RUBRIC: A System for Rule-Based Information Retrieval

BRIAN P. MC CUNE, MEMBER, IEEE, RICHARD M. TONG, MEMBER, IEEE, JEFFREY S. DEAN, MEMBER, IEEE, AND DANIEL G. SHAPIRO

Abstract—A research prototype software system for conceptual information retrieval has been developed. The goal of the system, called RUBRIC, is to provide more automated and relevant access to unformatted textual databases. The approach is to use production rules from artificial intelligence to define a hierarchy of retrieval subtopics, with fuzzy context expressions and specific word phrases at the bottom. RUBRIC allows the definition of detailed queries starting at a conceptual level, partial matching of a query and a document, selection of only the highest ranked documents for presentation to the user, and detailed explanation of how and why a particular document was selected. Initial experiments indicate that a RUBRIC rule set better matches human retrieval judgment than a standard Boolean keyword expression, given equal amounts of effort in defining each. The techniques presented may be useful in stand-alone retrieval systems, front-ends to existing information retrieval systems, or real-time document filtering and routing.

Index Terms—Artificial intelligence, evidential reasoning, expert systems, information retrieval.

I. The Information Retrieval Problem

THE three most common approaches to the textual information retrieval (see the vertices of the triangle in Fig. 1), when used in isolation, suffer from problems of precision and recall, understandability, and scope of applicability. For example, Boolean keyword retrieval systems such as the commercial DIALOG system operate at a lexical level, and hence ignore much of the available information that is syntactic, semantic, pragmatic (subject-matter specific), or contextual. The underlying reasoning behind the responses of statistical retrieval systems [2] is difficult to explain to a user in an understandable and intuitive way. Systems that rely on a semantic understanding of the natural language that is present in documents [3] must severely restrict the vocabulary and document styles allowed (e.g., to partially formatted, stereotypic messages).

In addition to being used by specialists, in the near future large on-line document repositories will be made available via computer networks to relatively naive computer users. For both classes of users, it is important that future retrieval systems possess the following attributes.

- Detailed queries should be posed at the user's own conceptual level, using his or her vocabulary of concepts and without requiring complex programming.
- Partial matching of queries and documents should be provided, in order to mirror the imprecision of human interests.
- The number of documents retrieved should be dependent upon the needs of the user (e.g., uses for the documents, time constraints on reading them).

Manuscript received November 7, 1983.

The authors are with Advanced Information & Decision Systems, Mountain View, CA 94040.

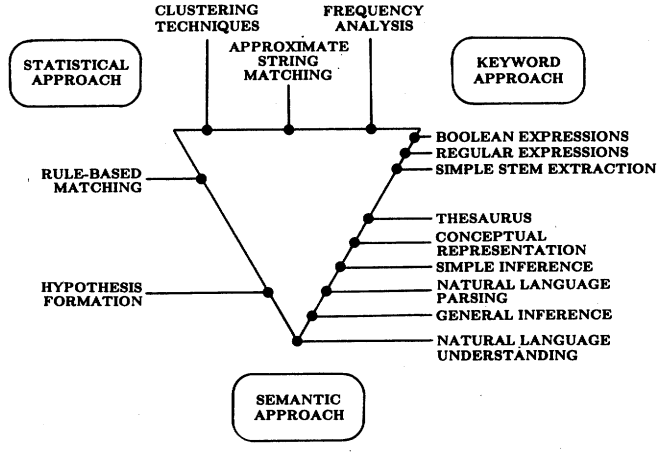


Fig. 1. The information retrieval triangle.

- A logical, understandable, and intuitive explanation of why each document was retrieved should be available.
- The user should be able to easily experiment with and revise the conceptual queries, in order to handle changing interests or disagreement with previous system performance.
- Conceptual queries should be easily stored for periodic use by their author and for sharing with other users.

II. A Knowledge-Based Approach

In order to address the issues raised above, we have created a prototype knowledge-based full-text information retrieval system called RUBRIC (for RUle-Based Retrieval of Information by Computer). RUBRIC integrates some of the best characteristics of all three basic approaches to information retrieval (Fig. 1) within the framework of a standard artificial intelligence technique. Queries are represented as a set of logical production rules that enable the user to define retrieval criteria using much better semantic and heuristic controls than can be found in current retrieval systems.

The rules define a hierarchy of retrieval topics (or concepts) and subtopics. By naming a single topic, the user automatically invokes a goal-oriented search of the tree defined by all of the subtopics that are used to define that topic. The lowest level subtopics are defined in terms of pattern expressions in a *text reference language*, which allows keywords, positional contexts, and simple syntactic and semantic notions. Each rule may have a user-provided heuristic weight. This weight defines how strongly the user believes that the rule's pattern indicates the presence of the rule's subtopic. Technical issues that arise when information retrieval is viewed as a problem in evidentiary reasoning are discussed in [6].

0098-5589/85/0900-0939$01.00

To perform a retrieval RUBRIC uses the set of rules for a topic to create a heuristic AND/OR goal tree that defines at its leaves what patterns of words should be present in documents, and in what combinations.

Document recall by RUBRIC is enhanced by the use of higher level notions than simple Boolean combinations of keywords. Retrieval precision is improved by the use of variable weights on each rule to define the certainty of match. These weights make it possible to present to the user only partial matches above some threshold. By tracing through rule invocation chains, an explanation facility allows the user to see exactly why a document was retrieved and why it was assigned its overall certainty or importance weight. This promotes experimentation and appropriate modification of the rule base. The retrieval vocabulary to be used is unrestricted, being left up to whoever creates the rules. Rule sets may be stored in public or private rule "libraries," so that useful subtopics may be shared among users, thus simplifying the task of defining new topics.

A rule-based query can be more complex than the keyword expression that might be used with a Boolean retrieval system. Therefore, we expect rule-based retrieval to be used initially for applications in which the same query is made repetitively over some period of time. In such situations people who are trained RUBRIC users but not programmers should be willing to expend more effort to develop a detailed rule-based definition of the query topic.

Although RUBRIC is a knowledge-based system, it really is not an expert system in the usual sense. In an expert system the system's knowledge base is an attempt to define what is known about some field of inquiry (e.g., infectious diseases, geology) in a useful form analogous to that used by human experts. Although the knowledge is never complete and perhaps not agreed upon by all experts, there exists some underlying theory or physical model that all concerned believe. In the case of information retrieval, as in other areas of preference such as politics or matters of style, there is no "right" answer. Hence, RUBRIC is really a system for capturing and evaluating human preferences. Preference systems are likely to play a much larger role in the future, as artificial intelligence tackles the problem of supporting complex, multiattribute decision making.

III. Expressing Query Topics as Production Rules

RUBRIC gains its power from the knowledge base of retrieval rules at its disposal. An example set of rules that defines the topic of the 1982 World Series of Baseball is given in Fig. 2. These 15 rules define a main topic, called World_Series, and a number of subtopics. The subtopics are used to define the main topic, but may also be used as query topics on their own or as subtopics of other main topics. This rule set is by no means complete; however, extensions in the form of additional rules are easy to make.

Each rule defines a logical implication; that is, the existence of the pattern on the left-hand side of the arrow ("=>") implies the existence of the topic named on the righthand side. Thus, a rule defines the topic or concept named in its right-hand side. There may be multiple rules about the same topic, and RUBRIC will use each as an equally valid alternate definition (i.e., there is an implicit OR). The left-hand side of a rule is its body, which defines a pattern to be matched. This can be the topic named in the right-hand side of another rule, a text reference expression (defined below), or a compound expression that defines the logical AND (denoted by "&") or OR ("|") of two or more other rule topics of text reference expressions. Explicit text to be matched without further interpretation is surrounded by quotation marks; names of topics and text reference language constructs are not. The last element in a rule is its weight, which is a real number in the interval [0, 1]. It represents the rule definer's confidence that the existence in a document of the pattern defined by the rule's left-hand side implies that the document is about the topic named in the rule's right-hand side. If a weight is omitted, it is assumed to be 1.0 (i.e., absolute confidence). Note that a weight is a number made up by a human user, based upon his or her experience and insight; a weight is *not* a statistical quantity.

```
team | event => World_Series

St._Louis_Cardinals | Milwaukee_Brewers => team

"Cardinals" => St._Louis_Cardinals (0.7)
Cardinals_full_name => St._Louis_Cardinals (0.9)

saint & "Louis" & "Cardinals"
   => Cardinals_full_name

"St." => saint (0.9)
"Saint" => saint

"Brewers" => Milwaukee_Brewers (0.5)
"Milwaukee Brewers" => Milwaukee_Brewers (0.9)

"World Series" => event
baseball_championship => event (0.9)

baseball & championship => baseball_championship

"ball" => baseball (0.5)
"baseball" => baseball

"championship" => championship (0.7)
```

Fig. 2. Rule base for topic of world_series.

A text reference expression may be a single keyword or phrase, or a lexical context within which two keywords or phrases must be found (e.g., word adjacency, same sentence, same paragraph). So, for example, one can specify that two patterns are of interest only if they occur in the same sentence. Fuzzy (partial) matching versions of these contexts are also allowed. RUBRIC's fuzzy pattern matcher returns a value in [0, 1] that is proportional to the degree that the phrases are in the desired context, i.e., inversely proportional to the logical distance between the two objects in the document. For example, when matching a fuzzy same-sentence context, two phrases in the same sentence might receive a weight of 1.0, within adjacent sentences 0.8, etc.

Rules often define alternate terms, phrases, and spellings for the same concept. Thus, rules can also provide a simple hierarchical thesaurus, with variable weights defining the degree of certainty with which a particular variant is to match. For example, in English "St." is used as the abbreviation for both "Saint" and "Street," and thus "St." is weighted less that the keyword "Saint" in Fig. 2. Rules can also aid multilingual information retrieval. For example, if the database contains text in multiple languages, then the lowest level(s) of rules

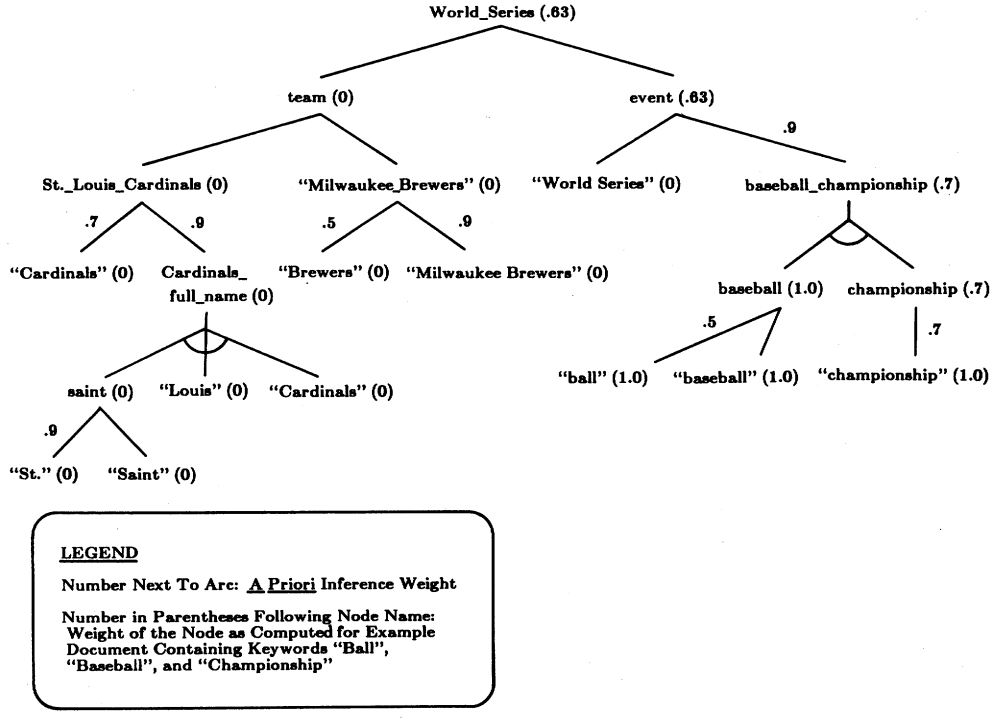


Fig. 3. Rule evaluation tree for world_series topic.

might define synonyms in each language of interest. The more conceptual language-independent rules higher in the hierarchy would remain unchanged.

It has been found useful to provide a new type of rule in RUBRIC, called a *modifier rule*, which enables the user to incorporate auxiliary (or contextual) evidence into the query. Auxiliary evidence is evidence that by itself neither confirms nor disconfirms a hypothesis, but which may increase (or decrease) our belief if seen in conjunction with some primary evidence. The form of such a rule is

if A, **then** C to degree w_1;
but if also B, **then** C to degree w_2

where if w_1 is greater than w_2 then B is disconfirming auxiliary evidence, and if w_1 is less than w_2 then B is confirming auxiliary evidence. This has the effect of interpolating between w_1 and w_2, depending upon the certainty computed for the auxiliary clause B. Thus we might have a rule of the kind:

if (the story contains the literal string "bomb"),
then (it is about an *explosive device*)
to degree 0.6;
but if also (it mentions a *boxing match*),
then (reduce the strength of the conclusion)
to degree 0.3

Here we see the concept of disconfirming evidence in operation; notice that by itself being about the concept *boxing match* is not evidence that can be used to support or deny the conclusion we are trying to establish.

Knowledge bases of rules are expected to evolve over time. Initially the set of rules provided in a knowledge base will capture a small portion of the kinds of knowledge required. New rules are easily added to RUBRIC, currently by means of a standard display-oriented text editor. Existing rules may be modified for experimentation to provide feedback for honing their logical structure, keywords, and weights.

IV. Query Processing

A set of rules defines a logical hierarchy of retrieval topics and subtopics (Fig. 3). A specific retrieval request is carried out by a goal-oriented inference process similar to that used in the MYCIN medical diagnosis system [4]. This process creates and evaluates an AND/OR tree of logical retrieval patterns. The root node of this tree represents a semantic topic or concept that the user wants retrieved; nodes farther down in the tree represent intermediate topics with which the root topic is defined; and nodes at the leaves of the tree represent patterns of words that are to be searched for in the database. Each arc in the tree is weighted such that the intermediate topics and keyword expressions contribute, according to their weight, to the overall confidence that the root topic has also been found. (Unlabeled arcs in Fig. 3 have an implicit weight of 1.0.) Arcs representing the conjuncts of an AND expression are linked together near their common base in Fig. 3.

RUBRIC supports a number of calculi for interpreting the rule weights. Weights are treated as certainty or partial truth values, not as probabilities. Each calculus defines how to combine the uncertainties during such logical deductions as AND, OR, and implication. The default method is to use the functions minimum, maximum, and product to propagate the weights across AND and OR arcs and implication nodes, respectively [4].

Referring to Figs. 2 and 3, we now describe how RUBRIC processes a query. (Annotated traces of the system's operation are found in [1].) When the user types in the conceptual query World_Series, RUBRIC searches its rule base for all rules that

Phrases Present in Document	Support for World_Series Topic
"World Series"	1.00
"Saint", "Louis", "Cardinals"	0.90
"Milwaukee Brewers"	0.90
"St.", "Louis", Cardinals"	0.81
"Cardinals"	0.70
"baseball", "championship"	0.63
"Brewers"	0.50
"ball", "championship"	0.45
none of the above	0.00

Fig. 4. Possible weights for world_series topic.

provide definitions for this topic (i.e., that have World_Series on their right-hand sides). There is only one such rule in Fig. 2, so RUBRIC expands that rule according to its lefthand side. The result is that the World_Series, team, and event nodes of Fig. 3 are created, as well as the two arcs between them. Since team and event are themselves the names of topics, rather than textual patterns, RUBRIC searches its rule base for their definitions. This process continues recursively until all leaf nodes of the tree contain textual patterns.

At this point each document in the database is matched against all of the phrases in the leaves of the tree. For a given document, if a phrase is found somewhere in the document, the corresponding node in the tree is assigned a value of 1.0, otherwise 0. Then the weights at the leaves are combined and propagated up through the tree to determine the overall weight to be assigned to this document.

For example, if a document contained the words "ball," "baseball," and "championship," and no other words referred to in the example rule base, then the nodes of the tree would be assigned the weights shown in parentheses in Fig. 3. The "ball," "baseball," and "championship" leaf nodes all receive a weight of 1.0, and all other leaves receive a weight of 0. The baseball node would then be assigned the value 1.0 because that is the maximum of (1.0 multipled by 0.5) and (1.0 times 1.0). Similarly, the championship node receives the value 0.7. Then, because it is an AND node, the baseball_championship node gets the value 0.7, which is 1.0 times the minimum of 1.0 and 0.7. The event node then gets the value 0.63, which is the maximum of (0 times 1.0) and (0.7 times 0.9). Since there are no keywords in the document that support the team subtopic, the overall weight of the match of the World_Series topic on this document is 0.63 (1.0 times the maximum of 0 and 0.63).

Other combinations of keywords and phrases in a document can satisfy the concept of World_Series to varying degrees. Fig. 4 shows the other weights possible for the World_Series topic, depending upon the dominant phrases that occur in the document.

V. User Interface

A user need only see the highest weighted documents. After the database has been searched, each document that was considered has an associated weight that represents the system's confidence that the document is relevant to the topic requested by the user. RUBRIC sorts these documents into descending order based upon their weights, and groups the documents by applying statistical clustering techniques to the weights. The user is then presented with those documents that lie in a cluster containing at least one document with a weight above a threshold provided by the user (e.g., 0.8 or above). Clustering prevents an arbitrary threshold from splitting closely ranked documents. The threshold may be varied depending upon how much time the user has available to read documents, how important it is not to miss any potentially relevant ones, etc.

RUBRIC is able to explain why a particular document was retrieved. This capability is very important for instilling confidence in users and helping them get a good enough feel for the operations of the system that they can successfully write and use their own retrieval rules. RUBRIC can display each rule that results in a nonzero weight being propagated, as well as the value of that weight. RUBRIC can also show each attempt to match a word or phrase to the document, along with whether or not it matched.

VI. Experimental Results

We have done preliminary experiments with RUBRIC to examine the improvements that can be achieved over a conventional Boolean keyword approach. As an experimental database for testing the retrieval properties of RUBRIC, we have used a selection of thirty stories taken from the Reuters News Service. Our basic experimental procedure is to rate the stories in the database by inspection (i.e., define a subjective ground truth), construct a rule-based representation of a typical query, apply the query to the database, and then compare the rating produced by RUBRIC with the *a priori* rating.

We concentrate on two basic measures of performance. Both of these are based on the idea of using a selection threshold to partition the ordered stories so that those above it are "relevant" (either fully or marginally) and those below it are "not relevant." In the first we lower the threshold until we include all those deemed *a priori* relevant, and then count the number of unwanted stories that are also selected (denoted N_F). In the second we raise the threshold until we exclude all irrelevant stories, and then count the number of relevant ones that are not selected (denoted N_M). The first definition therefore gives us an insight into the system's ability to reject unwanted stories (precision), whereas second gives us insight into the system's ability to select relevant stories (recall).

We selected as a retrieval concept "violent acts of terrorism," and then constructed an appropriate rule-based query. This is summarized in Fig. 5, where we make extensive use of modifier rules. An auxiliary clause is shown linked to its conclusion by a directed arc labeled "Modifier". Application of this query to the story database results in the story profile shown in Fig. 6. (Notice that for presentation purposes the stories are ordered such that those determined to be *a priori* relevant are to the left in Figure 6.) The performance scores for this experiment are

Precision: $N_F = 1$ when we ensure that $N_M = 0$, and

Recall: $N_M = 5$ when we ensure that $N_F = 0$.

This is almost perfect performance, being marred only by the selection of story 25, which, although it contains many of the

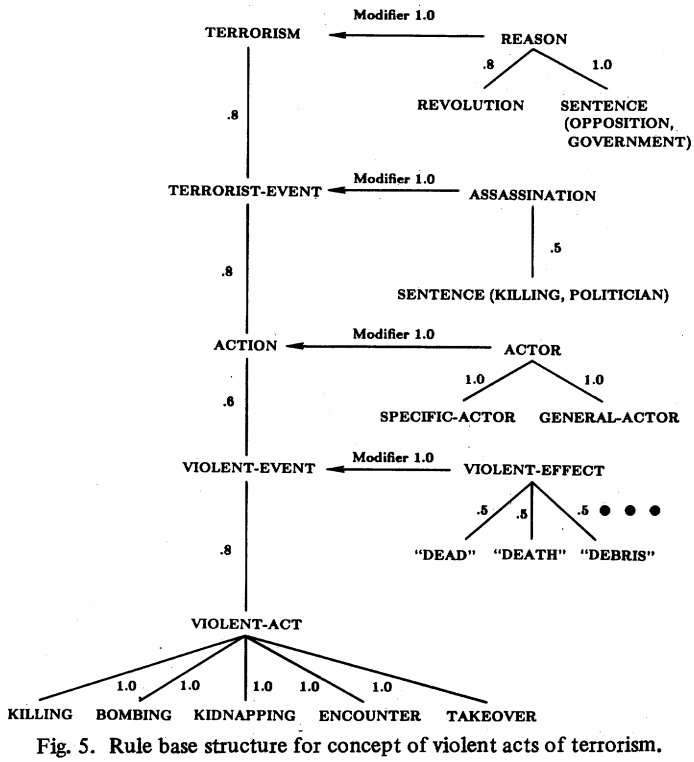


Fig. 5. Rule base structure for concept of violent acts of terrorism.

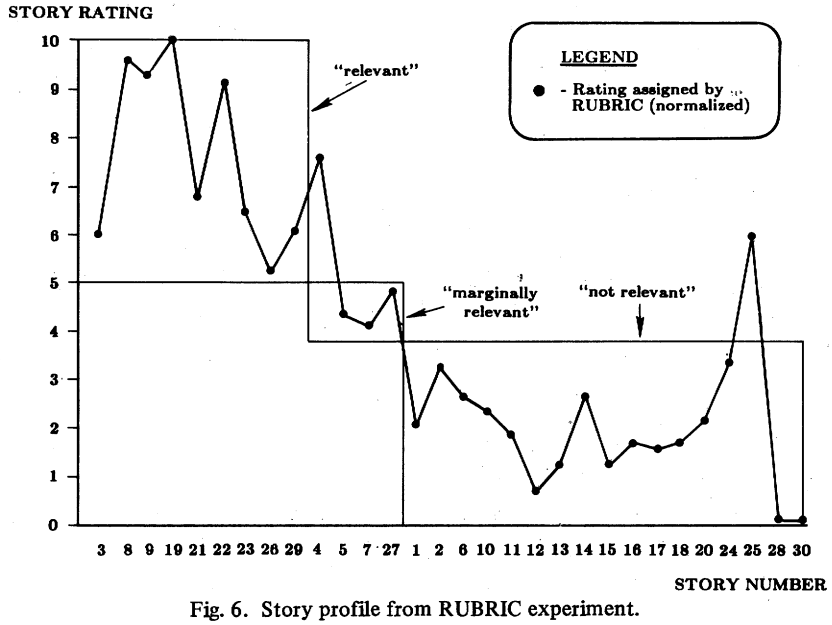


Fig. 6. Story profile from RUBRIC experiment.

elements of a terrorist article, is actually a description of an unsuccessful bomb disposal attempt.

To compare RUBRIC against a more conventional approach, we constructed two Boolean queries by using the rule-based paradigm and setting all rule weights to 1.0 (thus incidentally showing that our method subsumes Boolean retrieval as a special case). One of these queries is shown in Fig. 7 as an AND/OR tree of subconcepts. The only difference between the two Boolean queries is that in the first we insist on the conjunction of ACTOR and TERRORIST-EVENT (as shown), whereas in the second we require the disjunction of these concepts. The conjunctive form of the Boolean query misses five relevant stories and selects one unimportant story, whereas the disjunctive form selects all the relevant stories, but at the cost of also selecting seven of the irrelevant ones.

While these results represent only a preliminary test, we believe that they indicate that the RUBRIC approach allows the user to be more flexible in the specification of his or her query, thereby increasing both precision and recall. A traditional Boolean query tends either to over- or under-constrain the search procedure, giving poor recall or poor precision. We feel that, given equal amounts of effort, RUBRIC allows better

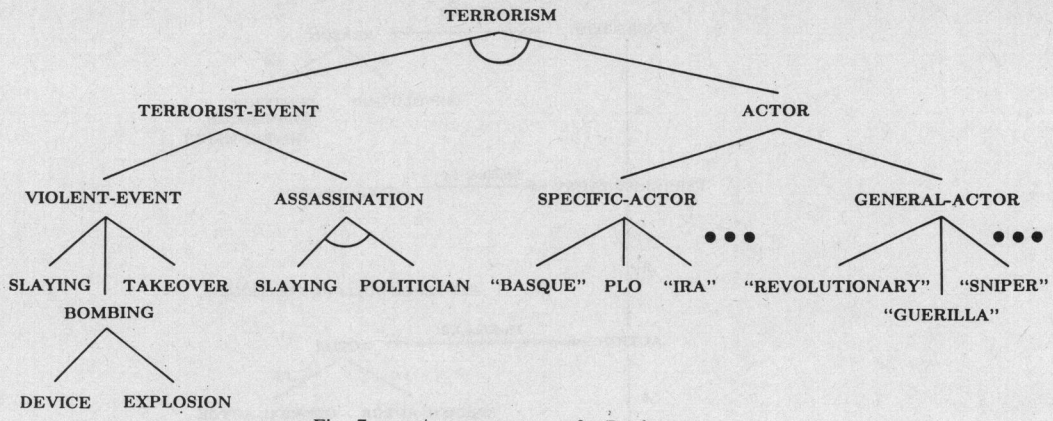


Fig. 7. AND/OR concept tree for Boolean query.

models of human retrieval judgment than can be achieved with traditional Boolean mechanisms.

We have also explored the effects of using different calculi for propagating the uncertainty values within the system [5]. Among these calculi are well-known classes such as those that use "max" and "min" as disjunct and conjunct operators, and those (so-called "Bayesian-like") that use "sum" and "product". Our initial conclusion is that the calculus used is not the major determinant of performance, but that it does interact with how rules are defined.

VII. Future Work

Much additional research and system development are needed to make RUBRIC usable. We are currently providing a better user interface and conducting more complete experiments. The interface for end users will include more focused interactive explanation, analysis of results for sensitivity to specific rules and weights, display of graphs such as Fig. 6, and rule editing. Experimentation will consist of defining, in conjunction with users, larger rule sets for a realistic retrieval domain and then using these rules to retrieve documents from a realistic database.

Other areas of possible future work include making rule evaluation and textual pattern matching more efficient, possibly through the use of heuristics to limit rule evaluation; exploring additional ways of representing and propagating uncertainty in both numeric and symbolic representations; ablative testing to measure how useful each system feature is; extending the text reference language to allow specification of the syntactic role that a word plays in a sentence (e.g., "ship" used as a noun versus as a verb); constructing a more general thesaurus that has a network structure rather than a hierarchical one like rules; and allowing retrieval from multiple remote databases.

VIII. Potential Applications

Application systems based on RUBRIC may be useful for information routing and change detection, in addition to information retrieval. For information retrieval RUBRIC could be extended to work on formatted documents such as messages or bibliographic entries, to work as a front end to existing databases and information retrieval systems, and to segment larger documents by subtopics. RUBRIC could be used to process messages in real-time, filtering the important ones and routing them to the appropriate recipient (human or another program). With RUBRIC, analyses of documents over time could detect statistical changes at a conceptual level rather than just in the use of individual keywords.

References

[1] B. P. McCune, J. S. Dean, R. M. Tong, and D. G. Shapiro, "RUBRIC: A system for rule-based information retrieval," Advanced Information & Decision Systems, Mountain View, CA, Tech. Rep. 1018-1, Feb. 1983.

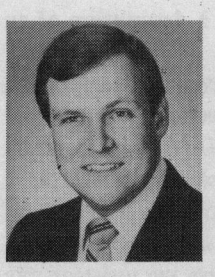
[2] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983.

[3] R. C. Schank and G. DeJong, "Purposive understanding," in *Machine Intelligence*, vol. 9, J. E. Hayes, D. Michie, and L. I. Mikulich, Eds. 1979, ch.24, pp. 459–478.

[4] E. Hance Shortliffe, *Computer-Based Medical Consultations: MYCIN*. New York: Elsevier, 1976.

[5] R. M. Tong, D. G. Shapiro, J. S. Dean, and B. P. McCune, "A comparison of uncertainty calculi in an expert system for information retrieval," in *Proc. Eighth Int. Joint Conf. Artificial Intell.*, A. Bundy, Ed. Los Altos, CA: William Kaufman, Aug. 1983, vol. 1, pp. 194–197.

[6] R. M. Tong, D. G. Shapiro, B. P. McCune, and J. S. Dean, "A rule-based approach to information retrieval: Some results and comments," in *Proc. Nat. Conf. Artificial Intel.* Los Altos, CA: William Kaufman, Aug. 1983, pp. 411–415.

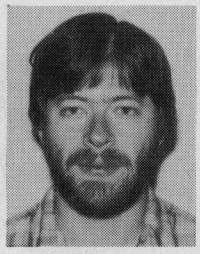


Brian P. McCune (S'74–M'80) received the B.A. (Honors) degree in mathematics from Oregon State University, Corvallis, and the Ph.D. degree in computer science from Stanford University, Stanford, CA.

His professional interests include artificial intelligence, emphasizing knowledge-based and expert systems; software systems, especially environments for software and knowledge engineering; decision support systems; and distributed computing. He is a cofounder of Advanced Information & Decision Systems, Mountain View, CA, where he is Vice President and Manager of the User Aids Program. Since 1980, he has

led research and development of interactive software systems to support decision makers such as battlefield commanders, managers, planners, analysts, system designers, and programmers. He is currently supervising work on expert systems for software engineering, database access, decision support, and a variety of analysis applications.

Dr. McCune is on the Editorial Advisory Board of *Defense Electronics* and the Advisory Board of *The Artificial Intelligence Report*.



Jeffrey S. Dean (S'77-M'79) received the B.A. degree from Hampshire College, Amherst, MA, in 1978, and the M.S. degree in computer science from Stanford University, Stanford, CA, in 1979.

He joined Advanced Information & Decision Systems in 1981, where he has pursued research in artificial intelligence applied to programming environments, software engineering, information retrieval, documentation, and user interfaces.

Mr. Dean is a member of the Association for Computing Machinery, the American Association for Artificial Intelligence, and Computer Professionals for Social Responsibility.

Richard M. Tong (M'80) received the Ph.D. degree from Cambridge University, Cambridge, England, in 1975.

His main research interests are in intelligent decision support systems, fuzzy set theory, theories of approximate reasoning, and artificial intelligence. He joined Advanced Information & Decision Systems in 1980 where he is the Department Head for the Decision Systems Department. He has been conducting a program of research designed to explore the effects of various forms of uncertainty representation in expert systems, and is currently leading a team that is performing research for the development of an intelligent information retrieval system.

Daniel G. Shapiro received the M.S. degree from the M.I.T. Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, in 1981.

Since that time, he has worked for AI&DS on the topics of intelligent editors, uncertainty representation, AI applied to information retrieval, and AI planning. He is currently directing research towards a planning system which will guide the DARPA autonomous land vehicle through long distances in cross-country terrain.

SPD: A Humanized Documentation Technology

MOTOEI AZUMA, TETSU TABATA, YOSHIHIRO OKI, AND SUSUMU KAMIYA

Abstract—**The SPD (Structured Programming Diagram) is a documentation technology used to design well structured programs. With SPD, designers can easily express functional structure, control structure, and physical layout of a program on one sheet of paper. Its straightforward expression appeals to both document writers and readers. SPD concept and conventions are introduced in this paper. SPD usage is then explained with a program design example. Other documentation technologies used in coordination with SPD are briefly touched upon. Finally, SPD reputation and evolution in the last ten years are reviewed.**

Index Terms—**Documentation, software development, Structured Programming Design.**

I. Introduction

NOBODY is successful in software development without sufficient documentation. Documentation provides information to support the effective design, management, implementation, and maintenance, and to facilitate the interchange of information. Documentation technology is important in order to accomplish development smoothly and efficiently, and to maximize the return on development investment.

Manuscript received November 7, 1983.

The authors are with the NEC Corporation, 7-15 Shiba, Minatoku, Tokyo 108, Japan.

As a result of software engineering research and development activities up to now, various useful programming technologies have been developed, such as structured programming [1]-[3], stepwise refinement [4], top-down design [5], one page coding [6], structured design [7], [8], composite design [9], modular programming method [10], [11], Warnier programming method [12], M. Jackson programming method [13], etc. Reflecting these technologies, flowchart usefulness was questioned [17], and various documentation techniques which support these programming technologies to be use, have also been developed, such as Warnier-Orr diagram [12], [14], [15], Jackson diagram [13], [16], NS chart [18], [19], Chapin chart [20], HIPO [21], etc. Although these documentation technologies differ in details, they have a common essential characteristic, that is, to represent a hierarchy of program functions and basic control constructs in a comprehensive fashion. This characteristic is the minimum required quality of a documentation technology for a modern programming method.

In order to improve software productivity, one of the best ways is to reuse existing programs. Ironically, it is obvious that the fewer new programs are created, the greater the productivity gained. However, it is very difficult to manage what and

0098-5589/85/0900-0945$01.00