Search Strategies and User Interface[†]

RICHARD H. GIERING

Mead Technology Laboratories, Research Park, Dayton, Ohio 45432

Received August 30, 1974

This paper discusses what in The Mead Corporation have found to be certain aspects of search strategy with respect to searching large data bases and certain problem areas that remain. The paper defines some problem areas, especially problem areas in the freeing-up of the interface language between the user and the search process. Four specific areas in the interface language are defined, and a recommendation is made concerning basic research into some linguistic meanings for the four areas.

What is a large data base? I would venture a guess that although each one of us has our own definition of a large data base, there are some subtle differences between those definitions. We at The Mead Corporation are involved with large data bases in many classes, and I have found that the people involved with large data bases of the bibliographic class are far ahead of others in their thinking concerning the use of large data bases in general. But the problem areas for other types of large data bases (large for other reasons) also present some formidable difficulties.

Now, under what criteria can a data base be large? First, a data base can be large because the number or records or entries in that data base is large. Five hundred thousand entries in a data base represent a large data base, independently of the size of each of the individual entries. A data base containing merely bibliographic information, including a limited number of manually assigned index terms (on the 500,000 entries), could well require storage of a mere 50 to 100 million characters of information. Not a very large amount of storage in anyone's evaluation, but a large number of entries.

The second criterion for defining a large data base, then, could be the numbers of characters involved. Again, a bibliographic data base with only 10,000 entries in it, but with each entry being the full text of the document involved, represents a data base whose character count is on the order of a billion.

A third measure against which a data base could be considered is a measure very important to the search or selection process. If the number of selectable elements, possibly a controlled thesaurus, is on the order of a thousand, independently of the size of the data base in characters or entries, the data base could be said to be small. On the other hand, in a full-text operation, where every potential word and every potential value, including all variants, are search-selectable elements numbering in the hundred-thousand area, the data base could be said to be large, especially if the software involved required a serial search of those elements.

Now, finally, a fourth condition under which a data base could be said to be large, again independently of the previous three, appears primarily when we get away from bibliographic data bases—and I believe that this Conference should address this question—because, as bibliographic experts, we have led the field. We have been at the "head of the pack," so to speak. In defining Information Science activities, designers of management systems, for example, are just beginning to think about large data bases in the way we in the bibliographic area have been thinking of them for years. This last condition, under which a data base might be said to be large, would be in the number of fields and/or files associated with the intricate make-up of the individual data base.

We will be addressing all four of these areas—all four of these definitions of large data bases—during our discussion. If, during the discussion, there is disagreement between what is said by two people, myself being possibly one of them, or two people within the audience during the question and answer period, it is quite possible that the difference lies in the assumed definition of some of the terms, including this one.

I would now wish to raise another question from a definition standpoint. Who, or what, is the user? We have been inundated recently with calls for standardization of systems specifications of systems intercourse. There have been a wide variety of complaints about having to learn multiple disciplines of access for multiple large data bases. I wish to propose a hypothesis and suggest that it be the subject of some discussion. Is it not possible that the reason there is such a dichotomy of user interaction specifications is that the individual definitions of what and who the user is vary? The attributes that are ascribed to this person called "the user" by system one vary and differ greatly from the attributes associated with this person called "the user" by system two. Until we have some commonality of understanding, until we agree on the minimum set of attributes for "the user," we will continue to have a dichotomy of systems.

The understanding and agreement as to what a user is are important in still another area. The on-line interactive use of data bases, large or small, is in its infancy. As the user population grows, it is quite logical to assume that the profile (make-up) of that population will continue to change. Unless we, as an industry, understand this changing phenomenon, we will not be adequately ready to support the end user. I am going to describe this end-user now by *my* definition—this is the definition of the end user profile as we in The Mead Corporation see it. Up until recently, end users, that is, the users on the terminal, have had at least a smattering of training in the Information Sciences technology. They have been able to assimilate meanings of terms such as "Boolean logic," "operators," "arguments," etc. For the most part, we have been building the system for our own inner circle of users. That is not the definition of the user as The Mead Corporation sees it. The definition I am about to give you looks into the future where, *via* the advanced technology of such things as cable TV, the end users of both large and small data bases will be scared out of their wits by such terms as Boolean logic, even though in their normal day-to-day, natural language communication, they use an "AND" and an "OR" all the time.

[†] Presented in the "Conference on Large Data Bases," sponsored by the NAS/NRC Committee on Chemical Information, National Academy of Sciences, May 22–23, 1974.

Definition: the end user of an on-line interactive data base system has no knowledge of formal logic, of formal discussion, of formal anything, but rather is a professional in a profession *outside* our Information Science profession. He is in a profession we do not understand, and, more importantly, he does not understand our profession and *doesn't want to.*

This, then, leads into search strategies. What search strategy can an individual of this category use. Frankly, we are back to search strategies that we always refer to as "K-I-S-S," *Keep It Stupidly, Simple.* For the most part, the users always refer to the most universally accepted term—whether that's a manually assigned thesaurus term or a text term. He thus obtains a large set of answers, which he then reduces until he sees a set that he thinks he can work with. Note that I am talking about "*he* thinks," "*he* can work with," "*his* discussion," and "*his* term."

What about the help he needs. It is very critical, in our estimation and in support of search strategy process, that the user has access to an on-line, interactive, multi-level tutorial to help walk him through, so to speak, the process of selecting the answers he wishes to use.

Obviously, such a user—one who is not oriented in the intricacies of Information Science—would love to be able to communicate with the computer in something akin to natural language. We have found it necessary to start making some subtle additions to search terminology available to the end user. Let me give you an example. We all, I am sure, understand the true meaning of an "AND" conjunct with respect to an "OR." We understand the implied meaning of the expression "(A and B) or C" as it is expressed with parentheses—and, by the way, the end user hates parentheses. They have absolutely no meaning because he never speaks with parentheses, nor does he ever write with parentheses. Linguistically, that expression may be simplified as "(DOCTOR and PATIENT) or LAWYER." This is implicit—we know that DOCTOR and PATIENT go together.

Using the same linguistic expression, "DOCTOR and HOSPITAL or OFFICE," this exemplifies another form of "A and B or C," and it has a completely different implied meaning. Here, DOCTOR is associated with either of them, and the parenthetical notation of the meaning of that linguistic expression is different A and (B or C). (DATA/CENTRAL) allows for both expressions because, in a search strategy, a user needs to be able to communicate in the way *he* is thinking of the question. Again, basic search strategy is to keep it simple. Obtain a large set of answers and narrow it down to the appropriate set needed. But it is also important that the user be able not to lose any sets and, if his narrowing process takes him too far down, he must be able to backtrack, move in another direction, and to wipe clean the other areas because he doesn't want to be reminded of those areas—he doesn't want to be constantly shown his previous mistakes.

Now this brings us to another part of the definition of who the user is. A professional in his own profession, utilizing a very complex thing called a computer, is exceptionally peer conscious. What do I mean? He surely does not want his peers to see him make a mistake. The more flamboyant a terminal is, the less likely a man is to use it. A loud, clattering teletype is a very flamboyant terminal; a large screen blinking as a monster is a very flamboyant terminal. A much more usable terminal would be the private desk-top unit, an example of which is shown (Figure 1). This one happens to be on my desk, but it is this kind of terminal that we find to be extremely important for the end user, especially the ones who are extremely peer conscious. We can discuss the intricacies of search strategy, but the actual search strategy used is highly dependent upon the profile of the user, and the user is not likely to try anything special. He will not try any special kind of combinations that

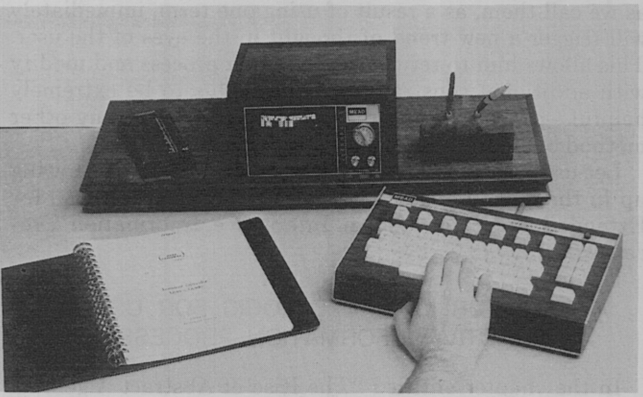

Figure 1.

he is not quite sure of if he feels that somebody might well be looking over his shoulder at what he is doing. I might add that this includes monitoring his activity. He doesn't want anybody to laugh at him, including those "stupid computer people," those "holier-than-thou Information Science experts." "I'll do it my own way before I do something that lets them get one up on me"—that's his attitude.

Now let me address another area of search strategy with respect to full-text systems. That area is synonymy or equivalences—we refer to it as "search-onymy." Of course, there are certain terms that will always and forever, within the context of a particular data base, be synonymous. These equivalent terms will, of course, be used by the appropriate data systems automatically. I would like to ignore that part of the problem since all of us have addressed that problem. People use them, they are used by the computer, and generally they are completely transparent to this terminal end user. He doesn't have to think about them; he knows about them when he sees the synonymous term "highlighted," but normally he doesn't have to worry about it. This applies to plurals and other forms of the word. What, however, about those terms that are synonymous sometimes, under certain contexts, and not synonymous under others. The mere display in alphanumeric order of the different word forms surrounding the word (selected for search) alphabetically is not fully satisfactory, since the display only answers the question of the various word forms and does not address the other question of synonymy: "is head synonymous with cranium?." One alternative solution is the display of a controlled dictionary or thesaurus in which broader and narrower terms are related to the individual term; the old controlled thesaurus used for manual indexing now has another use: aiding in search strategy. In fact, I would venture a guess that the use of the controlled thesaurus will increase in this area because the end user needs help. But the problem here goes well beyond that! What about the terms that have been forgotten in the controlled thesaurus? One cannot afford to continually update this thesaurus with every word seen in the text of the material on hand—it is just too expensive. We now come to the area that we have found to be extremely important in search strategies. And that is the ability to browse through a set of answers textually. The user can then subtly, and without realizing he is doing it, make use of a facet of natural language writing that indicates that the same term should not be used too many times in the same paragraph or set of paragraphs. Rather, synonymous expressions should be used to avoid monotony. As data bases, large data bases (and in this case I am using the term "large data base" especially with regard to the number of words (selectable words) in each entry and obviously the number of selectable terms in each data base), grow and they include more and more of the text, whether it be an abstract or the actual originating author's text, the ability to browse through the material, finding these searchonyms as we call them, as a result of using one term, immediately will trigger a new trend of thought in the eyes of the user. This allows him to return to the search process and modify with additional constraints. We find this to be extremely helpful—probably as helpful to the end user as any other method of aid in this area.

Let us now consider the problems that I foresee showing up in the not too distant future, which I entitle A Call for Extended Logic for Use in Interactive Information Processes.

A CALL FOR EXTENDED LOGIC FOR USE IN INTERACTIVE INFORMATION PROCESSING

In the chapter entitled "The Rise of Abstract Algebra," Carl B. Boyer states:

"The history of logic may be divided, with some slight degree of oversimplification, into three stages: (1) Greek Logic, (2) Scholastic Logic, and (3) Mathematical Logic. Whereas in the first two stages logic theorems were divided from ordinary languages, the third stage proceeds in a contrary manner—it first CONSTRUCTS a purely formal system, and only later does it look for an interpretation in everyday speech. It's floruit date (the stage) is really the year in which Boole's first book appeared."[1]

Since the days of Boole and DeMorgan[2] (Boole's first work was published in 1847), the succeeding century has been spent in the association with the fundamental laws[3] of logic defined in the formal system. All expressions can be defined in combinations of the basic fundamentals: intersection (&) and inclusive union (or).

With the rise of users of computer systems (which operate on logical circuits and/or programs), who themselves are not trained in the formalism of mathematic logic (Boole rightly claimed that logic is associated with mathematics rather than with metaphysics), it is necessary to close the circle described above by defining the complex elements of logic, so that the user might use a more simple form of expression in communicating his requirements. We may be entering into a fourth phase in the history of logic in which expansions of the basic two elements into formalized linguistic expressions are generated to allow for less ambiguity in communication—not only between man and automaton, but also man to man.

This period in the history of logic is not without its own problems in construction of formal expressions. As will be seen further in this presentation, an additional basic logic element may be in the process of being defined. Additionally, logic has been explained using sets (Venn diagrams). Sets have been, up to today at least, a one-dimensional concept. It is barely possible that we have embarked (using operators and expressions) on the definition of a second dimension in the set theory. We will discuss four problem areas requiring formal definitions:

1. Additional (complex) logical expressions
2. Arithmetic expressions that defy algebraic logic
3. Multidimensional expressions
4. Concatenation and other "G-connectors"

As interactive information processing becomes more used by non-information science professionals, it becomes apparent that extensions to logic are necessary for this unique set of information, especially when the information being processed is multivalued (*e.g.*, textural or periodic). It is readily admitted that there is no specification that cannot be properly written by the user or properly processed by computers using standard conventional Boolean logic. How many non-ADP professionals, who would have use of an information system, are readily able to discern the difference between the subtle ramifications in Boolean logic in order to enter semantically correct and syntactically definable elements of search? There are, of course, some, but their numbers are relatively few in comparison to the numbers of potential users of information systems. Extensions of logic, therefore, are needed in order to make easier the definition of the problem by nonprofessional users and, much more importantly, to make for efficient processing of the problem by the computer.

We have, in some sense, already done it for one expression. There is no need, in formal logic, to use the exclusive OR, defined, using the two basic elements, as:

$$(\text{A ER B}) \equiv (\text{A or B}) \ \& \neg (\text{A} \ \& \ \text{B})$$

In programming, however, it has been found to be cumbersome to use only the fundamental elements; we have, therefore, engineered an exclusive OR operation for programmers to use. Other combination forms (called MACRO) for complexities of logic need to be defined, based upon combinations of the basic elements.

A second example, already in use, is the AND condition. We are all fully aware of the relationship under standard Boolean logic of the & and the OR condition. Since there is an implied ambiguity, the expression "A and B or C" can have either of the following meanings (the word AND used linguistically rather than formally):

(A & B) or C
A & (B or C)

Classically the defined expression "A and B or C" has the meaning (A & B) or C, which linguistically is the combination of A and B or the single element C. This is exemplified by the actual conversational expression "Doctor & Patient or Lawyer." A great deal of effort and a great deal of rigorous mathematical definitions have gone into all of the ramifications of this logic. Witness DeMorgan's law in which the reversal of operators is effected by the establishment of parenthetical nesting of the NOT operator:

NOT (A or B) is identical to (NOT A & NOT B)

People, on the other hand, do not always think in a Boolean manner. In fact, it is this author's opinion that a great many people think in the reverse form and, without any other specification, the expression "A and B or C" tends, in normal conversational discussion, to take on the meaning established in the second line above, spoken as "the single element A in combination with either D or C." A conversational example is DOCTOR and HOSPITAL or OFFICE. This definition of logic has never been passed through the rigor of mathematical treatises, except in the use of parenthetical notation. No "not-reversal" (DeMorgan's duality) has been defined, and all of the rigor of the combination of the two has not been defined except by use of actual parenthetical notation. It is *believed* the NOT (A and B or C) would be, based on parenthetical notation, the same as saying NOT A or the combination of NOT B and NOT C. It is time that we in the Information Science profession applied the rigor of logic to this preceding linguistic expression so that our users, who are not necessarily mathematically oriented, are able to express themselves in their language either way, rather than having our language imposed upon them. Based upon the opinion that both forms of the logical AND need to be defined and used by users, both a superior (to OR) and inferior AND are definable in (DATA/CENTRAL). In this writing, the word AND is considered to be superior while the & is considered inferior:

A and B or C means A & (B or C)
A & B or C means (A & B) or C

```
1.#AUTHOR = JONES AND#DATE ≤ JUNE, '71

2.#AUTHOR = JONES AND#DATE ¬>JUNE, '71
```

Figure 2.

A second area is one in which there is at least one anomaly in normal logic expressions, as found in general computer-oriented languages. Consider, for example, the two search requests shown in Figure 2. Note that the pound sign refers to the field in which the conditions are to be satisfied, while the operators are defined as, for this example, ≤ meaning "less than or equal to" and the operator ¬> meaning "not greater than."

In normal algebra, the two statements have identical meaning! It is possible that in data base systems they do not. Definition of negation: The logical definition (in normal expression) of the negation process requires the removal, from the set of possible answers, of any answer satisfying the positive form of the negative expression, for example:

> "Any hotel but not the St. Francis" means to find a list of All Hotels and remove from that list any hotel positively named St. Francis.

Take a simplified form of the example. The entries are edited such that, on input, the field name DATE can contain only one value, the date of publication. That value, however, may be an actual date or it may be an indication of an unknown date. Please note that an unknown date is definitely not a date of zero time. An entry with an author named JONES with an UNKNOWN indication (absence of the date) would not satisfy the first statement; therefore, it would not be an answer. By not having a date, it cannot be considered to be not greater than and it would not be removed from the set of answers. Therefore, the only criterion valid to determine the satisfaction to the search is the initial JONES criterion; it, therefore, satisfies the second statement.

Admittedly, the above can be expressed in a different manner by the additional specification of the editing criteria that the field must contain either a valid date or it must contain the word UNKNOWN. Then the second search could be rephrased (to make it the same, logically, as the first statement) into:

#Author = Jones and #Date (¬>June '71 or = Unknown)

This becomes a human engineering or user-interface problem because it is beyond reason, in our belief, to expect the novice (or the non-information science oriented) terminal user to remember all of the various editing criteria for files that can and do contain in excess of a hundred fields and especially for files in which the fields were added at different times by different people. The solution, then, is to extend the language such that the concept desired can be expressed without resorting to looking up the edit criteria.

As information processing takes on more and more of the job of handling multivalued or textual data, we must understand the subtle difference between the two. Textual data here are defined as any data found in fields of individual entries (records) such that the structure of the data cannot be predefined; multivalued, on the other hand, refers to fields that can contain multiple separately searchable values. This, of course, includes the name of a person in a personnel record, his address, the name(s) of the school(s) he attended, etc., prior to the preestablishment of arbitrary codes for these data.

Now for a third area where extensions to logic (and especially their meanings for processing purposes) are necessary. (The "or" as used below is the normally used inclusive OR; the ER used below is the exclusive OR). In normal file handling logic, only two of the following nine possible combinations have defined meanings:

$$(\text{Field 1} \left\{\begin{matrix}\&\\ \text{OR}\\ \text{ER}\end{matrix}\right\} \text{Field 2}) = (\text{Value 1} \left\{\begin{matrix}\&\\ \text{OR}\\ \text{ER}\end{matrix}\right\} \text{Value 2})$$

Normally, the use of the conjunctive "&" connector and the exclusive OR-(ER) to the right of the operator is not defined. In the meaning below, the phraseology of the verb "appear" is used as though the ⟨operator⟩ is the logical appearance operator. The reader is reminded that the definitions have slightly subtle changes of meaning if the operator is different. Here we define all nine as follows:

1. (Field 1 or Field 2) ⟨operator⟩ (Value 1 or Value 2) means that the occurrence of either value (or both values) (a value may be a word, phase, or—if the operator is arithmetic—an arithmetic value) in either field (or both fields) satisfies the request and any such entry is considered valid for the retrieval and display process.* Example: Find all documents whose country-of-publication or country-of-nationality is Germany or France.

2. (Field 1 or Field 2) ⟨operator⟩ (Value 1 & Value 2) means that both Value 1 and Value 2 must occur in either Field 1 or Field 2 for the entry to satisfy the search and be available for retrieval and display. Example: Find all medical histories in which either Record-of-Treatment or Post-Operative-Care deals with both the heart and kidneys.

3. (Field 1 or Field 2) ⟨operator⟩ (Value 1 ER Value 2) means that either Value 1 or Value 2, but not both, must appear in either Field 1 or Field 2. Example: Find all medical histories in which either Record-of-Treatment or Post-Operative-Care deals with either the heart or the kidneys but not both.

4. (Field 1 & Field 2) ⟨operator⟩ (Value 1 or Value 2) means that either value must exist in both fields for the entry to satisfy the search. Example: Find all projects that had as both primary objective and methodology the use of either rockets or missiles.

5. (Field 1 & Field 2) ⟨operator⟩ (Value 1 & Value 2) means that both values must occur in both fields for the entry to satisfy the requirements of the search. Example: Find all chemical compounds that have both hydrogen and fluorine listed in the two fields: elements-used and elements-reacted.

6. (Field 1 & Field 2) ⟨operator⟩ (Value 1 ER Value 2) means that either Value 1 or Value 2, but not both, must appear in both Field 1 and Field 2.

7. (Field 1 ER Field 2) ⟨operator⟩ (Value 1 or Value 2) means that either Value 1 or Value 2 or both Value 1 and Value 2 must appear either in Field 1 or in Field 2 but not in both fields.

8. (Field 1 ER Field 2) ⟨operator⟩ (Value 1 & Value 2) means that both Value 1 and Value 2 must appear in either Field 1 or in Field 2 but both values are not to appear in both fields.

9. (Field 1 ER Field 2) ⟨operator⟩ (Value 1 ER Value 2) means that either Value 1 or Value 2 (but not both values) are to appear in either but not both fields.

We have run some preliminary studies attempting to define the results of both the positive and negative uses of these three connectors in nine combinations yielding 144 unique expressions. The definition of the negation operator (¬), it is believed, bears repeating: the negation operator, applied to a value, asks for any entry containing the pres-

* An example of the form when one value is textual and the other is arithmetic would be:

(Field 1 or Field 2) (= Value 1 & < Value 2)

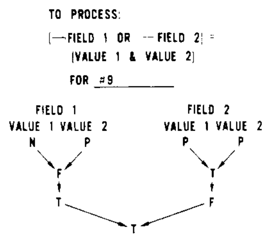

Figure 3.

16 COMBINATIONS:

ENTRY	FIELD 1		FIELD 2	
	VALUE 1	VALUE 2	VALUE 1	VALUE 2
1	P	P	P	P
2	P	P	P	N
3	P	P	N	P
4	P	P	N	N
5	P	N	P	P
6	P	N	P	N
7	P	N	N	P
8	P	N	N	N
9	N	P	P	P
10	N	P	P	N
11	N	P	N	P
12	N	P	N	N
13	N	N	P	P
14	N	N	P	N
15	N	N	N	P
16	N	N	N	N

Figure 4.

ence of that value to be deleted from the set of answers to which it applies. The negation operator, applied to a field (or segment), asks for the complement of that segment's action with reference to the specified combination of values.

In the study, there were 16 combinations of entries that were evaluated. In Figure 3, N refers to the nonpresence of the value in the segment (or field) while P indicates presence. The table was generated, based upon the way an implementation might process the data: Evaluate the right-hand portion of request first, but separately for each segment (or field) specified on the left side (independent of the sign associated with the segment); reverse, if necessary, for the sign of the segment; and combine, based upon the connector between the segments. *Ergo,* for entry number 9, for the following request (one of the 144 expressions), the processing is specified in Figure 4.

$$(\neg \text{Field 1 or } \neg \text{Field 2}) = (\text{Value 1 \& Value 2})$$

(It is noted that this request, as is the case for all of the above conditions, can be stated in terms of the basic Boolean elements. It is believed that this would be:

$$(\neg \text{Field 1} = \text{Value 1 \& Value 2})$$

or

$$(\neg \text{Field 2} = \text{Value 1 \& Value 2}).)$$

It should be apparent that many anomalies appear when the "not" operator is evaluated. Some of the propositions needing rigorous proof seem to be the following.

Proposition 1: The negative of an expression is not necessarily the negative of the parts.

Proposition 2: A not sign, applied to a complete Boolean request expression, can be applied only to the fields or to the values, but not both under discussion, and then DeMorgan's Law applied for values and does not affect the field specifications.

Proposition 3: When applying DeMorgan's Duality Law to the "taking in" of the NOT operator, parentheses may *not* be removed, as they normally define hierarchic relationships.

Now for the fourth area of discussion needing expansion. It has become apparent that many users of information systems (especially those processing text) will have a distinct requirement to specify relationships of the various segments (or fields) of information such that the fields can be considered to be concatenated without necessarily restructuring the physical file. A highly simplified example should suffice to define the problem. Consider a personnel file in which three of the fields or segments of that file are defined as Job-History-External, Job-History-Internal, Current-Position-Description. Please note that each of these fields contains large amounts of text. Consider the request: Search the records for anyone having experience in Missiles and also in Aircraft (disregard, for purposes of this problem, the additional words that might satisfy the requirement). It is not merely satisfactory to say:

(Job-History-External or Job-History-Internal or
Current-Position-Description) = Missile & Aircraft

The true logic of the expression above states that both words, Missile & Aircraft, must appear in any combination of Job-History-External, Job-History-Internal, or the Current-Position-Description fields. The condition expressed below is, in fact, that which is implied by this verbalization of the problem:

((Job-History-External or Job-History-Internal or Current-Position) = Missile & Aircraft) or
((Job-History-External or Job-History-Internal) = Missile & Current-Position = Aircraft) or
((Job-History-External or Current-Position) = Missile & Job-History-Internal = Aircraft) or
((Job-History-Internal or Current-Position) = Missile & Job-History-External – Aircraft)

Obviously, the statement of the expression of the problem as defined above is a bit horrendous for a nonprofessional to attempt. One solution is to restructure the file with the three previously defined fields now concatenated into one. A terminal user could then request the occurrence of both words in the one new field. He would, of course, have to wait (at the terminal) while the restructure (just for him, probably) took place; he would most likely become frustrated. It is believed, therefore, that the best approach is to define a new Boolean connecting expression called a GOR. Using the GOR, the problem can be defined thusly:

(Job-History-External GOR Job-History-Internal GOR
Current-Position) = (Missile & Aircraft)

As can be noted, the expression of the problem is now much simpler. Of course, the GOR has (currently) no meaning when used between two or more values to the right of the operator. It also has no meaning when the sign associated with any of the five elements (three segments, two values) is positive and the connector between the two values is OR. When either condition exists, then the answer is unique with respect to single connectors. This is caused by the fact that the concatenation (inclusive GOR) of the segments is based on positive values (presence of the value). Postulation: The negative of the concatenation is *not* the concatenation of the negative (nonpresence). Another way of saying it would be: absence of the same segment pair. It is believed that, as the science of automated information processing advances, a whole series of G-connectors will need to be defined to aid the non-ADP oriented user in expressing a request for the solution to his problem.

For example, the defined GOR is an inclusive concatenation of the fields under consideration. It is highly probable that, as usage of information systems grows, the need for a definition of the exclusive concatenation, referred to as GER, would be required. It could be defined as: "The inclusive concatenation "MINUS" those entries in which the same value appears in both segments."

Additionally, the concatenation–intersection (G&) might well be required. In both of these connectors, the location of the value in the multivalued segment may wish to be used as opposed to the normal GER and G& where only the presence of the word is used. As a final example, consider the following for a special G-negation:

```
An application is defined with one logical file containing
78 segments or fields. A predefined pseudo-segment
called SEG80 is defined as the concatenation of
Segments 1, 7, 13, 17, and 18.
```

Thus:

Seg 80 ≡ (Seg 1 GOR Seg 7 GOR Seg 17 GOR Seg 18)

A user, on an interactive terminal, wishes to access the file with the following request:

(Seg 1 GOR Seg 7 GOR Seg 13 GOR Seg 17 GOR Seg 18) = (Word & WD)

It is identical with

Seg 80 = (Word & WD)

Another user might wish to access the same file except that, for this user, the concatenation should not include Seg 13. The special G-Negation (G−) would allow this as:

Seg 80 G− Seg 13) = (Word & WD)

As long as only the GOR and the G− are the only available G-connectors, the processing formalism is not too complex; once, however, other G-connectors are required (especially the GER & the G&), the composite meaning may become most complex.

As can be seen, we have attempted to define four areas in which additional man-machine conversational linguistic expressions can and, we believe, should be defined to make easier the way our users may communicate with the computer. To define his search strategy (especially as the user becomes more knowledgeable), the user will want ever-increasing capabilities with even more simplified forms of expression until, in the long run, the user will be able to converse verbally in natural language with the computer.

LITERATURE CITED

(1) Boyer, C. B., "A History of Mathematics," Wiley, New York, N. Y., 1968, p 633.
(2) Augustus DeMorgan, 1806–1871.
(3) Boole, G. S., "Investigation of the Laws of Thought," Dover Publications, New York, N. Y., 1854.

Data Security[†]

M. J. ORCEYRE

IBM Corporation, Poughkeepsie, New York 12602

Received December 13, 1974

Data security is a rich and complex subject dealing with the protection of the computing capability from all threats to its continuity. Some fundamental elements of the process of achieving a reasonable, prudent measure of that protection are considered.

Data security is a rich and broad topic, one that is of increasing concern to data-processing-oriented people at all levels, and is receiving more (and more formal) attention from users and manufacturers alike. The definition of data security should indicate the scope, complexity, and pervasive nature of the subject: it is simply the safety of data (and necessarily also of the system) from improper disclosure, modification, or destruction—whether these are accidentally or intentionally caused. Note that this definition applies as well to the manual as to the EDP operation. Note, too, that it is a global definition; no threat to the continued well-being of the operation is excluded. I intend this to be a complete—however brief—discussion, so you may expect me to deal with all sorts of situations that threaten the safety of data, ranging from technologically complex penetrations of computing systems by highly trained intruders, to earthquakes, to coffee spilled into the machinery, and so on.

† Presented in the "Conference on Large Data Bases," sponsored by the NAS/NRC Committee on Chemical Information, National Academy of Sciences, May 22-23, 1974.

In fact, when you yourselves deal with data security, keep this breadth of scope in mind. After all, to protect data you must understand the threats to those data. To protect data completely against all threats is an unrealizable goal; to protect data to some reasonable extent against reasonably predictable and probable threats is a prudent and practical goal. To accomplish the latter, you must undertake a risk assessment, which involves gaining the clearest possible understanding of the nature of that which you must protect and also of the relative probabilities of the events that threaten the well-being of what you must protect. If you do not have this understanding, you cannot assess risks; if you cannot assess risks, you cannot prudently determine protective measures; and if you cannot prudently undertake protection you cannot know that you are protected.

It is my intent to review some fundamental elements of the process of achieving protection. One of these elements is a clear understanding of the need for protection, and this need—which you all must have to one degree or another—springs from a number of sources: