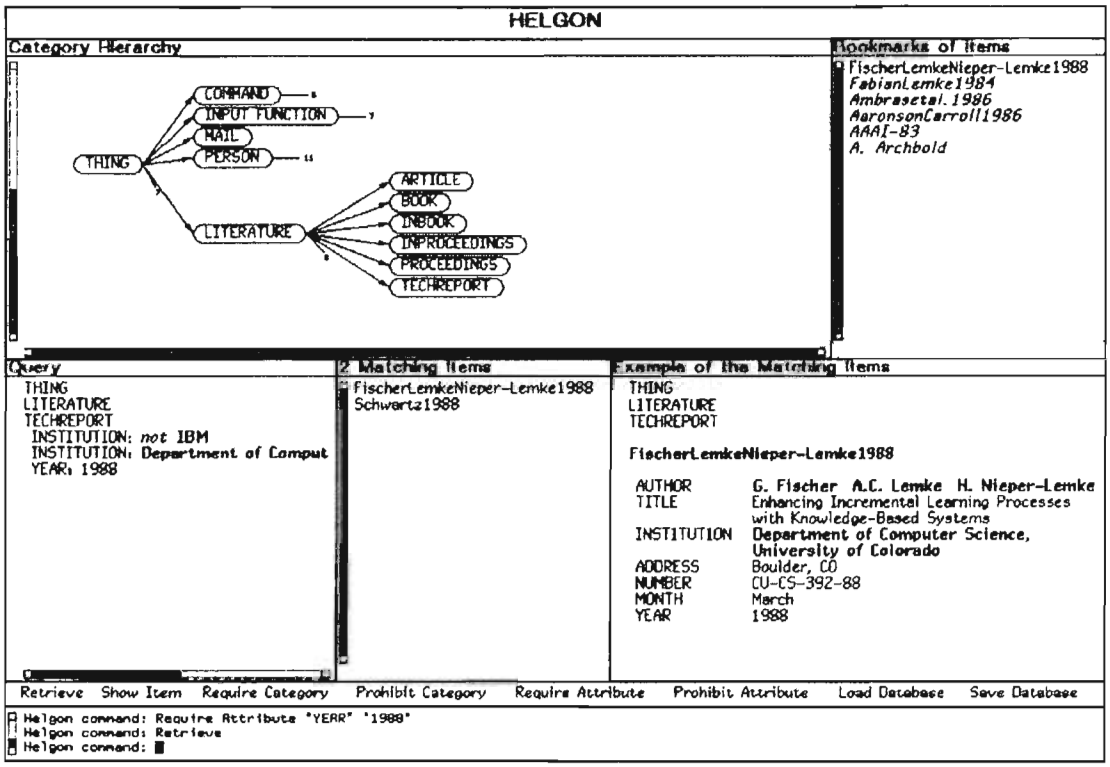


**Figure 3:** Literature References in HELGON

### Reuse and Redesign

One of the greatest promises of functionality-rich systems is that they offer new ways to construct systems: Instead of starting from scratch, designers can develop systems through *reuse* and *redesign* [2]. But without adequate system support, the probability of a user's finding the right tool for a specific need is small. Hypertext-based information retrieval tools, such as the SYMBOLICS DOCUMENT EXAMINER [13], solve some of the problems but do not support the construction of queries. Initial experiments in addressing these problems with HELGON have shown promising results. HELGON will not solve the reuse and redesign problem, but a tool like it will be a necessary prerequisite.

## DESCRIPTION OF HELGON

### System Architecture

HELGON is implemented in COMMON LISP and FLAVORS on SYMBOLICS LISP machines and takes advantage of high-level substrates for interface design, knowledge representation, and visualization:

- A "Program Framework" organizes the interaction and consists of a command loop, a set of commands, a window layout declaration, and a set of window display functions.
- "Presentation Types" define the presentation of an internal object on the screen. Presentations are mouse-sensitive, and operations, which appear in context-sensitive pop-up menus, are associated with them.
- KANDOR [11] serves as the knowledge representation language for the information store. The basic entities of a KANDOR knowledge base are categories, items, and attributes. The knowledge base is structured as a hierarchy of categories; items are an instance of one or more of these categories; and information is associated with items by means of attributes.
- TRISTAN [9] is a generic tool for displaying and editing directed graphs. It is used in HELGON for visual presentation of the structure of the knowledge base. It allows the partial display of any subset of nodes and the corresponding links of a directed graph.

### Retrieval by Reformulation in HELGON

The query is initialized with the root node of the category hierarchy. The list of items matching the query is shown in the **Matching Items** pane, and one of the matching items is shown in the **Example of the Matching Items** pane (see Figure 3). The query consists of categories and attribute restrictions associated with the categories. Categories as well as attribute values can be either "required" or "prohibited." The user does this by selecting them from the screen or a menu of alternative values or by typing them in on the keyboard. When the user makes additions to the query through input on the keyboard, only useful values, that is, values that exist in the knowledge base, are accepted. This prevents the user from imposing a restriction that would by itself lead to no matching items (e.g., because of a typographical error). The system gives help by completing partial input automatically if it is unique or by listing all possibilities that contain the current input as a substring.

Users can create the query top-down by selecting from the category hierarchy display the category that is expected to contain the desired information. But users may not know in what category the information is stored. Therefore, they can also work bottom-up by criticizing the example. A problem of this approach is that, in a large information space, the example given might be too far away from the desired information. Multiple specification techniques, e.g., first narrowing down the information space by selecting categories, then continuing by criticizing examples, are therefore important in dealing with complex information stores.

### Visualization of the Information Store

It is well-known that users become disoriented in large information stores [7]. HELGON allows therefore to display the structure of the underlying information store, i.e., a hierarchy of categories, graphically. Once users have found a category that seems likely to hold the information they are looking for, they can add it to the query with a mouse click. They can also use the graphical display to edit the underlying structure of the information store (e.g., new subcategories can be created).

### Browsing

In addition to assisting the user in defining a query, HELGON supports browsing in the information store. The graphical category hierarchy display can be used to browse categories. Links within the information units can be followed, that is, items that appear as attribute values of other items (displayed in bold face) can be inspected. And items that users looked up previously are added to the **Bookmarks of Items** and allow users to return easily to previous states of their information search.

### Editing by Reformulation

HELGON is not just a tool for *viewing* information -- one of the shortcomings we identified with the RABBIT and ARGON systems. It allows users to *edit* information and integrates the creation of new knowledge base items with the retrieval by reformulation paradigm. Users can thereby take advantage of the context created by an information search to editing without having to switch to the KANDOR level. They first use retrieval by reformulation to find an item that is similar to the new one, copy it, and use it as a template. In this way, they know which categories and attributes are reasonable to use; and because they see examples, they better understand what a category or attribute means. Values can often be reused, or they can be selected from a list of alternative values. They can also be typed in on the keyboard, a feature providing the same support (automatic completion, etc.) as it does in the formulation of a query. The query specification itself, which contains concepts used in the description of the information store, can also be transformed into a new item. This feature is in the spirit of the "specification by reformulation" paradigm of the BACKBOARD system [17].

## A SAMPLE DIALOGUE WITH HELGON

HELGON has been applied to several different domains including literature references, persons, electronic mail messages (an application that greatly extends access possibilities over those of the UNIX folder system), and a small subset of commands, flavors, and input functions available on SYMBOLICS LISP machines.

In the following sample dialogue with HELGON, the literature/person information store is used. The goal of the user, articulated in her situation model (cf. Figure 1) is: "Find the reference for the final ONR project report from CU." The initial state of the dialogue is shown in Figure 4.

One subcategory of **THING** (the root node of the category hierarchy) is **LITERATURE**. The user requires **LITERATURE** (see Figure 5) and retrieves. The matching items list is updated and now contains only literature entries. The user notes that the example belongs to a subcategory of **LITERATURE** called **PROCEEDINGS**, and she looks at the alternatives for **PROCEEDINGS** in the graphical category hierarchy display and decides that **TECHREPORT** is the right category to require. She does that and retrieves. The list of matching items now contains only technical reports. The **INSTITUTION** of the displayed example is **IBM**. The user knows that the report is not from **IBM** and therefore prohibits **IBM** (see Figure 6) and retrieves.

She realizes that she did not gain much through this last restriction and therefore decides to switch to the "speaker role" and use the keyboard to create a restriction for the institution. She wants to restrict it to **CU**, but the system immediately tells her that **CU** is not an institution it knows. She then types **Colorado**, gets a list of all existing institutions that contain **Colorado** as a substring, and selects **Department of Computer Science, University of Colorado** from this list.

She also knows that the technical report she is looking for is very new and therefore tries **YEAR: 1988** as an additional restriction. There are only two items matching the current query (this is the state shown in Figure 3), and by looking at the example the user is able to decide that this is the reference she is looking for because she recognizes its title.

After the user retrieves this technical report, she wants to know more about its authors. She clicks at one of the names, and HELGON displays the corresponding item. The user goes back to the technical report retrieved before by selecting it from the bookmarks. She creates an additional attribute called **KEYWORDS** for it and inserts **Final Project Report**. She can use this information next time to retrieve this piece of literature in the same way that she used the other attribute values before. A new University of Colorado technical report can now be entered by copying and editing the retrieved project report.

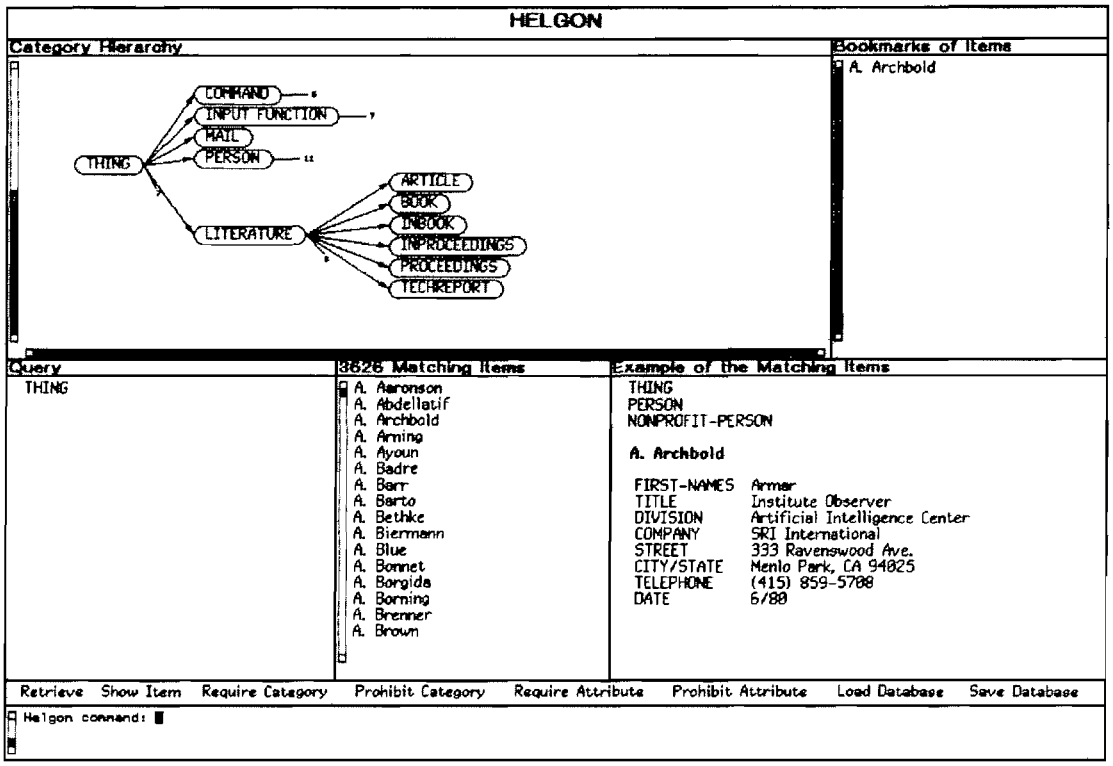


**Figure 4:** Initial State of the Sample Dialogue

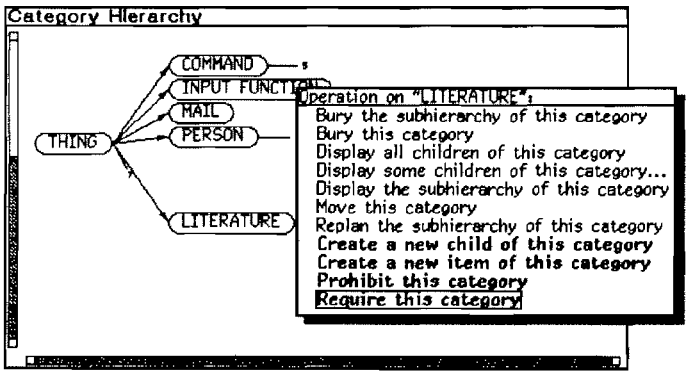


**Figure 5:** The Pop-up Menu on a Category in the Category Hierarchy Display

This figure shows the pop-up menu that appears on a category in the graphical category hierarchy display. The operations in roman are for manipulating the graphical display, the ones in boldface for modifying the knowledge base or the query. **`Require this category`** is selected to add the category **`LITERATURE`** to the query.

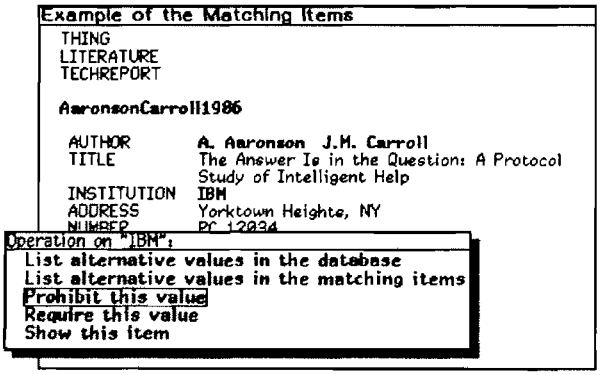


**Figure 6:** The Pop-up Menu on an Attribute Value

This figure shows the pop-up menu that appears on an attribute value of an item. It contains operations to require or prohibit this value in the query, to list alternative values for this attribute in the whole database or the matching items only (with the possibility to require or prohibit them), and, if the attribute value is an item itself, to show it. **`Prohibit this value`** is selected to prohibit **`IBM`** as **`INSTITUTION`** in the query.

## EVALUATION AND FUTURE PLANS

HELGON has been a joint research effort combining innovative system design and cognitive theory. An empirical evaluation of the information retrieval component of an earlier version of HELGON showed (for details see [5]) that

- Subjects wanted to directly enter information they knew from the beginning because finding the right example to be criticized can be tedious. This capability was therefore incorporated in a later version of HELGON.
- Subjects had problems understanding the terms used in the information store.
- Subjects had problems with the hierarchical organization of the information store.

These findings suggest extensions for future versions of HELGON. An explanation component describing the terms used can be added to increase users' understanding of the system model. We are also experimenting with an *inductive* retrieval algorithm, which returns a set of items that match the query to varying degrees [8].

Literature is a domain with which users are fairly familiar, and the entries in this domain are very homogeneous. The study showed that, after initial problems, users could use HELGON for literature searches without major difficulties. A challenge will be to apply HELGON to domains that are not as well-structured, homogeneous, or familiar to the users as the domain of literature. Our hypothesis, based on our experiences with the complex information store embedded in SYMBOLICS LISP machines, is that users will encounter more difficulties because the translation from the situation model to the system model is more difficult.

## ACKNOWLEDGEMENTS

The authors would like to thank the other members of the ARI project, especially Walter Kintsch, who contributed to the elaboration of the underlying theory, Peter Foltz who did the evaluation studies, and Andreas Lemke and Curt Stevens, who criticized our system-building effort and an earlier draft of this paper. Janet Grassia helped us edit the paper. We also thank Peter Patel-Schneider, who made ARGON and KANDOR available to us. The research was supported by Grant No. MDA903-86-C0143 from the Army Research Institute.

## REFERENCES

1. S.W. Draper. The Nature of Expertise in UNIX. *Proceedings of INTERACT'84, IFIP Conference on Human-Computer Interaction*, Elsevier Science Publishers, Amsterdam, September, 1984, pp. 182-186.

2. G. Fischer. Cognitive View of Reuse and Redesign. *IEEE Software, Special Issue on Reusability 4*, 4 (July 1987), 60-72.

3. G. Fischer. Mental Models -- A Computer Scientist's Point of View. In A.A. Turner (Ed.), *Mental Models and User-Centered Design, Workshop Report (Breckenridge, CO)*, Institute of Cognitive Science, University of Colorado, Boulder, CO, Technical Report, No. 88-9, 1988, pp. 15-26.

4. G. Fischer, W. Kintsch. *Theories, Methods and Tools for the Design of User-Centered Systems*. Department of Computer Science, University of Colorado, Boulder, CO, 1986.

5. P.W. Foltz, W. Kintsch. An Empirical Study of Retrieval by Reformulation on HELGON. In A.A. Turner (Ed.), *Mental Models and User-Centered Design, Workshop Report (Breckenridge, CO)*, Institute of Cognitive Science, University of Colorado, Boulder, CO, Technical Report, No. 88-9, 1988, pp. 9-14.

6. G.W. Furnas, T.K. Landauer, L.M. Gomez, S.T. Dumais. The Vocabulary Problem in Human-System Communication. *Communications of the ACM 30*, 11 (November 1987), 964-971.

7. F.G. Halasz. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM 31*, 7 (July 1987), 836-852.

8. M.C. Mozer. *Inductive Information Retrieval Using Parallel Distributed Computation*. ICS Report 8406, Institute for Cognitive Science, University of California, San Diego, La Jolla, CA, June, 1984.

9. H. Nieper-Lemke. TRISTAN, ein generischer Editor fuer gerichtete Graphen. In R. Gunzenhaeuser, H.-D. Boecker (Eds.), *Prototypen benutzergerechter Computersysteme*, Walter de Gruyter, Berlin - New York, 1988, pp. 243-257.

10. D.A. Norman, D.G. Bobrow. Descriptions: An Intermediate Stage in Memory Retrieval. *Cognitive Psychology 11* (1979), 107-123.

11. P.F. Patel-Schneider. *Small Can Be Beautiful in Knowledge Representation*. AI Technical Report 37, Schlumberger Palo Alto Research, October, 1984.

12. P.F. Patel-Schneider, R.J. Brachman, H.J. Levesque. *ARGON: Knowledge Representation Meets Information Retrieval*. Fairchild Technical Report 654, Schlumberger Palo Alto Research, September, 1984.

13. J.H. Walker. Document Examiner: Delivery Interface for Hypertext Documents. *Hypertext'87 Papers*, University of North Carolina, Chapel Hill, NC, November, 1987, pp. 307-323.

14. B.L. Webber, T.W. Finin. In Response: Next Steps in Natural Language Interaction. In W. Reitman (Ed.), *Artificial Intelligence Applications for Business*, Ablex Publishing Corporation, Norwood, NJ, 1984, Chap. 12, pp. 211-234.

15. M.D. Williams. What Makes RABBIT Run? *International Journal of Man-Machine Studies 21* (1984), 333-352.

16. M.D. Williams, F.N. Tou, R. Fikes, A. Henderson, T.W. Malone. RABBIT: Cognitive Science in Interface Design. *Proceedings of the 4th Annual Conference of the Cognitive Science Society (Ann Arbor, MI)*, Cognitive Science Society, August, 1982, pp. 82-85.

17. J. Yen, R. Neches, M. DeBellis. Specification by Reformulation: A Paradigm for Building Integrated User Support Environments. *Proceedings of AAAI-88, Seventh National Conference on Artificial Intelligence (St. Paul, MN)*, Morgan Kaufmann Publishers, San Mateo, CA, 1988, pp. 814-818.