RABBIT: An Intelligent Database Assistant

Frederich N. Tou
Michael D. Williams
Richard Fikes
Austin Henderson
Thomas Malone

Cognitive and Instructional Sciences Group
Xerox Palo Alto Research Center

Abstract

We have designed and implemented an intelligent database assistant to aid the user in formulating a query. The system, named RABBIT, relies upon a new paradigm for retrieval, retrieval by reformulation, based on a psychological theory of human remembering. To make a query, the user interactively constructs a description of his target item(s) by criticizing successive example (and counterexample) instances. One of the key innovations in RABBIT is that instances from the database are presented to the user from a well-defined perspective *inferred* from the user's query description and the structure of the knowledge base. Among other things, this constructed perspective prevents the user from creating semantically improper query descriptions. RABBIT particularily facilitates users who approach a database with only a vague idea of what it is that they want and who thus, need to be guided in the (re)formulation of their queries. RABBIT is also of substantial value to casual users who have limited knowledge of a given database or who must deal with a multitude of databases.

1. Introduction

RABBIT is an information retrieval interface which takes a new approach to information retrieval. The design of RABBIT began with an examination of ideas borrowed from cognitive science and knowledge representation. From those ideas, a new paradigm for information retrieval, retrieval by reformulation, has been developed, and a small experimental system based on that paradigm has been implemented in the Smalltalk programming language [Ingalls, 1978] on the Xerox Dolphin and Dorado personal computers [Lampson and Pier, 1980] and runs over a set of sample databases represented in KloneTalk [Fikes, 1981].

The motivation for designing a new kind of database interface was the unsuitability of existing database interfaces for casual users. Some database interfaces (e.g., SQUARE [Boyce et al, 1975] and SQL [Chamberlin et al, 1976]) require many hours of instruction to learn; others have a syntax which users find difficult to use and understand (e.g., the boolean expressions of DIALOG [Lockheed, 1979]). Interfaces based on the relational data model [Codd, 1970] usually require the user to know in advance which tables and attributes he will be needing, while users of network databases (such as ZOG [Robertson et. al., 1981]) frequently get lost during the course of their search.

RABBIT was designed to solve four problems which we conjecture to be major sources of difficulty for casual users attempting to retrieve information: (1) the user has incomplete knowledge about the *descriptive terms* needed to create a query, (2) the user's own intention is only partily articulated, (3) a considerable amount of information is known in the database about any given item, and hence, the presentation of that information needs to be limited or structured in some way, and (4) the structure of the database(s) is heterogeneous with the result that the 'shape' of the database changes depending upon where one is within the database.

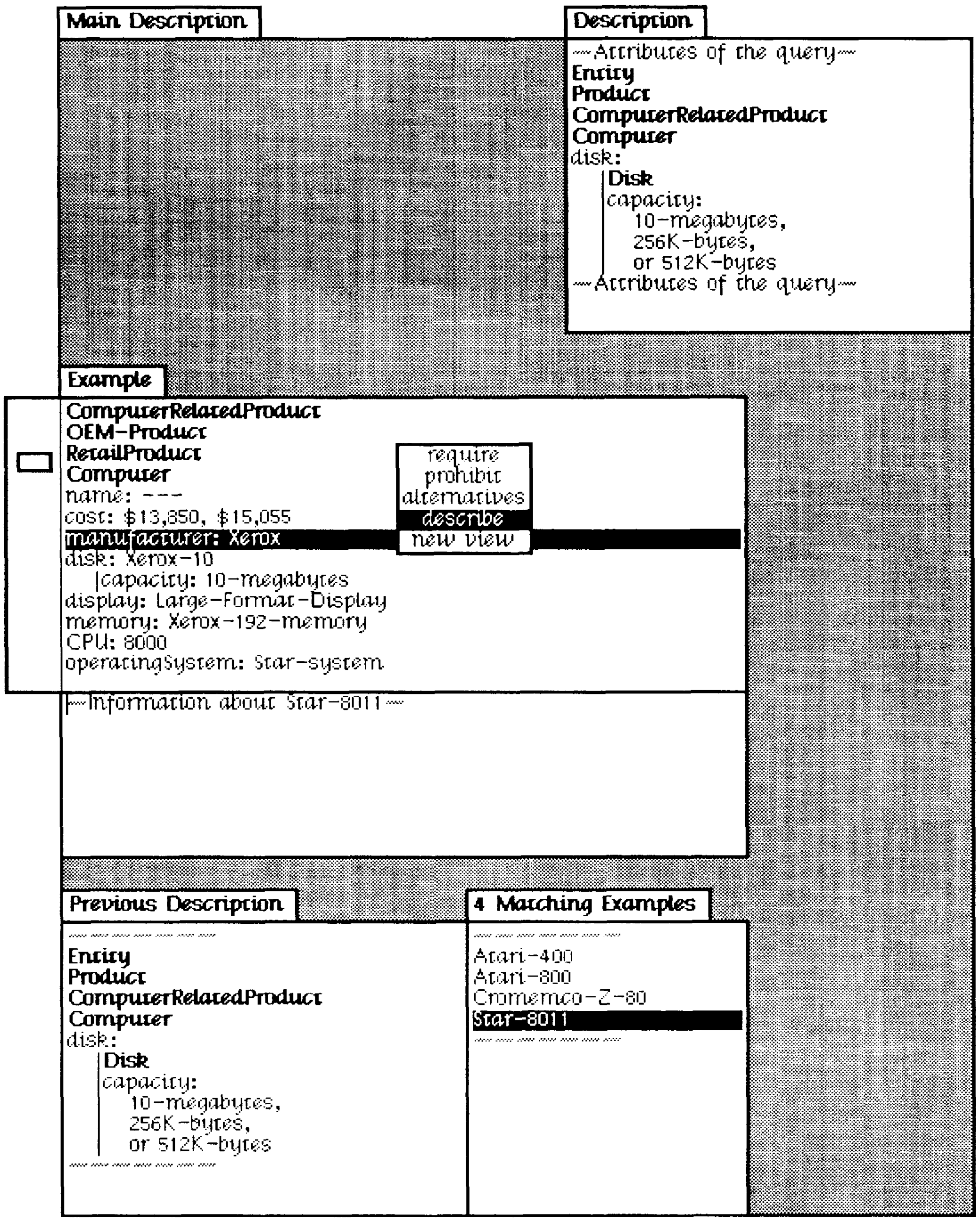
Two techniques of human remembering which RABBIT incorporates are *descriptive retrieval* and *retrieval by instantiation*. The basic tenet of descriptive retrieval is that people retrieve information from (their own) memory by iteratively constructing partial descriptions of the desired target item [Bobrow and Norman, 1975; Norman and Bobrow, 1979; Williams and Hollan, 1981; Williams, 1981]. Retrieval by instantiation postulates that the information retrieved each iteration of the retrieval process is in the form of an *instantiation*, i.e., an example item suggested (e.g., analogically or metaphorically) by the partial description [Williams, 1981].

2. Retrieval by Reformulation

The basic principle underlying RABBIT is a new paradigm for information retrieval elaborated from the notion of retrieval by instantiation—retrieval by reformulation. The user makes a query by incrementally constructing a *partial description* of the item(s) in the database for which he is searching. RABBIT provides a description of an *example instance*, an instance in the database which matches the user's partial description. The function of this example is to aid the user in articulating his tacit knowledge. The user can select the various descriptors from the example and incorporate those descriptors, or variations of those descriptors, into his partial description, thus, *reformulating* his initial query. This query-building process is iterative in that the user can at any time request the interface to retrieve a new example instance, one which matches the latest version of his (partial) description, and then use the descriptors of that new image to build up his query description further. As the user builds his query RABBIT is constructing (from the partial description) a perspective from which to present the next instance.

Figure 1 shows RABBIT in the midst of a retrieval interaction. The interface consists of four primary window panes. The 'Description' pane specifies an implicitly defined boolean expression which appears to the user as a partial description of the item(s) he is seeking. The 'Example' pane contains an example item which matches the partial description as of the last user initiated retrieval cycle from the RABBIT defined perspective. More precisely, it contains a description, called the *image*, of an instance from some well-defined *perspective* (e.g., "The Little Hsi Nan Restaurant" can be viewed from the perspectives of "a place which serves food," "an investment," and "a business."). The 'Matching Examples' pane lists instances which satisfy the partial description as of the last retrieval cycle. The 'Previous Description' pane contains the description used on the last retrieval cycle which determines the perspective for presentation of the example and the list of matching examples. The example pane command pop-up menu is also displayed.

The example instance mentioned above is a central element of the interface. It serves several purposes: it functions as a *template*, it permits *access* to additional descriptors, it provides *semantic resolution* of potentially ambiguous terms, and it frequently serves as a *counterexample*. The example instance is a template in the sense that its presentation (the image) provides a pattern for making a query via the descriptors comprising the instance's image. It permits access to new descriptive terms through the alternatives and describe commands



```
---Attributes of the query---
Entity
Product
ComputerRelatedProduct
Computer
disk:
   |Disk
   |capacity:
   |   10-megabytes,
   |   256K-bytes,
   |   or 512K-bytes
---Attributes of the query---
```

```
ComputerRelatedProduct
OEM-Product
RetailProduct
Computer
name: ---
cost: $13,850, $15,055
manufacturer: Xerox
disk: Xerox-10
   |capacity: 10-megabytes
display: Large-Format-Display
memory: Xerox-192-memory
CPU: 8000
operatingSystem: Star-system
|---Information about Star-8011---
```

```
Entity
Product
ComputerRelatedProduct
Computer
disk:
   |Disk
   |capacity:
   |   10-megabytes,
   |   256K-bytes,
   |   or 512K-bytes
```

```
Atari-400
Atari-800
Cromemco-Z-80
Star-8011
```

Figure 1. RABBIT Screen Display

elaborated below. It also provides semantic resolution in that the context of a term such as the role name 'manufacturer' establishes and refines the term's meaning. The role name 'manufacturer:' could refer to a person or a nation or a corporation. The statement 'manufacturer: Xerox' in the context of a description of a computer product resolves a host of potential meanings. The example instance is also a counterexample to the user's intentions since it is unlikely that the first instance retrieved will be exactly what the user is looking for. Rather than simply permitting the user to express his displeasure with the counterexample and have RABBIT try to guess what is wrong with it, the system tries to encourage the user to articulate what is wrong with the instance presented. The counterexample's simple presence serves to remind the user that his query description is incomplete or wrong and, in addition, point out the particular parts of his description which need correction or modification. Finally, since the amount of information known about the retrieved instance could be considerable, the information actually presented in the image is limited to be only that information which is known from a given perspective inferred from the query description which was the basis for retrieving that example instance. (E.g., information concerning the dinner menu or house specialty of a given restaurant would be available from the perspective of "a place which serves food" but not from the perspective of "a business." So if the user had begun his query with the descriptor '**Business**', then the image of the retrieved instance, even if it is a restaurant, would not, initially, include information about its dinner menu.)

The current implementation of RABBIT supports a small set (5) of basic operations for creating a query description given the descriptors provided in the image of the example instance. These operations, shown in figure 1, are require and prohibit (which specify that the given descriptor is or is not to be a descriptor of the retrieved instance, respectively), alternatives (which presents the user with a popup menu of alternative descriptors to the given one), specialize (which shows the specializations of the given descriptor), and describe (which allows the user to examine a description of a given descriptor or to describe recursively what that descriptor should be. ([Tou, 1982] and [Tou, Williams, Malone, Fikes, and Henderson 1982] contain a more complete discussion of the paradigm of retrieval by reformulation and the user interface to RABBIT.)

This paradigm of retrieval by reformulation, in effect, defines a form of interaction by which RABBIT can assist casual users in formulating queries. Much of the intelligence of RABBIT comes from control of this interaction by appealing to the conceptual structure of the database.

3. The KL-ONE Database

The KL-ONE epistemology for representing knowledge [Brachman, 1979] has had a major influence on the development of RABBIT. The experimental database which RABBIT accesses is a KL-ONE network. Our long term vision is to have RABBIT as a query assistant in a local machine with a local KL-ONE conceptual structure transforming the user's queries into acceptable forms to query remote databases. In a sense, what are now KL-ONE instances will be the data objects of these remote databases.

The descriptions which the user creates (the partial query description) and critiques (the instance descriptions, or images) are composed of two types of descriptors: instance classes and attribute-value pairs. Instance classes (shown in boldface in the text of this paper) denote general classes of instances (e.g., '**Business**', '**City**', and '**Entity**' are all instance classes), whereas attribute-value pairs specify the properties of a specific instance, and in particular, the relationships between that instance and other instances (e.g., the instance 'The-Little-Hsi-Nan' has attribute-value pair 'location: PaloAlto'). As was mentioned earlier, instances are the items in the database.

The correspondence between KL-ONE concepts and RABBIT descriptions is as follows. The instance classes comprising RABBIT descriptions are represented by KL-ONE generic concepts, while instances correspond to individual concepts. Attribute-value pairs are implemented as KL-ONE role-value pairs, and constraints (in the query description) on what values should be filling a attribute (called a value constraint) correspond to a value restriction on the corresponding role. In general, a query description in RABBIT corresponds to a (possibly new) generic concept which subsumes the (individual concepts corresponding to the) instances matching that query description.

The fact that KL-ONE is an inheritance network allows heterogeneous data to be easily represented in a structured format. So, for example, information common to a set of individual concepts can be factored out and associated with a generic concept subsuming those individual concepts, while the individual concepts retain that information which distinguishes them from one another.

In addition to the ability to construct perspectives, described below, we have been able to use the KL-ONE semantics to control the search for alternative descriptors when the user issues the alternatives command. The alternatives to a given instance class were originally those generic concepts which were "brothers" of the generic concept corresponding to the given instance class. But the number of brothers could be quite numerous, especially near the "top" of the heterarchy (i.e., close to the generic '**Entity**'). However, KloneTalk allows the builder of the database to partition generics into *decompositions*. So, for example, subconcepts of '**Restaurant**' such as '**ChineseRestaurant**', '**FrenchRestaurant**', and '**GermanRestaurant**' can be grouped under the decomposition 'cuisine', while subconcepts like '**PaloAltoRestaurant**', '**MountainViewRestaurant**', and '**LosAltosRestaurant**' can be grouped under the 'location' decomposition. Then if the user asks for alternatives to '**FrenchRestaurant**', he is shown only those alternative concepts which belong to the same decomposition as '**FrenchRestaurant**'.

4. Perspectives

One of the main uses of KL-ONE is the implementation of *perspectives*. A perspective is simply a way of describing an event or item from a particular viewpoint [Bobrow and Norman, 1975, Bobrow and Winograd, 1977, Goldstein and Bobrow, 1980, Goldstein, 1980]. We introduce the notion of a dynamic perspective; thus, the perspective from which the user views the instances in the database changes depending on his partial description and on where he is within the database. In RABBIT a perspective specifies which descriptors (instance classes and attribute-value pairs) should be included in the image of any instance presented to the user.

There are two distinct mechanisms RABBIT uses to construct a perspective. First it filters the attributes to be presented to a user by including only attributes implicitly acknowledged by the user. Since the partial description is a representation of the user's intent to the computer, that description should be the basis for determining what information should be included in the image of the example instance. In RABBIT the attributes included in the image are exactly those attributes which belong to the instance classes occurring in the partial description, while the instance classes in the image are the specializations of the instance classes in the partial description. Thus, if one were to see the computer descibed in figure 1 retrieved under the partial description **'Product'** (i.e. without the descriptor **'Computer'**) then only the attributes 'name', 'manufacturer', and 'cost' would be presented. Once the user refines the partial description to specify that he is seeking a computer, additional roles (e.g. 'disk,' 'cpu,' ...) would appear. A second mechanism for creating perspectives actually extends the perspective of any given instance beyond attributes directly held by the object. Note in figure 1 that because the user has created an embedded description about the disk of the computer sought, aspsects of the disk that the user considers important (e.g. capacity) have been compressed into the image of the computer presented.

Perspectives serve four main functions in the RABBIT interface:

Perspectives are used to control the amount of information presented to the user. As we mentioned earlier, the amount of information known about any given instance in the database could be substantial. In some databases we have considered it runs to the hundreds. The presentation of that information must be limited in some fashion.

Perspectives are also useful for facilitating the user's understanding of instances since the information provided in the image is with respect to a well-defined perspective (determined by the partial description). The set of attributes shown in the image should provide a fairly coherent and cohesive view of the example instance since that set arises from the definition of (a generic concept corresponding to) an instance class, which presumably has semantic integrity. We sometimes refer to this effect as *semantic resolution.* The name of an attribute standing alone might be ambiguous, but since an attribute commonly appears with other, related, attributes within some context (the perspective), the user may be able to infer the meaning of an unknown attribute from its context.

A third function of perspectives is to enforce a certain class of semantic consistency. As we said earlier, instances are always being viewed from some perspective inferred from the partial description. Consequently, the user does not have access to a particular attribute unless he has first made the instance class owning that attribute a required descriptor of his partial description. Thus, only attributes which are "appropriate" or "relevant" to his partial description are available to the user for inclusion in the partial description. For example, if **'Book'** and **'Science-Fiction'** are instance classes in the user's partial description, then there would be no way for the user to add the attributes 'employees: ' or 'CPU: ', which are not attributes of either **'Book'** or **'Science-Fiction'**, to his partial description. This notion extends even further in the use of embedded descriptions. For example, suppose the user sees the attribute-value pair 'disk: Xerox-10' in the image of the example instance Star-8011, but does not know what the Xerox-10 disk is. One option the user has is to examine the description of the Xerox-10 using the describe command and then create a description of the disk he desires for his computer. But the description of what value should fill the attribute 'disk: ' can not be any description, such as the description of a restaurant, but rather, the description of an object which can be the disk of a computer. The interface enforces that constraint by imposing the constraint that the description of the disk which the user creates must include as part of its descriptors the instance class(es) corresponding to the generic concept(s) which are the value restrictions of the role corresponding to the attribute 'disk: '. In addition to limiting what kind of description can be created, those instance classes also provide a set of attributes which indicate to the user some of the ways in which he can describe the object, in this case, a disk for a computer.

Finally, perspectives can be used to manage a non-uniformly structured database. If a database is non-uniform, then it should change as the user moves around within the database with the consequence that the kind and amount of information should also change. So as the user refines or modifies his partial description by adding or removing new instance classes, he is also changing the perspective for viewing example instances, and hence, the attributes which appear in the image. In particular, adding new instance classes has the effect of adding *new attributes* to the image of the next example instance retrieved. Consequently, the user does not need to concern himself with the shape of the database as expressed in the attributes of instance classes; all he must do is decide on the conceptual shape (e.g., is the user looking for a business, a product, or something else), and the corresponding attributes are made available to him automatically.

5. Future Work

We currently have a running implementation of RABBIT accessing a database of approximately 200 individual concepts and 50 generic concepts. With respect to further utilization of KL-ONE, we anticipate that KL-ONE structural descriptions are a means for supporting general constraints within query descriptions, but there are problems which need to be solved such as the operation and appearance of the user interface for creating and editing constraints. A more general issue is the question of how perspectives are created. In the current implementation, each instance class defines a perspective for viewing instances of that class. But this tight coupling between perspectives and instances seems to be too restrictive. It may be that the organization of perspectives should really be orthogonal to the organization of instance classes.

A broader area requiring further work is implementing RABBIT for a "real" database, by which we mean a moderately sized database (with thousands of items) which is changing. The current version of RABBIT can not be used to modify the database, but we feel that the ideas underlying RABBIT could be easily adapted for inserting and deleting data. And finally, RABBIT needs to be tested on actual users to determine its strengths and weaknesses—which ideas are useful and which should be modified or even discarded.

6. Summary

This paper has described an intelligent database assistant named RABBIT which relies on a new paradigm for information retrieval, *retrieval by reformulation*, based on a psychological theory of human remembering. The four main ideas underlying this paradigm are:

1) retrieval by constructed descriptions
2) interactive construction of queries
3) critique of example instances
4) dynamic perspectives.

The first three of these ideas had their origins in human psychology, but the development of the fourth idea above—dynamic perspectives—was motivated and influenced strongly by the KL-ONE knowledge representation language. One of the key ideas in RABBIT is the use of user interaction and structure of the database to construct a perspective from which to present any given instance. In RABBIT we have used perspectives as a mechanism for:

The first three of these ideas had their origins in human psychology, but the development of the fourth idea above—dynamic perspectives—was motivated and influenced strongly by the KL-ONE knowledge representation language. One of the key ideas in RABBIT is the use of user interaction and structure of the database to construct a perspective from which to present any given instance. In RABBIT we have used perspectives as a mechanism for:

--controlling the type and amount of information presented
--facilitating the user's understanding of instances
--enforcing certain kinds of semantic consistency
--organizing and managing heterogeneous data.

Our experimental implementation of RABBIT looks very promising, but only usage by real users can determine the effectiveness and usefulness of the paradigm of retrieval by reformulation.

Our experimental implementation of RABBIT looks very promising, but only usage by real users can determine the effectiveness and usefulness of the paradigm of retrieval by reformulation.

Acknowledgements

A portion of this work was carried out by the principal author under the auspices of the MIT intern program at Xerox PARC and is based on a thesis submitted in partial fulfillment of the requirements of the degrees of Bachelor of Science and Master of Science in the department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology in June, 1982. The authors would also like to acknowledge the original stimulus for this work steming from an exciting conference on artificial intelligence and human-computer interfaces sponsered by the Army Research Institute. In particular, we would like Stan Halpern, Janet Kolodner, and Albert Badre to know a part of what came from their efforts. We would also like to thank John Seely Brown, Tom Moran, Rick Cattell, Laura Gould, and Richard Burton for their patient discussions and guidance. Each contributed crucial pieces of the puzzle we are still putting together. Finally, we would like to thank the other members of the Cognitive and Instructional Sciences Group at Xerox PARC for their continuing support and critique throughout the development of RABBIT.

References

Bobrow, D.G., and Norman, D.A. "Some Principles of Memory Schemata," in D.G. Bobrow and A.M. Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science.* New York: Academic Press, 1975.

Bobrow, D.G., and Winograd, T. "An Overview of KRL: A Knowledge Representation Language," *Cognitive Science,* **1**, pp. 3-46, 1977.

Boyce, R.F., Chamberlin, D.D., King, W.F., and Hammer, M.M. "Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage," *Communications of the ACM* **18**, 11 (Nov. 1975), pp. 621-628.

Brachman, R.J., Bobrow, R.J., Cohen, P.R., Klovstad, J.W., Webber, B.L., Woods, W.A. "Research in Natural Language Understanding: Annual Report, 1 September 1978 to 31 August 1979," *BBN Report No. 4274.* Cambridge, MA: Bolt Beranek and Newman Inc., August, 1979.

Chamberlin, D.D., Astrahan, M.M., Eswaran, K.P., Griffiths, P.P., Lorie, R.A., Mehl, J.W., Reisner, P., and Wade, B.W. "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control," *IBM Journal of Research and Development* **20** (Nov. 1976), pp. 560-575.

Codd, E.F. "A Relational Model of Data for Large Shared Data Bases," *Communications of the ACM* **13**, 6 (June 1970), pp. 377-397.

Fikes, R. "Highlights from KloneTalk: Display-Based Editing and Browsing, Decompositions, Qua Concepts, and Active Role-Value Maps," Proceedingings of the 1981 KL-ONE Workshop, Jackson, New Hampshire, October, 1981.

Goldstein, I.P. "PIE: A network-based personal information environment." *Proceedings of the Office Semantics Workshop,* Chatham, Mass., June, 1980.

Goldstein, I.P., & Bobrow, D. Descriptions for a programming environment, *Proceedings of the First Annual National Conference on Artificial Intelligence,* Stanford, CA, August, 1980.

Ingalls, D.H. "The Smalltalk-76 Programming System: Design and Implementation," *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages,* Tucson, AZ: January 1978, pp. 9-16.

Lockheed Information Systems. *Guide to DIALOG Searching,* Palo Alto, CA, 1979.

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

Robertson, G., McCracken, D., and Newell, A. "The ZOG Approach to Man-Machine Communication," *International Journal of Man-Machine Studies* (1981) **14**, pp. 461-488.

Tou, F. *RABBIT: A novel approach to information retrieval,* unpublished M.S. thesis, Massachusetts Institute of Technology, Cambridge, Mass., forthcoming.

Tou, F.N., Williams, M.D., Malone, T.W., Fikes, R.E., and Henderson, A. RABBIT: an Intelligent Interface. Xerox Technical Report, forthcoming, 1982.

Williams, M.D. "Instantiation: A Data Base Interface for the Novice User," Xerox Palo Alto Research Center Working Paper, 1981.

Williams, M.D., and Hollan, J.D. "The Process of Retrieval from Very Long Term Memory," *Cognitive Science* 5 (1981), pp. 87-119.