1. INTRODUCTION

A reference expert system may be considered to be a system with a knowledge base covering various aspects of the reference process in a library setting. Knowledge bases generally consist of several components (such as databases, rule bases, frames, and semantic nets) that interact with an inference engine, a user interface, and each other. This paper will examine progressively more complex knowledge-based systems for reference that can be constructed from components like these, concentrating at first on combinations of databases and rule bases. This examination will lead to a classification of reference expert systems.

In Section 3, very simple architectures of a type common in other fields will first be considered. Arguments drawn from reference theory suggest that these simple architectures are appropriate primarily in dealing with directional reference transactions. In Sections 4 and 5, reference theory will be used to develop two additional architectures more appropriate to other reference transactions, such as ready-reference transactions. The classification of reference expert systems will be completed in Section 6 by examining further reference theory and then using it to develop variants on the three basic types. Section 7.1 will discuss briefly the use of reference knowledge bases for computer-assisted instruction. Section 7.2 will consider deep reference knowledge. The paper will conclude with some prognostications about future developments.
Fundamental to this paper is the premise that an examination of the design of reference expert systems may profitably be guided by the experience embodied in existing models of the reference process. This point of view, if not the exact analytical approach adopted here, has been expressed previously (Parrott, 1990). The classification that is based on these models was modified after presentation at the Clinic as a result of the suggestion there by Charles Bailey (University of Houston) and Lloyd Davidson (Northwestern University) that the scheme be extended to include hypertext systems. That led to a consideration of several other issues, including combinatorial completeness of the scheme. The end result is a classification significantly richer than that presented before the Clinic audience.

2. PRELIMINARY CONSIDERATIONS

2.1 Answers in the Reference Process

Because reference work is often a multistage process, with intermediate results before the desired information is obtained, the concept of answer can be somewhat ambiguous. At least three major types of answers can be distinguished:

1. **Desired Information.** For example, for a ready-reference question, an address might be the desired information. This type of information might be called the "final answer."

2. **Bibliographic Information** about reference tools, books, and other materials (printed or electronic) that experience has shown could contain the desired information. An example is bibliographic information about *Encyclopedia of Associations*. This kind of bibliographic information is generally referred to in this paper as "titles of information sources" (although more than the titles is intended), or abbreviated simply as "information sources." This kind of information may be thought of as an intermediate answer of the form: "The information you want is probably found in *Encyclopedia of Associations."

3. **Categories of Information Sources** that experience has shown could contain the desired information. For example, the experience of many reference librarians is that trade directories are a useful category of information source giving addresses of manufacturers. This kind of information is referred to as "type of information source" in this paper. This kind of information may also be thought of as an intermediate answer, one especially useful when instruction is important. It may be considered to be an intermediate answer of
the form: "The kind of information you want is generally found in trade directories."

2.2 Differences Between Databases and Rule Bases

Databases and rule bases embody different ways of organizing knowledge. Suppose a piece of knowledge about reference consists of certain descriptors (subject, geographical area, etc.) and associated answers (hours of opening, biographical information, etc.). A database approach puts that knowledge into records, with fields for the descriptors and fields for the answers. Below, several approaches to storing information in databases are distinguished.

1. A page-based approach, where the answer field contains some text and answers (possibly enough to fill a page or screen); the answers are not labeled as such to distinguish them from the text. In general, more than one answer is found in each answer field; conversely, one answer may appear in more than one record. For example, a record might give hours of opening for several branch libraries, or refer to several reference tools; the answers will not be labeled to distinguish them from other text (such as "The following are the hours of opening...").

2. A hypertext-based approach, where the answer field contains some text and answers (possibly enough to fill a page or screen); the answers are labeled to distinguish them from the text. The situation is identical to the page-based approach, except that individual answers are now labeled.

3. A single-answer-based approach, where the answer field contains some text and one answer. The answer is not labeled to distinguish it from the text. The situation is identical to the page-based approach, except that the answer field for a record contains only a single answer, which appears, moreover, only in that one record.

4. An item-based approach, where the answer field for a record contains nothing but one answer, which appears, moreover, only in that one record. For example, a record will give hours of opening for only one branch library, or refer to only one reference tool. A variant of this allows the answer field to have text, but requires the answer to be labeled.

A rule-based approach puts that knowledge into rules: the descriptors in the IF clauses and the answers in the THEN clauses. It is quite possible for the same answer to appear in the THEN clauses of several rules.

Notice two critical features in the above:

a. The possibility that a single answer appears in several places
in a file. If this is so, then it may be considered a disadvantage, since updating the answers will not necessarily be easy. But it may also be considered an advantage, as the following argument shows. Suppose we attach weights (confidence factors) to each answer. That is, for the set of descriptors, we have $X$ confidence that the answer will be useful. But if a single answer may appear in several places in the file, then we can assign it several different weights: one for each set of question descriptors. This is a much more realistic way of assigning weights to answers than simply assigning one weight to each. So, if a single answer may appear in several places (as in a rule base, a hypertext database, or a page-based database), updating may not be easy, but realistic weights are possible. Conversely, if a single answer appears in only one place (as in a single-answer-based database or an item-based database), then updating is easy, but realistic weights are not possible.

![Diagram](image)

**Figure 1.** A type 1R model with a backward-chaining rule base
There is a further point that needs to be considered. If, indeed, a single answer appears in several places in a file, rather than one, then the file may serve to eliminate possibilities in the following sense. Suppose, first, that a single answer can appear in only one place in the file. Then that answer will have a fixed set of descriptors attached to it. But if it could occur in several places in the file, it could have different sets of descriptors associated with it. Now, assume that these sets of descriptors are consistent with that single set in the former arrangement. But these sets do not need to exhaust all the possibilities of the former arrangement.

For example, suppose in the former arrangement that a tool is assigned descriptors such as: question-type = biographical, geographical-area = U.S.A., subject-area = chemistry, sector = academic, alive-or-dead = alive. But in the latter arrangement, there might be one place where the given answer is assigned only question-type = biographical, geographical-area = U.S.A., and subject-area = chemistry. And the only other occurrence of the answer might have question-type = biographical, sector = academic, and alive-or-dead = alive. Consider a question with question attributes: question-type = biographical, subject-area = chemistry, alive-or-dead = alive. These question attributes will match the correct tool in the first arrangement, but not in the second, since neither of the two occurrences of that tool in the second arrangement have the given cluster of attribute/values.

b. The possibility of identifying the answer field inside the answer field with precision. If this is so (as in an item-based database, a hypertext database, or an appropriately constructed rule base), then (1) it is possible to link a particular answer up with an external database, which will be considered later on, and (2) it is possible to assign a weight directly to a particular answer in the answer field, even if there are several answers in that field (that is, we have more realistic weighting than otherwise). Unlike the feature previously considered, there is an advantage only when this feature has a positive value (when precision exists). When this feature has a negative value (when precision is lacking), then there is no advantage.

We have thus identified two important features, each with two values. Of the resulting four values, three are advantageous in certain situations.

3. SYSTEMS WITH ONE SELECTION OPERATION

This section treats the simplest structures possible for a knowledge-
based system. These structures may have components like a database, a rule base, or a combination of both. But in these simple structures, the final answer is determined by a match (or selection operation) of the question attributes against only one of these components. Systems using only one selection operation in this manner will be called Type 1 systems.

Although, as will be seen, these structures have significant limitations, there are several motivations for beginning with structures as simple as this. The first motivation is a pedagogical one: applications of databases and rule bases as simple as this are easy to understand. The second is a practical one: basic structures like this are easy to implement using available software. The third motivation is an imitative one: these structures have proven useful in other fields.

3.1 Type 1 Systems with One Component and Realistic Weights

As noted in Section 2.2, realistic weights are possible in components such as a rule base (R), a hypertext database (H), and a page-based database (P). Systems built from only one of these components may be called 1R, 1H<sub>s</sub>, and 1P<sub>s</sub> systems, respectively. (The subscript s stands for matching against scope attributes.) Updating will not necessarily be easy in systems like this.

Suppose one has a Type 1R system. As an example of its operation, consider a transaction in which a user wants to know the hours of opening of a particular branch library, the Botany Library. Suppose further that the rule base contains the following rule:

IF the question type = hours-of-opening and the branch-library = botany
THEN the answer is "Monday to Friday, 8:30 AM to 10:00 PM; Saturday and Sunday, 1:00 PM to 6:00 PM."

If the inference engine uses backward chaining, then it will pick its rules one by one, and ask the user questions to determine the values of the attributes. When the engine reaches the rule above, if the user has not already revealed the value of question-type and branch-library, the inference engine will ask the user for these values. If the values match hours-of-opening and botany respectively, then the answer given in the THEN clause will be quoted. If the values do not match, another rule will be examined and more questions asked, if necessary.

If, on the other hand, the inference engine uses forward chaining, then it will ask the user a series of questions (using either a set of menus or frames), and then do a match against the entire rule base. If the user has given hours-of-opening as the question-type and botany as the branch-library, then a match is obtained on the rule mentioned above. The answer given in the THEN clause will be quoted. No more questions need to be asked, since they have all been asked at the beginning.
In Type 1Hₜ and 1Pₜ systems, the internal operation will differ from that of Type 1R systems. But all three cases share the following features:

1. the question attributes are somehow determined from the user;
2. they are matched against either the IF clauses of rules in a rule base, or the descriptors in a page-based database or a hypertext database;
3. the answer is in the THEN clause of the rules, or the answer field in the database; and
4. a given answer may appear in more than one place in the rule base or database, hence allowing for realistic weights, but also allowing problems in updating the answers.

3.2 Type 1 Systems with One Component and Easy Updating

As noted in Section 2.2, easy updating of answers is possible in components such as a single-answer-based database (S) or an item-based database (I). Systems built from only one of these components may
be called 1S\textsubscript{s} and 1I\textsubscript{s} systems, respectively. (Again, the subscript \textit{s} stands for matching against scope attributes.) Realistic weights will not be possible in systems like this.

In systems like this, question attributes are matched against a database (S or I) of factual information. Since no matching is done against a rule base, the question attributes cannot be determined through backward chaining by an inference engine. As in forward chaining on a rule base, either a set of menus or frames must be used. Once the question attributes have been determined, their values are matched against the attributes (or the scopes) of the database records. The matched records are then used in constructing the display used to answer the question.

Suppose one has a 1I\textsubscript{s} system. As an example of its operation, consider the same kind of question as before, namely, a transaction in which a user wants to know the hours of opening of a particular library branch, the Botany Library. Suppose further that the database contains a record including the following fields and values:
• question-type = hours-of-opening
• branch-library = botany
• answer = "Monday to Friday, 8:30 AM to 10:00 PM; Saturday and Sunday, 1:00 PM to 6:00 PM."

The question attributes gathered through the reference interview will match this record. The system will then display the contents of the answer field, which contains the hours of opening of the Botany Library.

The situation for a 1S, system is the same, except that the answer field will contain not only the answer proper, but additional text as well, and the former is not labeled to distinguish it from the latter. For example: answer = "Hours of opening of the Botany Library: Monday to Friday, 8:30 AM to 10:00 PM; Saturday and Sunday, 1:00 PM to 6:00 PM."

3.3 Type 1 Systems with More Than One Component

In Section 2.2, we concluded that three values of features were advantageous in some situations: realistic weights, easy updating, and precision. Considered in Section 3.1 were Type 1 systems with realistic weights, but not easy updating; some had precision, some did not. In Section 3.2, we considered Type 1 systems with easy updating, but not realistic weights; again, some had precision, some did not. In both those sections, we looked at one-component systems. The question arises: Is it possible, by considering systems with more than one component, to generate the other combinatorial possibilities? In particular, can one construct Type 1 systems that have both realistic weights and easy updating or neither?

Examine the possibilities in two-component systems. The first case is a system constructed of two components each of which allows realistic weights but not easy updating. A bit of reflection shows that such a system is equivalent to the systems in Section 3.1. As an example, consider one in which the first component is a rule base and the second is a page-based database, denoted Type 1RP. The system will first determine the question attributes, perhaps using a set of menus. Then the attributes will be matched against the IF clauses of the rules in the rule base. The THEN clause of a matching rule will point to a page in the database. Since both the components allow the specification of realistic weights, the total system will certainly allow it too. But since neither component allows easy updating, the total system cannot allow it. Hence the system is equivalent to those in Section 3.1.

The second case is a system constructed of two components each of which allows easy updating, but not realistic weights. Similarly, such a system is equivalent to the systems in Section 3.2. The third case
is a system whose first component allows easy updating but not realistic weights, and whose second component allows realistic weights but not easy updating. Such a system is equivalent to the systems in Section 3.1, since it clearly allows realistic weights but not easy updating (since the answer that must be updated lies in the second component).

![Diagram](image)

Figure 4. A type 1RI or 1RS model with a backward-chaining rule base

The fourth case is the interesting one. Here, the first component allows realistic weights but not easy updating, and the second component allows easy updating but not realistic weights. Such a system allows both realistic weights (since one of the two components allows it) and easy updating (since the answer to be updated lies in the second component, which allows easy updating). Before proceeding to examine this case in detail, it should be noted that we have not been able to generate the combination in which neither realistic weights and easy updating hold.

A system corresponding to the fourth case operates as follows. It matches question attributes against a rule base (or H database) as in Section 3.1, but does not store the final answers in the rules (or records
of the H databases). Instead, each rule (or record) points to one or more entries in an I or S database of final answers, which is consulted in constructing the display used to answer the question. Note that the first component allows the assignment of realistic weights to the answers, and the second component allows easy updating of the answers.

Finally, note that P databases have been deliberately excluded as the first component in this type of system. Although a P database will allow realistic weights, it lacks precision in specifying the answer; hence it cannot make a proper connection with a second component. To see this, suppose that the system used a page-based database to determine pointers. But then any given page might have several pointers on it, with no clear indication (to the system) where on the page the pointers occur. It would then be impossible for the system to determine what in fact the pointers actually are; the connection to the second component would thus not exist.

Suppose we have a 1RI system. As an example of its operation, consider the same kind of question as before, namely, a transaction in which a user wants to know the hours of opening of a particular library branch, the Botany Library. Suppose further that the rule base (the first component) contains the rule:

IF the question has certain attributes,
THEN go to record Y in the database of items for the factual information.

Somehow, the rule base carries out the reference interview, determining the question attributes, which match the above rule. That rule points to a record in the database (the second component); the answer field of the record is then displayed, giving the hours of opening of the Botany Library.

All other types of systems like this (Types 1RS, 1HI, and 1HS), will also clearly allow both realistic weights and easy updating.

3.4 Comparison of Type 1 Models

In Section 2.2 were identified three advantageous features that a system might have: realistic weights, easy updating, and precision in specifying answers. Before comparing the various types of systems described in the last three sections, let us consider whether all three of these features are useful in a Type 1 system. These Type 1 systems correspond to a model of a particular type of reference transaction put forward by William Katz (1982, pp. 72-75), which may be called Case 1 of Automatic Retrieval. The basic idea here is that, in some transactions, after data gathering (the reference interview), the data are used to extract the final answer from the librarian's memory. No recourse to intermediate information sources or reference tools is necessary. Hence, transactions like this will generally be directional in nature, giving
locations, hours of service, and so forth. But in directional questions, the answers are unlikely to involve uncertainty: if the librarian is unsure, she will check some source (and therefore go beyond her memory and the bounds of a Type 1 transaction). Yet if no uncertainty is involved, then there is little point in assigning weights (confidence factors). From this, it follows that weights are not particularly useful in Type 1 systems.

What about precision in specifying the answer? With precision, the answer field either labels the answers (to distinguish them from additional text) or contains only one answer with no additional text. Without precision, the answer field may contain additional text that, for example, might recapitulate the question attributes, or name the field (as in “The hours of opening of the Botany Library are . . .”). As noted in Section 3.3, precision is necessary in the first component of a two-component system; it is not necessary in the second component, which is where the final answer lies. It is difficult to imagine a situation when precision in the final answer is essential in a Type 1 system, since the information in the answer field will be processed by a human being (not another system component), and humans are easily able to parse the answer proper from additional text. In Type 2 and 3 systems, the information in this answer field is not necessarily going to be processed by a human, so this argument will not hold there. In Type 1 systems, however, precision in the final answer is irrelevant.

In conclusion, there is only one important feature distinguishing the performance of Type 1 systems: ease of updating. We may therefore compare our systems as follows:

1. Easy updating: Types 1I_s, 1S_s, 1RI, 1RS, 1HI, and 1HS.
2. Not easy updating: Types 1R, 1H_s, and 1P_s.

3.5 Implementations of Type 1 Models

The implementations identified below appear to be restricted to Type 1P_s and 1H_s models. Some of the systems categorized as Type 1P_s systems might, however, actually be of other kinds, such as Type 1I_s. The situation is not entirely clear, since the system descriptions in the literature do not always provide adequate details of implementation.

1. An early system, REFLES, handles factual data such as data associated with directional transactions (Bivins & Palmer, 1980). It uses a page-based database indexed by subject, and hence is of Type 1P_s. Bivins was associated with another system that handles factual information, REFLINK (Bivins & Eriksson, 1982), which uses a page-based database with access via a subject index or a tree structure of menus. It is also of Type 1P_s.
2. The Reference and Information Station (Purdue University Undergraduate Library) has menu access to a page-based database of factual information for answering directional questions (Smith & Hutton, 1984; Smith, D., 1989). It is therefore of Type 1P_s.

3. The Information Function (IF) at Carnegie-Mellon University provides (within the online catalog) menu access to page-based information on library announcements, locations, services, and tips in using the catalog (Diskin & Michalak, 1985). It is thus of Type 1P_s.

4. ORA (Online Reference Assistance) at the University of Waterloo Library (Parrott, 1986), has menu and keyword access to page-based directional information and other features as well. It is thus of Type 1P_s.

5. The Information Machine (Fadell & Myers, 1989) at the University of Houston Library has menu access to a database of pages. Its pages contain directional-type information (locations, times, regulations, phone numbers) and other features. So, it is of Type 1P_s.

6. The Apple Library Tour (Ertel & Oros, 1989) uses a hypertext database to provide directional and other information. It is thus of Type 1H_s.

4. SYSTEMS WITH TWO SELECTION OPERATIONS

Simple expert systems in many other fields are able to operate quite satisfactorily using Type 1 architectures, that is, they are able to do one match and then provide the final answer. In reference work, this type of direct provision of factual information (i.e., without recourse to a reference tool) will generally be confined to answering directional transactions—that is, requests for directions, information about local services, hours of opening, etc. Much of the expertise of a reference librarian, however, is in locating information sources that may contain the required information, rather than in knowing the required information itself. More complex architectures are needed for this; they may be combined with Type 1 architectures to allow directional questions, too.

The salient feature of these complex architectures, then, is that they allow information sources to be prescribed as intermediate answers before obtaining a final answer. From this, four parameters of system behavior of Type 2 systems emerge. To show this, we consider the original three advantageous features discussed in Section 2.2, and see which, if any, are valid in a Type 2 system (they all are). We then see whether any other feature might be advantageous (an additional feature is uncovered).

1. The fact that an information source may not contain the required
information introduces an element of uncertainty into these considerations that was absent in Section 3. Realistic weights may now be important. Indeed, the ability to rank information sources by likelihood of success is a mark of an experienced librarian.

2. Ease of updating may be a concern in Type 2 systems as well, since bibliographic data on an information source may change with time. As before, updating will be easiest when an intermediate answer appears in only one location.

3. If we want the system to use knowledge about particular intermediate answers (information sources) to perform actions, then the system must have precise access to that knowledge. That is, it must know that a certain string of characters in a field corresponds to the title of an information source. This will allow us, for example, to link up to an external online CD-ROM database. In the latter case, we will have a full implementation; if a person must leave the terminal to consult the tool, we will have a partial implementation.

4. In the previous three points, we have reconsidered the three advantageous features discussed in Section 2.2. Another advantageous feature, peculiar to information sources, may now be added. Information about information sources is generally more structured than a final answer. In particular, we may have indexing attributes which tell us which fields in the information source are indexed. This may be important, since it might affect search time. Also, if the system knows which fields in an information source are indexed, then it will be able to deduce how the source should be searched (e.g., "search index A on value b" or "browse for value b"). For a partial implementation, this will be given only as part of the prescription to the user; for a full implementation, it will allow the expert system some control over the second matching operation, that on the information source itself.

So, for some (if not all) complex architectures, important features of system behavior include: realistic weights, ease of updating, precision, and the indexing attributes of an information source. Sections 4 and 5 each consider a particular model (both derived from Katz) of more complex reference transactions in which information sources are to be consulted.

The first model to be considered may be called a Type 2 model. It is derived from Case 2 of Katz's Automatic Retrieval Model (Katz, 1982, pp. 72-75). The basic idea here is that, after information gathering (the reference interview), the data are used to determine one or more information sources that may contain the desired factual information; the sources are then consulted. This paradigm might apply to directional transactions where the librarian has to consult an information source
(not necessarily cataloged; possibly an in-house publication). The paradigm also applies to ready-reference transactions and substantive transactions.

Before going any further, it is useful to note that, although the attributes of final answers in Type 1 systems are of one kind only, intermediate answers in Type 2 systems (information sources) may have two distinct kinds of attributes: scope attributes (subject, geographical-area, etc.) and indexing attributes (which fields are indexed).

4.1 Type 2 Systems with Realistic Weights

It should first be observed that we have no control over the design of the final database used by a Type 2 system. Hence, when we speak about realistic weights (or easy updating, later), we intend the behavior of the first subsystem, that involved in the selection of an information source. As noted in Section 2.2, realistic weights are possible in components such as a rule base (R), a hypertext database (H), and a page-based database (P). Systems whose first subsystem is built from only one of these components may be called 2R, 2H, 2Hi, 2P, and 2Pi systems, where the subscript i indicates that the database has information about the indexing attributes of the information sources to be recommended. Updating will not necessarily be easy in systems like this.

These first Type 2 systems function essentially like Type 1R, 1H, and 1Pi systems that produce an intermediate answer in the form of one or more information sources (realistic weights now make sense). The system then goes to the database comprising each information source and matches the question attributes against that database in order to determine the final answer. So, two selection operations (or matches) are used: the first to determine a set of information sources and the second to match the question attributes against the databases comprising these sources to obtain the final answer.

Suppose we have a Type 2R system. As an example of its operation, consider a transaction in which a user wants to find biographical information on Linus Pauling, a chemist. That is, the question attributes are: question-type = biographical; personal-name = Pauling, Linus; geographical-area = U.S.A.; and subject = chemistry. Suppose the rule base contains a rule saying:

IF the question-type is biographical, and the geographical-area is U.S.A.,
THEN Who's Who in America may be useful.

Clearly, the question attributes will match this rule, and Who's Who in America will be among the tools recommended. Suppose the system has access to this tool as an online database (for example, in CD-ROM form). Then the question attributes will now be matched against the
database; effectively, this means that the personal name will be matched against the database. If a match occurs, then it will be a final answer. The same secondary matching will be carried out with any other recommended tools.

As with Type 1R systems, the determination of the question attributes may be either by backward- or forward-chaining on the first matching operation. Fine tuning of the attributes may be done if the results of the second matching operation are unsatisfactory.

In Type 2H, 2Hsi, 2P, and 2Ps systems, the internal operation will differ from that of Type 2R systems. But all five cases share the following characteristics:

- the question attributes are somehow determined from the user;
- they are matched against either the IF clauses of rules in a rule base, or the descriptors in a page-based database or a hypertext database;
- the answer is in the THEN clause of the rules, or the answer field in the database;
- a given answer (title of an information source) may appear in more than one place in the rule base or database, hence allowing for realistic weights, but also allowing problems in updating the answer; and
the question attributes are then matched against the final database (the information source itself) to obtain the final answer.

It should be noted that Type 2R systems may or may not allow precision or indexing attributes, depending on how the rule base has been designed. Hence a Type 2R system may have the system features of any of the four hypertext or page-based systems mentioned above.

Systems that know about indexing attributes (Types 2H_{si} and 2P_{si}) will have an additional match. In our example, the record also has \textit{indexed-field} = personal-name. After the first match that determines the info-source-title (the intermediate answer) for the record, there is a second match that is \textit{not} a selection operation: it merely verifies that one of the indexed fields in the selected tool corresponds to one of the question attributes for which a value is known. Here there is a match, since the personal-name field in \textit{Who's Who in America} is an indexed field.

Special care needs to be taken in the construction of a 2P_{si} system. Since every page can have only one set of indexing attributes attached to it, all sources listed on a given page must have the same set of indexing attributes. This problem does not arise with 2H_{si} systems, since we have precise labeling of information on hypertext pages, and can therefore assign individual indexing attributes to each label on a hypertext page.

4.2 Type 2 Systems with Easy Updating

As in Section 4.1, when we speak about easy updating, we intend the behavior of the first subsystem, that involved in the selection of an information source. As noted in Section 2.2, easy updating of answers is possible in components such as a single-answer-based database (S) or an item-based database (I), but they do not allow realistic weights. There are four kinds of Type 2 systems that may be built from only one of these components. They may be called Type 2S_{vi}, 2S_{si}, 2I_{vi}, 2I_{si} systems. (The subscript i indicates that the subsystem has information about the indexing attributes of the information sources that it will recommend.)

These second Type 2 systems function essentially like Type 1S_{i} and 1I_{i} systems that produce an intermediate answer in the form of one or more information sources. The system then goes to the database comprising each information source and matches the question attributes against that database in order to determine the final answer. So two selection operations (or matches) are used: the first to determine a set of information sources and the second to match the question attributes against the databases comprising these sources to obtain the final answer.

Suppose we have a 2S_{i} system. As an example of its operation,
consider the same question as in Section 4.1, namely a request for biographical information on Linus Pauling, a U.S. chemist. Suppose further that the database of information sources contains a record including the following fields and values:

- question-type = biographical
- geographical-area = U.S.A.
- subject = chemistry
- info-source-title = "The following information source may be useful: Who's Who in America."

The question attributes gathered through the reference interview will match this record. Unfortunately, a Type 2S₃ system cannot link effectively with an information source in the form of an external electronic database, since the system is unable to tell which part of the info-source-title is actually the title and which is additional text. The system is a partial implementation of a Type 2 model since the second match must be left to the user of the system.

Suppose, instead, that we have a Type 2I₃ system. As an example of its operation, consider the same question as above. But suppose that
the info-source-title in the record has the value: *Who's Who in America* and that the record also has *indexed-field = personal-name*. After the first match, which determines the info-source-title (the intermediate answer) for the record, there is a second match that is not a selection operation; it merely verifies that one of the indexed fields in the selected tool corresponds to one of the question attributes for which a value is known. Here there is a match, since the personal-name field in *Who's Who in America* is an indexed field.

![Diagram](image)

Figure 7. A type 2RS or 2RI model with a backward-chaining rule base

Two points should be noted. First, the system has precision in identifying the information source (since no additional text is present); hence it is possible to have a *full implementation* in which the system links to an external database. Second, the system knows about the indexing attributes of the external information sources; hence the expert system retains control over which index to search. The recommendation will be to search the personal-name field of the latter tool. This is done in the final match.

The behavior of the two remaining types, 2S<sub>si</sub> and 2I<sub>si</sub>, may be deduced from the above descriptions of 2S<sub>s</sub> and 2I<sub>s</sub> systems. Although
all four types differ in questions of precision and indexing attributes, each of them allows easy updating but not realistic weights.

4.3 Type 2 Systems with Realistic Weights and Easy Updating

In Section 4.1, we considered Type 2 systems with realistic weights in the first subsystem but not easy updating; some had precision and indexing attributes and some did not. In Section 4.2, we considered Type 2 systems with easy updating in the first subsystem but not realistic weights; again, some had precision and indexing attributes, some did not. In both those sections, we looked at systems with the first subsystem built from one component. As in Section 3.3, it is possible to construct multicomponent subsystems that have both realistic weights and easy updating, but not subsystems that have neither of these features. Such a multicomponent subsystem matches question attributes against a rule base (or H database) but does not store the final answers in the rules (or records of the H databases). Instead, each rule (or record) points to one or more entries in an I or S database of final answers, which is consulted in constructing the display used to answer the question.

So, as in Section 3.3, the first component must be R or H in order to provide realistic weights; and because we need precision in our link to the next component, the R must be designed to allow that (the H always allows it). The second component must be S or I in order to allow easy updating; it may or may not have indexing attributes. Hence, we have the following possibilities: 2RS, 2HS, 2RSi, 2HSi, 2RI, 2HI, 2Ri, and 2Hi.

Suppose we have a 2RS system. As an example of its operation, consider the same kind of question as before, namely, a transaction in which a user wants to find biographical information on Linus Pauling. Suppose, further, that the rule base (the first component) contains the rule

IF the question-type is biographical, and the geographical-area is U.S.A.,
THEN go to record Y in the database of single answers for some information sources that may be useful.

The question attributes match this rule, and record Y in the S database gives an answer something like: "The following tool may be useful: *Who's Who in America.*" In this case, the system has realistic weights and easy updating but neither precision nor indexing attributes. Consequently, the system is limited to a partial implementation (it cannot perform an online link to an external database); furthermore, it cannot recommend which index of the database to search.

Suppose we have a 2HIi system. As an example of its operation, consider the same kind of question as before. Suppose further that the H component has a page indicating that for biographical information
covering the U.S.A., the user should press a "button" leading to record Y in an item-based database (with indexing attributes) of information sources. The question attributes match this button on the page, and record Y gives the answer: *Who's Who in America*. In addition, the question attributes are matched against the indexing attributes of record Y, and the system recommends using the personal-name index. In this case, the system has realistic weights and easy updating, as well as precision (since the second component is item-based) and indexing attributes. Consequently, a full implementation is possible. In addition, the system can retain some control over the final match on the external database, namely the decision of which index to search.

<table>
<thead>
<tr>
<th>SYSTEM TYPES</th>
<th>REALISTIC WEIGHTS</th>
<th>EASY UPDATING</th>
<th>PRECISION ON TOOLS</th>
<th>INDEXING ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Ss</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2Ssi</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>2ls</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>2lsi</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2R 2Ps</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2R 2Psi</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>2R 2Hs</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2R 2Hsi</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2RS 2HS</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>2RSi 2HSi</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>2Ri 2Hi</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>2Rli 2Hli</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 8. Comparison of Type 2 models

4.4 Comparison of Type 2 Models

We are now in a position to compare the various Type 2 models in terms of the four system features: (1) the assignment of realistic weights, (2) easy updating, (3) precision in identifying information sources (and the consequent possibility of full implementations), and (4) knowledge about indexing attributes.
Of the sixteen possible combinations of these four features, only twelve are feasible, since in Section 2.2 we eliminated the possibility of a system with neither realistic weight nor easy updating. Even if it were possible to construct systems like that, it would be hard to justify doing so, since these systems would lack the two most desirable features of the four.

Each Type 2 model has been assigned to one of these twelve categories. It should be noted that the presence of a Y means that the feature is allowed, not that the feature is required; the presence of an N means that the feature is not allowed. For example, those models in Figure 8 with Y for realistic weights certainly allow the system to have realistic weights, but the system designer is not required to implement this by actually setting up the records or rules so that they have weights attached. Those models with N for realistic weights cannot have realistic weights at all.

4.5 Implementations of Type 2 Models

Because the literature describing implementations does not always give details of system design, it has been difficult to classify some implementations. For example, some systems classified as using I databases may actually use S databases or even P databases. And even the use of a hypertext design tool does not necessarily guarantee that the resulting system uses a hypertext database as we have defined it.

Finally, it should be noted that most implementations, with the exception of item 6 below, appear to be partial implementations, that is, they do not have direct access to external electronic databases.

1. An early system, REFSEARCH, was constructed by a group of researchers (included among them was Howard White, now at Drexel) at the University of California, Berkeley (Meredith, 1971). The system has detailed classifications and scopes for the database of reference tools and is a Type 2I, system.

2. Both REFLES (Bivins & Palmer, 1980) and REFLINK (Bivins & Eriksson, 1982), give, among other things, brief instructions for handling unusual searches (e.g., patents), evidently mentioning the information sources for searching patents. Both systems have subject access to the page-based database. REFLINK also has a hierarchical set of menus. So, both REFLES and REFLINK are of Type 2P, (as well as Type 1Px).

3. The Reference and Information Station (Purdue University Undergraduate Library), which was mentioned under Type 1P, models, gives menu access to pages on reference tools that might help in preliminary work in a subject area (Smith & Hutton, 1984; Smith, D., 1989). Hence, this system is of Type 2P, (and of Type
1P₃). It also features an electronic suggestion box as well as a statistical subroutine for collecting data on use of the system.

4. The Online Reference System (ORS) works by annotating selected records in the automated circulation system (Chisman & Treat, 1984). It allows direct subject access and menu access by type of reference work (and specific class assignment, too) to annotated records in the automated circulation system. Hence, this item-based system is of Type 2I₃.

5. The Information Function (IF) at Carnegie-Mellon University (Diskin & Michalak, 1985) provides (within the online catalog) menu access to online versions of library publications. This page-based system is of Type 2P₃ (and Type 1P₃).

6. The National Agricultural Library (NAL), Beltsville, Maryland, developed a small "demonstration" expert system called ANSWERMAN (Waters, 1986) to help library clients find answers to ready-reference questions. It uses a series of menus to narrow down the subject of the question and the type of tool needed (directory, encyclopedia, atlas, etc.). A set of choices from these menus activates a rule that points to a record in a bibliographic database giving a brief bibliographic description, call number, and, occasionally, an exact page reference. We shall consider other features of ANSWERMAN later in Sections 6.1 and 6.6. Using the same expert system shell, NAL has also developed AquaRef, an expert system for a specialized field, aquaculture (Hanfman, 1989). These item-based systems are both of Type 2RI.

7. POINTER is a system developed by Karen F. Smith at the Library of the State University of New York, Buffalo, for aiding library clients in locating U.S. federal government publications (Smith, K. F., 1986, 1989). POINTER points to reference tools that will help the user find both specific publications and publications on a particular subject. It uses menus to narrow down the type of question being asked. This page-based, menu-driven system is of Type 2P₃.

8. ORA (Parrott, 1986), developed at the University of Waterloo, allows menu and subject access to pages listing information sources. Hence, this system is of Type 2P₃ (and Type 1P₃).

9. PLEXUS is a system developed at the Central Information Service, University of London, as a referral tool for use in public libraries (Vickery & Brooks, 1987; Vickery et al., 1987). It is an ambitious creation including knowledge about the reference process, information retrieval, certain subject areas, reference sources, and library users. The system uses rules, frames, and semantic networks. It employs user modeling and a sophisticated blend of natural language processing, frames, and semantic networks for handling the reference interview for subject queries. Although the subject
domain is limited to gardening for the prototype phase, it is intended to be broadened in the second phase of development.

PLEXUS uses a database of information sources of four types: publications, organizations, databases, and experts. Hence PLEXUS is an item-based system which begins with an elaborate system for determining the question attributes. Next, rules are used for transforming the question attributes into a concept map, which is then matched against the database of information sources. These rules correspond to various search formulation tactics and term tactics articulated by Marcia Bates (1979). Since we are actually doing a match between the question attributes (in concept-map form) and our item-based database of information sources, PLEXUS must be a Type 2I system. Incidentally, these types of Bates tactics are also used to modify the concept map if the search misfunctions in some way, e.g., too many or too few hits. Although PLEXUS does not appear to use the Bates WEIGH tactic, it does use user modeling (see below under Section 6.5). In conclusion, PLEXUS is of Type 2I, with several Bates variants.

10. The Information Machine (Fadell & Myers, 1989) is a page-based, menu-driven system that includes pages listing specific information sources. Hence the system is of Type 2P, (and 1P).

11. The Technical Writing Assistant uses a natural language expert system to determine the question attributes, which are then matched against a database of information sources (Butkovitch et al., 1989). This item-based system is of Type 2I.

12. A prototype system developed by Trautman and von Flittner (1989) uses a database of online databases classified by nine attributes. It has several submodules that, among other things, determine the viewpoint (subject), construct a user model, transform the question attributes to a Boolean search, and rank the output. This item-based system is of Type 2I.

13. The Apple Library Tour (Ertel & Oros, 1989) uses a hypertext database mainly to provide directional information. It appears, however, to include hypertext pages referring to information sources as well; if that is the case, then it is of Type 2H (as well as 1H).

14. Paul Carnahan (1989) shows how to construct a hypertext system that uses Boolean searching of keywords to find reference tools. The system allows the search to be limited further by material type. The search card is essentially an interface program that carries out the reference interview and subsequent match against the database stack (containing information on the various reference tools). The database stack seems to consist of hypertext pages each of which is restricted to one tool only. Hence the possibility of realistic weights cannot be implemented; on the other hand, easy updating is possible.
Thus, the design of this hypertext database forces it to behave like an item-based or single-answer-based database. Although superficially the system seems to be of Type 2H, it is probably more correctly classified as Type 2I or 2S.

5. SYSTEMS WITH THREE SELECTION OPERATIONS

Why bother going beyond Type 2 systems? Type 2 systems allow us to model the fact that librarians use specific strategy (prescribing the use of specific information sources). But, reference librarians sometimes also use general strategy (prescribing the use of categories of information sources); this is acknowledged in another model of Katz (discussed below), and is the basis for Type 3 systems. General strategy, like specific strategy, forms an intermediate answer, and therefore may not always be part of the explicit prescription to the user. But, even if not explicitly stated, general strategy has these advantages:

1. It serves to eliminate from consideration those categories of tools that it does not recommend. This is useful, since many tools may match the usual scope attributes (subject area, geographical area, etc.) but may actually be of very little use in answering the type of question being considered. This is a practical advantage that may be of use in any implementation.

2. It represents a classification of our specific strategies, and therefore allows us to organize our reference knowledge better. This may be useful to the people formulating the reference knowledge; it does not help the user directly.

3. Some inference engines allow explanations (a kind of limited instructional feature). Including knowledge about general strategy allows explanations of explanations of specific strategy by indicating that a specific strategy is an instance of a particular general strategy.

4. Intelligent CAI systems (Intelligent Tutoring Systems), are a more comprehensive instructional approach. Including knowledge about general strategy in such systems allows them to teach it. In fact, an ICAI system virtually requires the teaching of general strategy, since people find it easier to learn specific strategy if it is presented as a consequence of general strategy (Clancey & Letsinger, 1981).

Type 3 models are derived from Katz's Translation Device Model (1982, pp. 76-81). The basic idea here is that:

- after data gathering, a useful type (or types) of information source is determined;
• the latter data plus the data gathered are then used to determine a particular information source (or sources) that may contain the desired information; and
• that source (or sources) is then consulted.

As with the Type 2 models, the Type 3 models apply to directional, ready-reference, and substantive transactions.

Type 2 systems had only one match whose design we could control; hence we needed to consider all combinations of only four features for combinatorial completeness. Type 3 systems, however, have two matches whose design may be controlled. We therefore must consider all combinations of eight features for combinatorial completeness, for a total of 256 possible combinations. But we can reduce this number considerably by reasoning. The eight features are:

1. realistic weighting (first component)
2. easy updating (first component)
3. precision (first component)
4. indexing attributes (first component)
5. realistic weighting (second component)
6. easy updating (second component)
7. precision (second component)
8. indexing attributes (second component)

In the following, we shall examine five of the above eight features and show that the values of none of them may usefully be varied. The arguments will demonstrate either that a given feature must always have a particular value (e.g., positive), or that a given feature is of no interest.

The third feature, precision of labeling in the first component, must always have a positive value. The situation here is that we have come to a record or rule that recommends a certain type or types of information sources. But if the answer field does not label the type or types precisely, that is, if it precedes or follows the type with additional data, then it cannot pass the types on to the component of the system in which particular information sources are determined. (A human could, of course, parse this information out, but we assume that the system cannot: that is, it considers the answer field simply a meaningless jumble of characters.) Hence, in a Type 3 system, precision of labeling in the first component must always have a positive value.

The fourth feature, indexing attributes in the first component, is unnecessary. Indexing attributes are important so that we can determine how a particular tool is to be used. Although we could include a default value of this feature for a class of tools (for example, “A trade directory generally has an index by manufacturer name”), there is always the
possibility that that default may be overruled by the indexing attributes for a particular tool in that class. Furthermore, if we do want indexing attributes, then we must certainly include specific indexing attributes, exactly because we cannot count on a tool conforming to type. So, if the only purpose of this default is to specify the indexing attributes of particular tools, it is superfluous. But what other purpose could it possibly serve? We therefore overlook indexing attributes for the first component.

The fifth feature, realistic weights for the second component, must always be negative. The argument here is rather more elaborate. First, by definition, in a Type 3 system: The type of information source must be determined first, as a necessary preliminary to determining second the particular information sources. Hence, the second match must be not only on the question attributes, but also on the type of information source. This definition implies that, even if a model deduced a particular information source using two matches like this, it would not be a true Type 3 model if one could find another model that would do that in a single match.

Suppose, for the moment, that a Type 3 system could have realistic weights for the second component; that is, suppose that realistic weights of the second component could be positive. (On this assumption, we proceed to demonstrate a contradiction.) Let the first component be equivalent to any first component in a Type 2 system. But, let the second component be equivalent to only a Type 2 first component with realistic weights (or realistic weights and easy updating).

Given the question attributes, the match in the first component determines the type of information source. The question attributes and type are then matched in the second component to get a particular source. Now, as mentioned above, the generally applicable practical advantage to calculating the type of information source is that it acts to eliminate possibilities in the second match. For example, for a biographical question, if we determined that appropriate types of tools include only biographical dictionaries, general encyclopedias, etc., then the second match will exclude all tools that do not satisfy these types, even though they satisfy all the scope attributes, like geographical area.

But if the second component has realistic weights, as assumed, then it also has eliminative capabilities, as established in Section 2.2. Hence, we can use the eliminative capabilities of the second component to accomplish what the calculation of type did; hence type is unnecessary. This contradicts the definition of Type 3. Therefore, a true Type 3 system cannot have realistic weights for the second component.

The sixth feature, easy updating for the second component, must always be positive. This follows from (1) the previous result, that realistic weights (second component) must always be negative, and (2) we cannot
have both realistic weights and easy updating negative in the same component (Section 2.2).

Let us now return to the first feature, realistic weighting for the first component. We shall now argue that it should always be positive. Suppose that it were negative; that is, suppose that the first component did not have realistic weighting. Then the first component would be either an I or S database. But we concluded above that precision must be turned on in the first match, so the first component would be an I database. Now, we deduced above that a true Type 3 system cannot have realistic weights for the second component. Thus, the second component would be an I or S database.

So, if realistic weights are turned off, then the system matches the question attributes against an I to get a type of information source. Then it matches the question attributes plus the type of information source against an I or S database to get a particular information source. But recall that in Section 3.3 we argued that a two-component system in which both components allow easy updating but not realistic weights is equivalent to a one-component system that allows easy updating but not realistic weights. So, on our assumption that realistic weighting for the first component is turned off, our system collapses to a Type 2 system. Thus realistic weighting must be positive in the first match in a true Type 3 system.

Finally, there are only three features that can be varied combinatorially (for a total of eight possible combinations):

1. Easy updating for first component
2. Precision for the second component
3. Indexing attributes for the second component

Of the remaining five features:
4. Realistic weights for first component must always be Y
5. Precision for first component must always be Y
6. Indexing attributes for first components are irrelevant
7. Realistic weights for second component must always be N
8. Easy updating for second component must always be Y

5.1 Characteristics of Type 3 Systems

In the previous section, we established five constraints on Type 3 systems; of these, four involved fixing the values of features. We now enumerate some implications of some of those fixes:

- By (4) above, the first component must be one of: R, P, P, H, H, H, RS, HS, RS, HS, RI, HI, RI, or HI.
- By (5) above, we must have precision in the first component, so we are left with first components of: R, H, H, RI, HI, RI, or HI.
By (7) above, the second component must be one of: $S_s$, $S_{si}$, $I_s$, or $I_{si}$.

By (8) above, the second component must have easy updating, but that applies to all four possibilities just found.

So we are left with $7 \times 4 = 28$ possibilities to be distributed over eight categories. Rather than enumerating these possibilities and describing several models, we shall save the enumeration for the comparison chart in the next section. Here we shall simply describe the model (Type $3RI_{si}$) for which an implementation exists, namely REFSIM (Parrott, 1988, 1989).

Suppose we have a Type $3RI_{si}$ model. As an example of its operation, consider a transaction in which a user wants to find biographical information on chemist Linus Pauling. Suppose the rule base contains a rule saying:

** IF the question-type is biographical, 
** THEN biographical dictionaries may be a useful type of information source.

Clearly, the question attributes will match this rule, and biographical dictionaries will be among the types of tools recommended. Suppose further that the database of information sources contains a record including the following fields and values:

- type-of-info-source = biographical-dictionary
- geographical-area = U.S.A.
- subject = chemistry
- info-source-title = Who's Who in America
- indexed-field = personal-name.

The question attributes gathered through the reference interview, and the deduced type of information source, will match this record. Now, after this first database match, the indexing attributes of *Who's Who in America* are checked against the question attributes. Since the question attributes include a personal name, and the information source is indexed by personal name, the prescription will be to use its personal-name index. Note that if appropriate fields had not been indexed, it would have been necessary to use either the Bates STRETCH variant (Section 6.3) or the Bates SCAFFOLD variant (Section 6.4). The recommendation after the first database can indicate not only useful information sources, but also the techniques by which they should be searched for the given question.

Since this first database is an I database, the system will be able to send its information over to an external database (such as a CD-ROM system) for the final match. (Because REFSIM is a $3RI_{si}$ system, the latter feature is allowed in REFSIM; but it was not implemented.) And, since the first database knows about indexing attributes, the expert system retains control over which index to search in the external database.
If a match occurs, then it will be a final answer. The same secondary matching will be carried out with any other recommended tools found through other matches on the first database.

5.2 Comparison of Type 3 Models

By combining the enumeration considerations at the beginning of Section 5.1 with information from Figure 8 (comparing Type 2 models), we are able to construct Figure 10, which enumerates and compares Type 3 systems.

![Diagram]

Figure 9. A type $3RS_s$ or $3RI_s$ model with a backward-chaining rule base

6. VARIANTS ON THE BASIC MODELS

In Sections 6.1 to 6.4, we see how certain Bates (1979) search tactics introduce variants in some of the models examined above. The PATTERN tactic is deliberately excluded here, since it is so fundamental that it may be thought of as the basic form of many of the operations
that are being modified by other Bates tactics in the variants below. In Sections 6.5 and 6.6 we identify other variants.

6.1 Variant 1: WEIGH

With the Bates WEIGH tactic, a weight is assigned to each recommendation to indicate its effectiveness and efficiency in solving the problem. If we allow a system (as opposed to a person) these kinds of weights, we shall call it a WEIGH variant. Now, a glance at the diagrams for our models shows that several important operations may be involved in making any recommendation. Hence, in general, several operations in a model may contribute to the final calculated weight. We may consider a WEIGH variant to arise somehow from modifications (adding weights) to the fundamental operations in a given model.

<table>
<thead>
<tr>
<th>SYSTEM TYPES</th>
<th>EASY UPDATING (1st comp.)</th>
<th>PRECISION ON TOOLS (2nd comp.)</th>
<th>INDEXING ATTRIBUTES (2nd comp.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3RSs 3HsiSs</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3RSs 3HsiSsi</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>3Rls 3Hsis</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>3Rlsi 3Hsisi</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3Rls 3Hlils</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3Rlisi 3Hlii</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>3Rlils 3Hlii</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>3Rlli 3Hlli</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 10. Comparison of type 3 models

It should be noted that Type 2 and 3 models involve consulting sources that may or may not have the desired information; hence, some ranking by likelihood of speedy success would be useful. The WEIGH variant is therefore perfectly natural in these models. It is hard, however,
to make a case for using WEIGH variants in Type 1 models, since no intermediate sources are consulted and the final answers are given directly.

Let us now consider the different kinds of operations that have appeared in our Type 2 and 3 models, and how the WEIGH variant might affect them. Those operations include:

1. Commands to perform the reference interview. Weights can be added to the value of each question attribute gathered in the reference interview by asking the user to indicate the importance of each value supplied.

2. Commands to search a system component that allows realistic weights (a rule base or certain kinds of databases) to select an information source or a type of information source (Type 3 only): 2R, 2P, 2P_i, 2H_i, 2RS, 2HS_i, etc. or any Type 3 system. Modifications might be of two types:
   (a) Adding weights to the rules or records, to express the likelihood that an information source or type of information source will be useful for the given set of attributes.
   (b) Adding weights to the search commands to express the fact that if we match on a broader or narrower term (see SUPER, SUB, etc. below) than the user really wants, then the likelihood of finding the desired information in an information source that matches is different from what it might otherwise be.

3. Commands to search any other kind of database (without realistic weights) to select an information source: 2S, 2S_i, 2I, 2I_i, or any Type 3 system. Modifications might be of two types:
   (a) Here, different weights cannot be specified for different sets of question attributes. The best that can be done is to add one set of weights to each item in the database to express the degree of coverage for that source, given its stated scope.
   (b) Adding weights as in (2b).

4. Commands to consult a database of information sources for additional information (Types 2RS, 2HS, 2RS_i, 2RI, 2HI_i, etc.). No modifications to these commands would be reasonable.

Implementations:

- The ANSWERMAN system (Waters, 1986) of the National Agricultural Library, is a Type 21 rule-based system activated by menu choices, and has the capability of attaching weights to its recommendations. Hence, it may also be considered a WEIGH variant of the (2a) type mentioned above. Using the same expert system shell, NAL has also developed AquaRef, an expert system for a specialized field, aquaculture (Hanfman, 1989). All these Type 21 systems use weights of the (2a) variety above.
The prototype system REFSIM (Parrott, 1988, 1989) of Type 3I sub allows weights of the (2a) and (3a) varieties above.

A prototype system developed by Trautman and von Flittner (1989) implements weights of the (1) variety.

6.2 Variant 2: SUPER, SUB, and Other Term Tactics

Sometimes we want more information than we find using the selection operations as described earlier. In general, a system can retrieve additional information by either:

a. allowing matches on reference tools whose scopes are broader than those in the original question attributes, or
b. narrower (if we renegotiate the question), or
c. allowing matches on reference tools whose types are narrower than the type calculated (if we renegotiate the question).

Bates (1979) described term tactics, which help in part of this process. The term tactics move from one search term to a different one; for example, the SUPER tactic moves to a broader term, the SUB tactic moves to a narrower one. The processes in the previous paragraph can be effected by adding (i) term tactics just after any of the selection operations, with a control loop to retry the selection operation, and (ii) a semantic network of terms on which the term tactics operate. If we allow a system (as opposed to a person) to do this sort of thing, we shall call it a SUPER variant, etc.

We now consider the different types of selection operations and the effects that SUPER, SUB, and other term tactics might have on them:

1. Commands to search a rule base or database to select an information source or a type of information source (Type 3 only).

If our question attributes match too few (perhaps none) of the IF clauses of any of the rules in the rule base, or the descriptors of any of the records in the database, then we can use technique (a). A SUPER term tactic (operating on a semantic net) could broaden a particular attribute of the question, and then retry the match.

Consider a biographical question restricted to France, and suppose that there is a rule that says:

 If question type is biographical AND geographical scope is Europe, THEN use Z.

SUPER (operating on a semantic net) could broaden our geographical attribute to Europe and match the rule. And this rule will also be appropriate for a biographical question with geographical scope of France. Rules like this would retrieve additional sources, but these
Sources might be less effective than sources involving a direct match. Similar considerations hold for a database.

Alternatively, we can use technique (b). Suppose our rule says

IF question type is biographical AND geographical scope is Paris, THEN use W.

SUB (operating on a semantic net) could narrow our geographical question attribute to Paris and match the rule. But the question attributes would first have to be renegotiated to ensure that the user is interested in Paris. Similar considerations hold for a database.

Technique (c) arises only in Type 3 models, where our question attributes include the calculated type of information source. Notice that we cannot broaden the type and then rematch. For example, suppose that the system had first determined that an appropriate type of information source for a telephone number is a telephone directory. If we broadened telephone directory to directory, we might be referred to directories that systematically exclude telephone numbers. But we can narrow the type of information source, for example, to government telephone directory, by using the SUB term tactic. Here it would be necessary to renegotiate the question to see how the type of information source should be narrowed.

2. Commands to consult a database of information sources for additional information (Types 2RS, 2HS, 2RS₁, 2RI, 2HI, etc.). As with WEIGH, no modification to these commands is reasonable.

3. Commands to match the question attributes against a database of information within an information source. The same considerations apply as in cases (1a) and (1b), except that we would generally use a set of term tactics larger than SUPER or SUB. The question attributes need to be renegotiated not only for SUB, but for several other term tactics, including RELATE, NEIGHBOR, TRACE, and FIX.

Implementations

PLEXUS uses the (1a) variety of SUPER when a search statement is being modified because too few information sources were retrieved in the match against the database. This is done by replacing a term by its parent term in BSO, the semantic net used in PLEXUS. PLEXUS also has rules implementing some of the Bates search formulation tactics. For example, after determining the question attributes, PLEXUS uses rules for transforming the question attributes into a concept map, which is to be matched against the database of information sources. A prototype system developed by Trautman and von Flittner (1989) also has some rules like the latter.

REFSIM uses the (1a) variety of SUPER on the REFSIM rule base for choosing a class of information sources if no matches are found.
REFSIM also implements the (1a) variety of SUPER on the database for choosing a specific information source, again, if no matches are found on the given terms.

6.3 Variant 3: STRETCH

With the STRETCH tactic we use an information source for a purpose for which it was not intended. Hence we must first be able to work effectively with single sources, so we must have precision on sources. And we must second have access to information about the intended uses of sources, so we must have all the information about an information source in one place. We must therefore limit ourselves to the following models: 2I_s, 2I_{si}, 2RI, 2HI, 2RI_{ji}, 2HI_{ji}, and all Type 3 systems with precision on tools in the second component.

Source attributes express intended use. But it is too extreme to allow the ordinary question attributes (subject, etc.) to fail to match the ordinary source attributes. An alternative is to consider failure to match unusual values like the type of information source or the indexing attributes. Various cases are examined below.

1. Match on question attributes, then try but fail on type of information source. This means we must have a Type 3 model. The STRETCH tactic will involve a rule of the form:

   IF we have a proper match between the ordinary question attributes and the scopes of the information sources, AND there is NOT a match on the type of information source, THEN try the resulting information sources anyway.

2. Match on question attributes, then try but fail on indexing. Since our model must allow matching on indexing attributes, it must be of Type 2I_{si}, 2RI_{ji}, 2HI_{ji}, or of any Type 3 with precision on tools and indexing attributes in the second component. The STRETCH tactic will involve a rule of the form:

   IF we have found an information source matching the ordinary question attributes, AND IF none of the source fields for which we have input values are indexed in the source, THEN use that information source AND browse over all the data in the information source.

3. Match on question attributes, then try but fail on either indexing or types of information sources. This will require a Type 3 model with precision on tools and indexing attributes in the second component, and will involve broadening the search in the manner of both the (1) and (2) varieties.

   REFSIM (Parrott, 1988, 1989) implements the (2) variety of STRETCH variant discussed above.
6.4 Variant 4: SCAFFOLD

The essence of the SCAFFOLD tactic is that we construct an indirect pathway passing through more than one information source in order to reach an information source that will contain the desired information. Hence, we must first be able to work effectively with single sources; we must have precision on sources. And we must second have ready access to all the information about each source, in order to make sure that sources in a pathway have consistent scopes; so we must have all the information about an information source in one place. We must therefore limit ourselves to item-based models. But, to construct the pathway, we must know which fields in our sources are indexed; so we must also have indexing attributes. Hence, as with variety (2) of STRETCH, our model must be Type 21si, 2RIi, 2HIi, or of any Type 3 with precision on tools and indexing attributes in the second component. (Note: A SCAFFOLD variant temporarily forces a 3I system to behave like a 2I system, since it circumvents the command to determine the type of information sources, and operates only on individual sources.) There are at least three types of SCAFFOLDS:

1. A particular tool contains the type of information desired, but is not indexed so that it can accept any of the question attributes as input. A SCAFFOLD tactic here would:
   - assume the final tool and
   - construct the pathway in reverse order so that proper output/input links hold, until we
   - reach a tool that can serve as an initial tool.
   REFSIM implements a two-source version of this type of SCAFFOLD.

2. A particular tool is indexed so that it can accept at least one of the question attributes as input, but it does not contain the type of information desired. A SCAFFOLD tactic here would assume the initial tool, then construct the pathway in forward order so that proper output/input links hold, until we reach a tool that can serve as a final tool. This type of SCAFFOLD is the reverse of the first one.

3. A particular tool contains the type of information desired, and is indexed so that it can accept at least one of the question attributes as input, but the input value does not give a unique output (as with "Smith" for a large author index). A SCAFFOLD tactic here would:
   - go to a source that has fewer entries (e.g., one narrower in scope) and has an index for our initial input value;
   - perform a Boolean AND match on the fragmentary input value and other values to be sure we get the correct match (for example, we might try to find all Smiths working in Biochemistry at the University of Leeds); and
• take this more precise value to the final tool.

In this type of SCAFFOLD, unlike the others, we need to be able to do a Boolean-AND match with truncation on the fragmentary value. We can recast these additional requirements as additional Bates tactics. Requiring a Boolean AND here is equivalent to the Bates search formulation tactic, EXHAUST, in which a search is rendered more precise by ANDing all relevant concepts. Requiring truncation in the manner described is equivalent to her term tactic NEIGHBOR, in which we seek additional terms by looking at neighboring terms (in the example given, looking at all terms beginning "Smith"). So, this third SCAFFOLD brings in two more Bates tactics.

Essentially, the SCAFFOLD involves finding ways around the artificial boundaries imposed by the publication process. Insofar as we succeed, we temporarily create a meta-source or imaginary source that links together the information found in several sources, in order to create the effect of a more powerful source.

6.5 User Modeling

Type 1, 2, and 3 models all include the system's model of the user attributes. This suggests that user modeling is a commonplace feature of reference expert systems. But few reference expert systems have actually implemented it. At the present time, therefore, it is probably better to consider user modeling an optional feature of these various types of models.

Implementations

PLEXUS is one of few systems implementing user modeling. A series of menus is used to determine characteristics of the user. This information is later used, for example, to determine how much explanation of certain tools to give, or to decide how much effort to devote to finding material. REFSIM also provides some support for user modeling.

A prototype system developed by Trautman and von Flittner (1989) also implements user modeling.

6.6 Access to Actual Information Sources

For Type 1 models, the desired information resides inside the expert system. The basic structures of our Type 2 and 3 models, however, explicitly include access to information sources that might contain the desired information. Few current reference expert systems implement this kind of access through an electronic interface; instead, they generally
stop at prescribing tools, and leave the consultation of the tools and the final matching to the user. Type 2 or 3 systems lacking this kind of electronic access may be called *partial implementations* of Type 2 or 3 models. It may be noted that a partial implementation of a Type 2 model will have the same basic structure as a Type 1 model, but may have features not necessary or possible in a Type 1 model, such as realistic weights and indexing attributes.

**Implementations**

The ANSWERMAN system of the National Agricultural Library, mentioned in Section 4.5, is a rule-based system activated by menu choices, and has the capability of functioning as either a consultation system or as a front end to external online databases and CD-ROM reference tools. Using the same expert system shell, NAL has also developed AquaRef, an expert system for a specialized field, aquaculture (Hanfman, 1989). These systems may be the only current reference expert systems that allow this capability. It is safe to predict, however, that this kind of capability will grow considerably in the future.

**7. EXTENSIONS TO REFERENCE KNOWLEDGE BASES**

**7.1 Developing Intelligent Tutoring Systems for Reference**

How can a knowledge base on reference practice and theory be used either to instruct library clients or to train reference librarians? Just as expert systems may be used to simulate the professional in the consultational process between client and professional, computer-assisted instruction (CAI) systems may be used to simulate the teacher in the instructional process between teacher and student. But CAI systems are inflexible and inefficient to construct. The situation has been improved with the development of Intelligent Tutoring Systems (ITS), also known as Intelligent Computer-Assisted Instruction (ICAI) systems (Dede, 1986; Peachey & McCalla, 1986). Unlike a CAI system, an ITS typically uses a knowledge base for its subject expertise (as does an expert system) and an additional knowledge base for its teaching expertise.

The subject-expertise knowledge base for a reference ITS is the same as that for an expert system for giving reference advice. So, a single knowledge base could drive both. Since people (unlike machines) find it easier to remember and apply a rule presented as a logical consequence of a strategy, an ITS knowledge base should include heuristic rules giving overall strategy, not just specific strategy (Clancey, 1979). Hence a reference ITS knowledge base will need, for example,
rules pointing to classes of information sources, since such rules represent
general strategy in choosing information sources. Therefore, reference
ITS applications must be Type 3 systems.

Implementations
A prototype ITS system for reference, REFSIM, has been described
in some detail in the literature (Parrott, 1988, 1989). A special feature
of REFSIM is the simulation of live reference transactions to teach the
reference interview and the rationale behind search strategy prescription. 
REFSIM is a partial implementation of a Type 3I model. That is, it
does not allow access to external electronic information sources.

<table>
<thead>
<tr>
<th></th>
<th>SURFACE KNOWLEDGE</th>
<th>DEEP KNOWLEDGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN</td>
<td>usually useful</td>
<td>surface knowledge fails</td>
</tr>
<tr>
<td>ARTICULATION BY EXERTS</td>
<td>easy</td>
<td>usually difficult</td>
</tr>
<tr>
<td>DEDUCTIONS</td>
<td>efficient</td>
<td>generally inefficient</td>
</tr>
<tr>
<td>DEEP EXPLANATIONS</td>
<td>not possible; can only quote surface knowledge involved</td>
<td>explanations of explanations are possible, by quoting deep knowledge underlying surface knowledge</td>
</tr>
<tr>
<td>DEEP ERROR DETECTION</td>
<td>not possible; can detect only errors in understanding surface knowledge</td>
<td>some subtle cognitive errors can be detected, by testing deep knowledge understanding</td>
</tr>
<tr>
<td>CREATION OF NEW SURFACE KNOWLEDGE</td>
<td>not possible</td>
<td>possible, but generally time-consuming computations are required</td>
</tr>
</tbody>
</table>

Figure 11. Surface vs deep level knowledge

7.2 Articulation of Deep Reference Knowledge
Deep Knowledge in a subject domain may be thought of as the
first principles learned from school or books. This is often the first
type of knowledge to be acquired in the professional domain. An expert
usually relies on a different type of knowledge, called surface knowledge or heuristic knowledge, which consists of rules of thumb or short cuts
learned from the expert’s experience or from experience passed on by mentors (Harmon & King, 1985). A novice normally does not have access to this kind of knowledge.

Deep reference knowledge must correspond to some kind of first principles underlying reference practice. A natural assumption is that some subset of information science underlies reference practice. It is not a great step from the above assumption to the following hypothesis:

Surface reference knowledge tends to be concerned with the sources of information, but deeper levels of reference knowledge tend to be more concerned with the information itself and the people associated with it.

This approach allows us to establish solid logical links between knowledge in information science and in library science.

---

![Diagram](image-url)

**Figure 12. Upper deep structure for a rule**

Most of the deepest knowledge deals directly with people and human motivation, and consequently often involves certain *modal* concepts, that is, concepts concerned with what CAN BE, SHOULD BE, IS DESIRED TO BE, WILL BE, etc. This kind of knowledge cannot be expressed well using classical logic, which is concerned rather with
what IS. Modal logic, an extension of classical logic, is required to represent knowledge like this and to establish the validity of deductions based on this knowledge.

To clarify matters, we shall look at some examples of deep rules from which the following surface rule on biographical information can be deduced:

IF the type of ready-reference question is biography, the person is dead, and the occupation was academic, THEN consult indexes of academic journals for obituaries.

Some of the upper-level deep knowledge from which this surface rule can be deduced is shown in Figure 12. This rule can be derived from about twenty deep knowledge statements (including those in Figure 12).

8. CONCLUSION: FUTURE PROSPECTS

Many current reference expert systems do not implement some important features of the reference models considered. Of these, user modeling is probably the most critical, and is therefore a promising area for future development. Many current systems might also be improved through the implementation of the Bates WEIGH tactic (e.g., using confidence factors) and the provision of interfaces to external databases. If the last-mentioned facility becomes commonly implemented in reference expert systems over the next few years, then it is only a matter of time before reference expert systems merge with information-retrieval expert systems to form sophisticated front-end systems that can guide a user from one electronic tool to another and give assistance in searching each one of them.

But there is one caveat that must be added. It is widely believed (Walters & Nielsen, 1988) that expert systems in general (including the sort considered in this chapter, as well as those postulated in the last paragraph) have no real future unless the question of the “brittleness” of current expert systems is addressed. Current expert systems are considered to be brittle rather than “robust” since, as they move outside of their areas of expertise, there is a drastic drop in their ability to handle the situation, rather than a graceful degradation. Some researchers believe (Walters & Nielsen, 1988) that providing a knowledge base with deep structure, although a time-consuming process, is a good way of overcoming these limitations of current expert systems.

What is the situation for reference expert systems? The surface structure of reference heuristics and information retrieval heuristics is being well explored in current systems, and the proposed rules seem reasonably consistent with one another. The first principles of
information science (which it is reasonable to assume underlie the previously mentioned surface-level heuristics) have been rather less well explored. But the relationship between these two types of knowledge has scarcely been examined at all. This will have to be remedied if we are to make significant progress in creating more intelligent systems. It is this author's conviction that this will indeed happen, and that, moreover, the mapping of these logical links will eventually become as important to the library and information sciences as the mapping of the human genome has become to the medical and biological sciences.

REFERENCES


