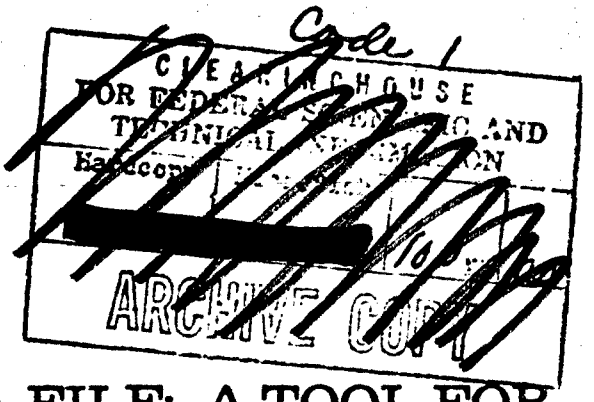
MEMORANDUM
RM-4793-PR
DECEMBER 1965



RELATIONAL DATA FILE: A TOOL FOR MECHANIZED INFERENCE EXECUTION AND DATA RETRIEVAL

Roger Levien and M. E. Maron

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM

RM-4793-PR

DECEMBER 1965

RELATIONAL DATA FILE: A TOOL FOR MECHANIZED INFERENCE EXECUTION AND DATA RETRIEVAL

Roger Levien and M. E. Maron

his research is sponsored by the United States Air Force under Project RAND—Con-act No. AF 49(638)-1700—monitored by the Directorate of Operational Requirements id Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. iews or conclusions contained in this Memorandum should not be interpreted as presenting the official opinion or policy of the Un'ted States Air Force.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA •

PREFACE

This Memorandum describes the background and status of a current RAND project dealing with automatic data storage and retrieval. The research emphasizes the development and testing of logical techniques for data retrieval and inference-making. These techniques are being implemented in the form of computer routines, and tested on a large corpus of factual information concerning the field of cybernetics.

This Memorandum provides an overall view of the system under development and describes the major design choices. Subsequent Memoranda will detail the components of the system.

The research reported here should be of interest to Air Force organizations concerned with storage, retrieval, and inference-making from large bodies of data. Among the potential areas of application are command and control, research management, technical information dissemination, and advanced development planning.

SUMMARY

This Memorandum describes the background and status of a current RAND project dealing with automatic data storage and retrieval. The research emphasizes the development and testing of logical techniques for data retrieval and inference-making. These techniques are being implemented in the form of computer routines, and tested on a large corpus of factual information concerning the field of cybernetics.

Sections III through VI present the theoretical base for the proposed system. A major decision in designing a data retrieval system of this sort concerns the nature of the language that will be used. Unlike some question-answering systems, information is not stored in ordinary (natural) language. Section III discusses the reasons for this decision, examining some of the logical and linguistic aspects of the overall problem, and explaining the selection of an artificial language, an interpreted relational calculus, for use as the information language.

Section IV introduces and summarizes the theory of relations. The theory of relations is that part of modern logic that deals with relations, their properties, and the relationships that hold among them. The section includes a discussion of how relations can be represented and how some operations on given relations can be computed so as to derive new relations.

Section V describes some typical data retrieval requests and how they can be executed once incoming data have been mapped into the artificial information language. It

also mentions some of the statistical operations, such as correlations and trend analyses, that can be executed.

Section VI details the key problems of inference: the different kinds of inference that can be made, what role they play in a data retrieval system, and what problems must be solved in order to mechanize inference-making. A key feature of the system is the ability of the user to frame an inference schema and instruct the machine to execute it. This technique allows the man and the machine to collaborate in deriving a conclusion required to answer a given request for information.

Sections VII through IX describe techniques for practical realization of the data file. A major problem area is at the input end of the system, where data must be acquired, organized, and preprocessed before it can enter the computer. Section VII outlines resolutions of those problems of data sources, acquisition, extraction, etc. Input aides extract data from source documents and enter them onto forms according to precise rules. The computer then translates from forms to relational language sentences. This section discusses the motivation and structure of the input forms.

Section VIII deals with the output problems. The sources and types of output requests that the file will be expected to satisfy are described first. Then the problem of communicating those requests to the file and the procedures for processing them are discussed.

Section IX treats some of the storage and processing problems; e.g., how data will be structured in memory, the organization of dictionaries and thesauri, and the steps in the programming process.

Sections X through XII consider the data file as a whole, examining its principal features, application, possible extension, and relation to other data retrieval research.

Section X reviews the question of literature searching and shows how the data retrieval system embodies features that permit very effective literature searching. It summarizes the hypothesis that context data of the type this system is designed to handle offer a promising approach to more effective literature searching.

Section XI presents some of the next steps that will be undertaken to extend the capability of the system.

The final section summarizes and evaluates the work and contrasts it to other activities in this field.

ACKNOWLEDGMENTS

The early ideas and plans behind the research described in this Memorandum were generated by the two authors, but the study has been enlarged and enhanced by others who have since joined us. The notions developed here have ramifications in several directions and we have been fortunate to have competent colleagues who are participating in what is now a joint effort. We are deeply indebted to Don Cohen and Gerald Levitt for their excellent execution of the enormous programming tasks. The logical and linguistic problems at the core of the research are complex, and we have been fortunate to have the constant and competent collaboration of J. L. Kuhns. Finally, we are grateful for the diligence of Wade Holland and his able assistants in handling the severe input problems.

CONTENTS

I. INTRODUCTION

INITIAL REMARKS

The purpose of this Memorandum is to describe the status and direction of a current RAND project dealing with the problems of automatic information processing and data retrieval. The work clusters around the problems involved in storing factual information in a computer in the form of basic sentences, and logically operating on those sentences as needed in order to respond appropriately to a request for information. In a most general way, this type of computer application deals with automatic language data processing. It concerns the mechanization of those logical techniques required to operate on language so as to derive and retrieve answers to questions. Within the general field of automatic language data processing, one could characterize this work under the heading of automatic data retrieval or automatic question answering. However, the objectives go beyond the simple retrieval of data and direct questioning, and include automatic literature searching as well. Furthermore, in order to mechanize those processes required for deeper data retrieval and more complex literature searching, this work extends into the field of automatic inference-making.

AN EXPERIMENTAL CORPUS

Logical techniques for data retrieval and inference-making cannot be evaluated in isolation. Any realistic determination of their effectiveness would demand that the techniques be evaluated by putting them into practice, by

empirically testing an actual corpus of data. The subject matter selected for the experimental corpus was the context of cybernetic research. Naturally, key sources of information about the context of cybernetics are professional books and journal articles. These materials give information about who is doing what, and which papers are about what subareas of cybernetics. Thus, the system contains data at two different logical levels: a) about people, places, committees, etc.; b) about information--e.g., that paper P was indexed under category C. This distinction is necessary to further clarify where this work fits in the general field of automatic language data processing.

LITERATURE SEARCHING VERSUS DATA RETRIEVAL

Mechanized literature searching is the problem of automatically answering the following kind of question: What publications (books, papers, reports, documents, etc.) discuss the subject of logic machines? Mechanized data retrieval, on the other hand, is the problem of automatically answering questions of the following kind: How many keys were there on the original logical piano of Jevons? In the case of literature searching, the appropriate response by a mechanized system is a collection of relevant items or reference to such a collection. (Relevance is the key concept in any theory of literature searching.) In the case of data retrieval, an appropriate response is the answer to a given question.

Literature searching does not directly answer a question; rather, it provides the answers to a secondary question by putting the requester in touch with data which, in turn, will

satisfy his information need. Data retrieval, on the other hand, answers questions more directly. In the present system, which deals with two levels of information, the data retrieval function includes literature searching (Sec. X discusses the details of this interrelationship).

SOME OTHER DISTINCTIONS

A major point of departure in the design of this system concerns the nature of the language used to store the factual data: Instead of using natural (i.e., ordinary) language, this system uses a formalized language based on the calculus of relations. All elementary facts are described in this language in terms of the relationship between two entities. More complex sentences are reducible, according to strict rules, to basic relational sentences. This language permits precise selection and recall to interrogation because its grammatical form causes no vagueness or ambiguity. In data retrieval systems employing natural language, data (i.e., sentences whose "contents" allege to answer a given question) are selected by matching words in the sentences with words derived from the request. In the present system, the basic selection takes place by matching sentences, i.e., by matching words plus syntactic structure. The use of a formal language, and not ordinary language, marks one of the key distinctions between this data retrieval system and some others.*

*Simmons has surveyed a variety of question-answering systems and the reader is referred to that survey for a comprehensive overview of systems related to the one presented here [1].

A major problem in the design of any automatic data retrieval system concerns inference; i.e., how to derive and make explicit unexpressed data needed to answer a given question. The precise formal language facilitates generating routines to aid in the automatic derivation of implied conclusions. Thus, this system aims beyond simple data retrieval toward inference-making techniques; this key feature is another point of departure separating this system from some other data retrieval systems.

DIRECTION AND STATUS

The primary problems are of a logical and linguistic nature, dealing with the structure of the basic information language and the nature of inference. The linguistic problem appears at both "ends" of the system; i.e., at the user end where a request is formulated, and at the input end where the data is "translated" from its primary sources to its representation in this system. These problems, of course, are logical as well as linguistic, since they intersect with those growing out of the structure of the formal information language. Clearly, a key logical problem surrounds the entire notion of mechanizing aspects of inference-making.

The mechanized data retrieval and inference system uses a conventional digital computer. Therefore, the problems do not concern engineering--only the software or programming routines. Since we are evaluating techniques in terms of how they actually work, we have started to collect and organize a sizable corpus of data on the context of cybernetics. A rather substantial data base has

been already extracted and keypunched, and although the programming aspects are sizable, the implementation of a fully programmed system is well under way. Preliminary print-outs listing any selected aspect of the input data can already be conducted. The logical rules required to analyze requests and transform them into search instructions have been partially completed.

II. THE NOTION OF AN EXPERIMENTAL CORPUS

MOTIVATION

If information retrieval (literature searching, data retrieval, etc.) is to become more a science than a collection of techniques, work in this field must be experimental as well as theoretical. Ideas and theories must be tested to determine whether they can survive. Furthermore, experimental testing of competing theories must be conducted in order to compare and rate them. Just as the theoretical physicist must test his hypotheses against the observations obtained by his experimental colleagues, so also must workers in information retrieval test their ideas against the results of application in an information system.

For these reasons, among others, we decided to search out, collect, organize, and use an actual corpus of data upon which to test our ideas concerning data retrieval and inference-making. The experience of working with an actual corpus has another advantage which should not be overlooked: entailed problems which might not otherwise have been recognized and attacked are uncovered and solved.

THE QUESTION OF CRITERIA

By what considerations should one be guided in searching for a suitable body of data to serve as the experimental corpus? A very practical consideration suggests that, since a fair amount of time and effort must go into collecting and organizing the proposed data bank, the information in question should have some value independent of its function as a purely experimental corpus. Thus the data should be

neither synthetic (i.e., artificially constructed), nor vacuous. Furthermore, although the corpus should be manageable from a logical and linguistic point of view, also it should provide a good sampling of the kinds of problems one might expect in a real-world situation. The data must be accessible so that one may reasonably amass a critical size; it must involve enough basic sentences to give significance to the tests. For all of these reasons, the experimental corpus was of the kind characterized above, on the context of research on cybernetics.

THE SUBJECT OF CYBERNETICS

What is cybernetics? In 1948 the late Professor Norbert Wiener published his now famous book, Cybernetics: Or Control and Communication in the Animal and the Machine. In the book, he explains that he needed a term to denote the broad subject of information processing in complex systems and that is what "cybernetics" means. Cybernetics is the broad study of information processing, communication, and control. It includes all those theoretical notions, techniques, procedures, devices, and machines that relate to information processing, communication, and control, as well as the application of information, communication, and control theories and techniques to the synthesis of artificial machines, to the analysis of natural nervous systems, and even to society itself (which can be viewed as a complex information and control system).

Cybernetics is a complex and growing subject, and even the experts do not agree on its exact nature, scope, and status. However, for the present purposes, cybernetics is

interpreted in its broadest sense (as do the Soviets) as the collection of studies and techniques that cluster around information, communication, and control. These include: information theory and applications; mathematical logic and its applications to computers and automata theory; computer design, organization, components, and applications; artificial intelligence; information-processing aspects of biology, psychology, and physiology; control theory and applications; mathematical economics and operations research.

SOME FURTHER REMARKS

It is important to make explicit and sharpen the distinction between data on cybernetics and data about the context of research in the field of cybernetics. This distinction can be clarified by considering the analogy between the work of a physicist and the work of a sociologist of physics. The former studies particles and waves and tries to find and test hypotheses about the behavior of such physical entities. The sociologist of physics, on the other hand, does not look at physics as such but at the physicists and at what they do. He is interested in how they communicate with one another, that they write and review papers, give presentations, win honors and professional recognition, experiment on linear accelerators, belong to special organizations, have an inclination toward music, etc. Similarly, the laws, methods, results of cybernetics are not important here; the primary interest is the context within which such work is carried on: people who do work on cybernetics, the computers they use, the institutes with which they are affiliated, the journals

they publish in, the descriptive categories under which their papers are indexed, and so on. The context of cybernetics research and not the content is the subject of the experimental corpus. Thus, the system is designed to respond to inquiries of the following sorts:

- Who is the author of Information Theory in Biology?
- Where is Professor F. G. George located?
- What papers (on cybernetics) have been jointly authored by a logician and a neurophysiologist?
- Where is research on pattern recognition going on?
- Which papers have cited MacKay's paper which was presented at the 1959 Teddington Conference?

III. LOGICAL AND LINGUISTIC PROBLEMS

SOME INITIAL DISTINCTIONS

The problem in its simplest form is to describe certain facts (e.g., that a certain person is a member of the editorial board of a professional journal that publishes papers primarily in the field of cybernetics) and then store the corresponding information in a computer so that it can be processed in various ways. It is important at the very outset to be clear about the set of facts (situations, events) which are to be described and how the corresponding descriptions are to be worded. The facts are those non-linguistic entities having to do with people, publications, institutes, etc. The language, on the other hand, consists of those linguistic entities (such as words, sentences, etc.) and their rules that we use to describe the facts. Following Uspenskii, this language is called the information language;* it should not be confused with either the programming language or the so-called "machine language."

THE CHOICE OF AN INFORMATION LANGUAGE

Many of the critical logical and computer problems concerning how the stored information is to be processed hinge on the structure of the information language selected. Therefore, we must examine those criteria of adequacy that

*Our discussion of an information language and of its role in the design of an automatic information processing system has been influenced by the paper of V. A. Uspenskii [2].

an information language must satisfy in order that our choice is well guided.

Clearly, the information language must be rich enough to allow us to describe all relevant aspects of the facts. It must contain the right kinds of both logical and content-bearing words, and its grammatical rules must be flexible enough to permit the construction of sentences of sufficient generality for adequate description of the facts. Its rules must be precise and the terms of its vocabulary sufficiently well explicated so as to avoid ambiguity. Finally, because current computers cannot comprehend language as do humans, the information language must be formalized; the desired processing must be describable in terms of its logical syntax. If one can spell out the desired form of information processing in sufficient detail and in a precise and unambiguous way, then that procedure can be embodied in the form of a computer routine and executed automatically. Furthermore, a formal description of the language is demanded not only for precise retrieval, but, more importantly, for the inference-making which is to be incorporated in the system (as will be described in more detail subsequently).

NATURAL LANGUAGE AS THE INFORMATION LANGUAGE

Ordinary (natural) language has much richness, flexibility, and versatility; however, it suffers from its lack of precision. That is, ordinary language has no explicit formation rules that enable one to characterize precisely sentences in terms of their syntax. And, of course, ordinary language has no explicit transformation rules.

Sentences in ordinary language can be ambiguous; meaning depends on context. Conventional computers must be instructed on how to process data in terms of a syntactical description of the linguistic entities in question. Can natural language be formalized? Uspenskii [2] suggests that, in principle, it can, but that for all practical purposes its complexity makes the prospect of formalization essentially hopeless. This leaves as the alternative the use of an artificial language as the information language.

AN ARTIFICIAL INFORMATION LANGUAGE

First of all, what is meant by an "artificial information language?" There must be precise rules that prescribe how sentences of the language may be formed in order to be grammatically correct. There must be transformation rules and truth rules. The terms of its vocabulary must be precise and there must be rules that describe which terms are synonymous, etc. As Uspenskii [2] correctly points out, one must be guided in the construction of an artificial information language by first making an analysis of those facts (situations, events) which are to be described by the language in question. Let us now look more closely at the "universe" which we propose to describe--at the class of things and their relationships with which we are concerned.

We are interested in the people who are active in the field of cybernetics and the institutes, universities, or laboratories with which they are affiliated and the locations of those institutes. We are interested in the papers

which they publish in professional journals and which they present at professional society meetings, conferences, and symposia. We are interested in the specific cybernetic subject categories under which their papers are indexed and the computing machines which are used to help them with their research. And so on. Given classes of people, papers, institutes, journals, meetings, cities, subject categories, etc., the basic data can be described in terms of the relationships between members of these classes.

Some symbolism and notation aids discussion of the information language created to describe these classes and their relationships. Upper case letters of the alphabet denote classes and lower case letters with subscripts denote individual members of the class. The following is a list of classes:

A = People,
B = Professional papers,
C = Books,
D = Organizations (universities, institutes, etc.),
E = Computing machines,
F = Professional, honorary, or government societies, committees, or councils,
G = Professional journals,
H = Subject index tags (within the general field of cybernetics),
J = Professional meetings, conferences, symposia,
K = Geographical locations (cities, etc.),
L = Professional awards, prizes, honors, etc.

The upper case letter "R" denotes relationships. A partial list of relationships follows:

R_1 = AUTHOR OF
R_4 = AFFILIATED WITH

R_7 = HAS PROGRAMMED
R_{14} = ON THE EDITORIAL BOARD OF
R_{18} = PRESENTED PAPER AT
R_{22} = LOCATED AT
R_{25} = CITED IN
R_{27} = INDEXED UNDER
R_{34} = PUBLISHED IN

Since we wish to use knowledge of these relationships to make inferences, it is natural to select a relational formal language as the information language. We refer to this as the "Relational Information Language." This artificial information language is an interpreted relational calculus whose atomic sentences are of the following form:

$$a_{21}R_{14}g_6 \ , \qquad (1)$$

which might state "A. B. Smith (a_{21}) ON THE EDITORIAL BOARD OF (R_{14}) Cybernetics (g_6)."

Relevant information will be mapped into the Relational Information Language by humans (but with computer assistance, as described in Sec. VII). And, of course, other data will be associated with the basic (atomic) relational sentences (e.g., source of information, dates, etc.).

Does the Relational Information Language have the richness required so that logically complex expressions of ordinary language can have an equivalent counterpart in the artificial information language? If the subjects were on cybernetics instead of information about work being done on cybernetics, it would be quite difficult to construct a

sufficiently rich information language. But given the more limited objective, viz., to store and process information about people, places, papers, machines, institutes, meetings, etc., and their relationships (as listed above), the prospect of success is rather high.

The information language is an interpreted relational calculus. The calculus of relations is that chapter of modern logic which deals with relationships and rules for deriving new relationships from other given relationships. New relations may be obtained by definition and by derivation. There will be logically strict derivations of new relationships and, of most value, plausible derivations when the relationships between relationships are a function of their content and not merely their form. The following three sections detail these ideas.

IV. THE LOGIC OF RELATIONS

The theory of relations is that chapter of modern logic which deals with relations, their properties, and with the relationships that hold among relations. Since this Memorandum uses its notions and techniques and applies them to work on data retrieval and inference-making, this section outlines some of its key concepts.

In everyday life, as well as in mathematics, one is confronted with a variety of relations. Things are located in space, giving rise to spatial relations; events are related temporally; there are causal relations, kinship relations, ordering relations, etc.

Relations can be classified according to the number of things that they relate. For example, the relation TEACHER OF holds between pairs of individuals and is called a dyadic, binary, or two-place relation. The statement "x GAVE y TO z" involves a relationship among three things; it is called a triadic relation. An example of a four-place (or tetradic) relation is "x WAS TEACHER OF y DURING SEMINAR ON z AT UNIVERSITY u." One can, of course, get relations of any number of places.

We shall be concerned primarily with two-place relations and it is conventional to symbolize such instances of relations in either of the following two ways:

$$R(x_i, y_j)$$

$$x_i \; R \; y_j$$

We say that x_i is a predecessor of y_j with respect to the relation R. And y_j is a successor of x_i relative to the relation R. The clas. of predecessors is called the domain of R. The class of successors is called the range or converse domain of R. The union of the domain and the range is called the field of R.

Every relation has a sense or direction; i.e., it holds from the predecessor to the successor. Now, when a relation R holds from x_i to y_j, some relation also holds from y_j to x_i. This latter relation is called the converse of the former relation and it is denoted as $\breve{R}$. Thus, $\breve{R}$ holds between y_j and x_i if and only if R holds between x_i and y_j. For example, consider the following binary relations and their converse relations:

AUTHOR OF	AUTHORED BY
TEACHER OF	STUDENT OF
CITED	CITED BY.

REPRESENTATIONS OF RELATIONS

Relations can be represented in a variety of ways. These alternatives allow elaboration on some key notions in the theory of relations and are relevant to the computing aspects of the data retrieval problem.

First, notice that just as mathematical functions can be represented by tables of values, so can relations such as HUSBAND OF be represented by the set of all pairs of husbands and wives. Such a list is called the extension of the relation. Alternatively, a directed graph or arrow diagram is a mode of representing relations. This is a configuration of arrows and nodes in which the nodes

represent individuals of the field and the arrow connecting nodes represents the holding of the relation for that pair. A relation has a sense or direction (viz., from predecessor to successor), here represented by the arrow. Again, the domain element is the node at the tail of the arrow and the range element is the node at the head end of the arrow. As an illustration, consider the following example dealing with the relation AUTHOR OF. Assume that there are four individuals (authors), Jones, Smith, Robinson, and Martins, denoted as a_1, a_2, a_3, and a_4, respectively. Assume further that there are papers denoted as b_1, b_2, and b_3. Jones is an author of b_1 and of b_2; Smith is an author of b_2, and Robinson is an author of b_2. Finally, Martins is the author of b_3. We can symbolize this information and represent it as the following list of instantiations of R:

$a_1 \; R \; b_1$

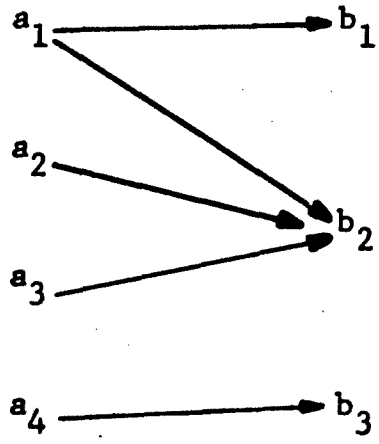
$a_1 \; R \; b_2$

$a_2 \; R \; b_2$

$a_3 \; R \; b_2$

$a_4 \; R \; b_3$

In terms of an arrow diagram, it appears in the following way:



A third way to represent relations is in terms of Boolean matrices of ones and zeros. Enter a 1 in the i^{th} row of the j^{th} column if and only if the relation holds between the i^{th} member of the field and the j^{th} member of the field; otherwise, enter a 0. The Boolean matrix that corresponds to the arrow diagram above is as follows:

	a_1	a_2	a_3	a_4	b_1	b_2	b_3
a_1	0	0	0	0	1	1	0
a_2	0	0	0	0	0	1	0
a_3	0	0	0	0	0	1	0
a_4	0	0	0	0	0	0	1
b_1	0	0	0	0	0	0	0
b_2	0	0	0	0	0	0	0
b_3	0	0	0	0	0	0	0

DERIVED RELATIONS

Just as new relations can be derived from ones that are already defined, as for example in terms of their converse, so also new relations may be generated by combining other relations. Consider as an example the case where x is the brother of u and u is the wife of y. Of course, given the above, a relation holds directly between x and y; namely, the relation of brother-in-law. Now the relation that holds between any x and y, if there is a u such that "xRu" and "uSy", is called the relative product of R and S. The relative product of R and S is denoted as "R/S." Thus the relative product of BROTHER OF and SPOUSE OF is the relation BROTHER-IN-LAW. Following logical notation, write R^2 as an abbreviation for R/R, R^3 for R^2/R, etc. These relations are called the (second, third, etc.) powers of R.

The operation of relative product is quite valuable in the calculus of relations. We can indicate one aspect of its usefulness by the example described earlier.

R denotes the relation AUTHOR OF and $\breve{R}$ denotes its converse AUTHORED BY. We now take the relative product of R and its converse, i.e., $R/\breve{R}$, which represents the new relation CO-AUTHOR WITH, where each a is considered to be co-author with himself. If we represent the relationships between authors and papers in terms of a Boolean matrix, multiplication of the R matrix and the $\breve{R}$ matrix generates the new matrix $R/\breve{R}$ CO-AUTHORED WITH.

V. INFORMATION PROCESSING FOR DATA RETRIEVAL

Section III described the role of the information language in the design, organization, and operation of a data retrieval system. Section IV described the logical structure of the calculus of relations, which provides the form of our information language. This section assumes that input data has been acquired, and processed (both manually and by machine) to the point where it has been completely and correctly mapped into the Relational Information Language. (Section VII gives some of the relevant details of this "translation.") The purpose here is to describe, from a logical point of view, the types of operations that can be executed and how the processes are effected given an information language based on the calculus of relations.

VERIFICATION

One type of request is to ask for some sort of verification, as when a user wants to know whether a given sentence is true. For example, he wants to know whether A. B. Smith is on the editorial board of Cybernetics; he expresses this request in the form of the relational sentence "$a_{21}\ R_{14}\ g_4$." (By means of dictionary, thesaurus, etc., the translation is made from "A. B. Smith" to "a_{21}," "Cybernetics" to "g_4," and "on the editorial board of" to "R_{14}.") If this data has entered the system at the input end, then a "copy" of that sentence ("$a_{21}\ R_{14}\ g_4$") is stored somewhere in the computer memory. Thus the computer routine needed to answer the request entails a search that attempts to match

the request sentence with those data sentences previously stored.

Notice that the exact nature of the routine hinges on the way in which relational sentences are represented within the computer. That is, if relations are represented as pairs, listed according to the relations in question, then a simple search of the corresponding list will locate its desired sentence. If, on the other hand, relations were represented in terms of Boolean matrices, then the search routine would simply go to the R_{14} matrix and checks to see whether or not there is a 1 at the 21st row and the 4th column. If so, this means, of course, that "$a_{21}\ R_{14}\ g_4$" is true. But, from a logical point of view, the routine required to decide whether or not a given sentence is true simply amounts to a search of the stored data.

LIST VALUES OF ONE VARIABLE

Another type of request is the case where a user asks for the name of a predecessor or a successor for a relation, or for the relation itself. For example, a user wants to know who is on the editorial board of Cybernetics. Assume that this request takes the form of "$x\ R_{14}\ g_4$." The retrieval routine responds to this type of request by searching to find all x's that are in the relation R_{14} to g_4. Instead of searching for a match of the entire relational triplet as for verification, this case demands a match only in part and a readout of the other part. Other

examples of this type of request are "a_{21} R_{14} y" and "a_{21} R_i g_4"; i.e., for what journals is A. B. Smith on the editorial board and what relations hold between A. B. Smith and Cybernetics? In each of these cases, one and only one variable in the relational triplet expresses the request.

In all three cases, the computer routine attempts to match those parts of the input query which are fixed (constant). And, of course, if the relations were represented in terms of matrices instead of lists of sentences, the search would be comparable to looking in a column (or a row) to see whether or not any entries had been made in a given matrix.

LIST VALUES OF TWO VARIABLES

The next most general retrieval request contains two variables in the relational triplet. For example, a user asks to find out about all the relations that hold for a given individual. Denote this as "a_{26} R_i y"; i.e., read out all the sentences where a_{26} is the domain element (or the range element). Another example might be denoted as a_i R_5 a_j; viz., tell me all the pairs of individuals for whom the relationship TEACHER OF holds. The computer would search and print out the corresponding list of ordered pairs that satisfy the relation R_5 (teacher of).

All the above groups of retrieval routines are trivial once both the input data and the user's request have been mapped into sentences of the Relational Information Language.

Incorporation of more selective retrieval can be made by introducing truth functional connectives as part of the

"grammar" of a request. In this way a user could, for example, interrogate the system and request a listing of all individuals who have either relationship R_5 or R_8 and also R_{19} to some other given item. Such request sentences can become rather complex, containing many variables and logical connectives.

DATA MANIPULATIONS

With data precisely identified and available for recall, a natural step is to make counts of retrieved data to obtain statistical information about work being conducted on cybernetics. For example, the computer could be programmed to execute simple routines to answer the following kinds of questions:

How many a_{27} R_1 b_j?

(How many papers did A. B. Smith author?)

How many b_6 R_{50} b_j?

(In how many papers was the paper b_6 cited?)

LOGICAL PROCESSING

A next step in the processing of data from a file of this sort would be toward checking, detecting, and calling attention to redundant information. In a sense, the opposite of redundancy is inconsistency; the system should be programmed to detect and call attention to inconsistencies

contained in the explicit data. For example, if one sentence says that A. B. Smith is the managing editor of Cybernetics and another says that J. K. Jones is the managing editor of Cybernetics, are these necessarily inconsistent? It depends. Do these refer to the same time? Are there two journals with the same name? Can a journal simultaneously have two distinct managing editors? Different kinds of information are necessary to answer these questions which, in turn, must be processed to decide questions of inconsistency. In principle, we will be able to handle these situations, but as yet no such consistency routines have been written.

VI. INFERENCE MAKING FOR DATA RETRIEVAL

THE PROBLEM OF IMPLICIT DATA

Assume that the following information has entered the system and is now stored in memory: "A. B. Smith is the author of 'On Cybernetics'," and "'On Cybernetics' is indexed under pattern recognition techniques." These two sentences can be symbolized in information language as follows:

$$a_{21}R_1b_{50} \qquad (1)$$

$$b_{50}R_{27}h_{17} \ , \qquad (2)$$

where R_1 = AUTHOR OF and R_{27} = INDEXED UNDER.

If the above two sentences are true, then A. B. Smith has published on the subject of pattern recognition. This fact can be denoted as

$$a_{21}R_{46}h_{17} \ , \qquad (3)$$

where R_{46} = PUBLISHED ON. The system explicitly stores the information represented by sentences (1) and (2). Sentence (3), on the other hand, is only implicit in the data store. Thus, although logically contained in (1) and (2), it can be made explicit only through some process of logical derivation.

The reason for making this distinction between information that is explicit and information that is only implicit is to point out the important role of logical derivation in

any question-answering (or data retrieval) system. Consider, for example, the user who wants to know whether A. B. Smith has published on the subject of pattern recognition (viz., is "$a_{21}R_{46}h_{17}$" true?). If the routine for answering this query merely searches the contents of memory in an attempt to match the request sentence against those already stored, clearly the system cannot provide the correct response. In order to respond appropriately, the system must incorporate routines and procedures to make explicit the required information.

To repeat: We will be accepting and storing large amounts of data extracted from publications in the field of cybernetics. Those data will be formulated as sentences in the Relational Information Language. Clearly, numerous sentences that are not contained in memory at a given time must be true, given the truth of the ones that are contained in the file. The ideal data-retrieval system, when unable to answer a query by any of the stored explicit input sentences, will seek out an answer implicit in the stored data. In a logical sense, the system should employ the technique of inference or logical derivation. How, then, can inference be mechanized?

STRICT INFERENCES

Logic is the study of inferences of two kinds: strict or deductive inferences; and plausible, probabilistic, or inductive inferences. Consider first the problem of mechanical (and other) procedures employed to derive strict inferences.

Of course, certain logical languages have procedures for mechanically generating all strict (logical) consequences of a given set of initial premises. However, unless certain complex rules are constructed which enable only the important or relevant conclusions to be derived, these procedures will continue to produce all implied conclusions--most of which will be trivial and uninteresting. However, in the case of the Relational Information Language, no such general mechanical decision procedures exist. In the face of this "impasse," one alternative is to select in advance and program accordingly certain inference schemata that can be used as needed. In order to explain this notion of inference schemata which are pre-selected, consider the following example: Let a relation R be transitive, i.e., the following is logically true:

For all a, b, and c, if aRb and bRc, then aRc. (4)

This can be symbolized as

$$(a)(b)(c)\ [(aRb)(bRc) \supset (aRc)]\ .$$

This inference scheme represents a strict inference, because the conclusion (aRc) is logically implied by the premises (aRb), (bRc). Assume now that a user interrogates the system and asks whether "aRc" is true and that the sentence is not explicitly stored. If the system had the information that R is transitive, then it could form the inference schema (4), search its memory for the antecedent

sentences (aRb) and (bRc); and, if it found them, it could print out that the request sentence "aRc" is also true.

Two points are important: First, by knowing the logical properties of some of the relations contained in the information language, the system can be programmed to construct certain inference schemata (like (4) above). These schemata could then be used regularly to make explicit at the input end data which otherwise would be implicit. This means that the system selectively derives strict conclusions from incoming data and stores these conclusions explicitly in memory, ready for subsequent interrogation. Second, the user himself could form such inference schemata as needed in each individual case, in order to have the machine derive a conclusion in response to his request. That is, if a sentence he desires (such as $a_{55}R\ c_{16}$) is not stored explicitly, he can instruct the system to execute an inference schema which he offers it in order to see whether the desired conclusion can be logically derived from stored data (i.e., "$a_{55}R\ b_j$" and "$b_jR\ c_{16}$"). (The next section elaborates on the role of the user in constructing inference schemata.)

PLAUSIBLE INFERENCES

A "plausible" inference is the transition from premises which are true to a conclusion which is only plausible. Again, in the case of strict inference, if the premises are true the conclusion must be true, because logically the conclusion is completely contained in those premises. In the case of a plausible inference, on the other hand, if the premises are true the conclusion need not be true;

i.e., the premises only partially contain the conclusion. The truth of the premises of a plausible argument confers a degree of partial truth on the conclusion.

A schematic example will clarify the logical status and use of plausible inferences. Consider the following expression:

$$\text{If } (aR_1b) \text{ and } (aR_{71}c) \text{ and } (bR_{19}d) \text{ and } (dR_{66}e),$$
$$\text{then it is plausible that } (eR_{55}c). \qquad (5)$$

The person who asserts (5) believes that given the truth of the sentences in its antecedent (i.e., the so-called premises), the consequent of (5) (the conclusion) will probably also be true. The conclusion need not be true; in fact it could be false for a given instance of a, b, c, d, and e. When an individual asserts a statement like (4), he has formalized an empirical generalization which may lead to false conclusions as well as to true ones.

In such schemata, the degree of plausibility is, of course, relative to the individual who asserts it. Consider how such schemata could be used with our data retrieval system to derive plausible data that are not otherwise explicitly available. A user approaches the system to inquire whether it is true that "$e_{15}R_{55}c_5$." That sentence is not explicitly stored and consequently the computer cannot respond affirmatively. The user now realizes that if the sentences constituting the antecedent to (5) are true (i.e., if they are stored explicitly), then the conclusion is plausible. He constructs the

specific inference schema corresponding to (5) and the machine executes it to see whether or not its premises are true. That is, the machine searches its store to find an a, b, and d such that

$$(aR_1b) \text{ and } (aR_{71}c_5) \text{ and } (bR_{19}d) \text{ and } (dR_{66}e_{15})$$

is true. If so, the user now knows something about the probable truth of "$e_{15}R_{55}c_5$" which he did not know before.

Thus, by allowing the user to construct plausible inference schemata, and programming the system to execute them upon command, it is possible to derive plausible conclusions, i.e., conclusions whose truth is only partially relative to the truth of the stored data.

Notice also how this may be generalized. Instead of asking whether "$e_{15}R_{55}c_5$" is true, the user may ask: For which e does R_{55} hold with c_5? The machine now simply looks for sets of a, b, c_5, d, and e for which the relations expressed in the antecedent hold.

Two additional problems are: first, how to interpret the notion of degree of plausiblity; and then, given any interpretation, how to estimate the <u>degree</u> of plausibility. For the present purposes, a subjective interpretation to the concept of degree of plausibility is sufficient; it reflects the strength of psychological belief on the part of the user, or the kinds of odds he would want if he were to bet on its truth. Since the individual who constructs the inference schema will be the one to use the resulting information, it is not necessary to provide an interpretation that is objective, i.e., binding on all users of the

system. A different user might use the same inference schema and assign a different degree of plausibility to it. Since we are suggesting a completely subjective interpretation for the concept of plausibility, its degree (amount, value, etc.) can be estimated intuitively by the user himself.

Finally, notice that for many kinds of plausible inferences, the degree of plausibility is a function of the time intervals during which the events described in the premises occurred. Consequently, those temporal data also should be included as part of the premises proper. Since we are capturing temporal data in relation form, this is no problem in principle; at worst it is a further complication that the user must deal with in constructing a plausible inference and in estimating the degree of plausibility to be associated with it.

VII. INPUT PROCEDURES

THE PROBLEM

The previous sections concerned the theoretical foundations for a practical data retrieval system. This section begins consideration of procedures and techniques.

The first problem encountered in practice is that of providing data to the data file--the input problem. To solve it, procedures must be developed for gathering data sources, for extracting data, and for entering it into the file.

Types of Data

The types of data included in the cybernetics data file may be characterized in two ways: by content and by form.

The Content of the Data. The content of the data file includes both foreground and background information. The foreground includes assertions about the context of cybernetics research; sentences about the relations among persons, organizations, publications, research areas, positions, and so on. Yet many analyses of such foregrounnd data will demand the use of background information about subjects not contained within the context of cybernetics research. For example, geographic information about the location of cities in states and states in regions would probably be required in order to establish what organizations in the eastern United States are supporting research on pattern recognition.

The Form of the Data. The data in the file assume three forms: relational sentences, derivation rules, and thesaurus entries. The basic datum is a primitive sentence in the Relational Information Language; that is, an assertion of the form aRb, "entity a is in relation R to entity b." In a number of cases, however, an abbreviation in file volume can be achieved by storing instead a set of rules that will allow the derivation of a specific sentence when it is required. These are the built-in inference schemata explained in Sec. VI. Data appearing in the file as entries in a thesaurus are assertions of the semantic equivalence of two or more expressions: for example, the assertion that "MIT" and "Massachusetts Institute of Technology" are different names for the same object.

Sources of Data

As Sec. II explained, the major sources of data on the context of cybernetics research are publications, both scientific and general. In addition, information comes from personal knowledge.

Collection Procedures

Two properties of the collection procedures deserve mention. First, the procedures are source responsive. Data are collected and extracted as they become available in a particular source and not as they are needed to respond to a particular query. Second, the sources are exploited in separate passes, each pass searching for a different type of data.

For example, a scientific or technical publication is exploited in three passes. During the first pass, bibliographic data concerning the subject publication are collected. During the second pass, data about the publications cited by the subject publication are extracted. In the third pass, the text of the publication is scanned for data about the context of cybernetics research, such as statements that note the author's indebtedness to others, that tell where the research was conducted and its present status, that identify the research sponsors, and so on.

EXTRACTION FROM DATA SOURCES

Since data appear in the sources in diverse forms and at low density, complete input mechanization is infeasible at present. Humans must identify the appropriate data and present them to the machine in a precise, standard form. Two questions arise: First, what should the precise, standard form be? Second, how can data so expressed be translated into the information language for storage?

Representation of Extracted Data

There are three alternatives for representation of extracted data: the input personnel could extract data from the sources and represent it directly in sentences of the Relational Information Language; they could represent the data in English language sentences; or they could employ a special notation developed for the input process.

Data entry in the Relational Information Language would eliminate the need for subsequent computer translation; however, considerable human effort would be necessary

to identify, record, and check the large number of sentences needed to record even basic data; the probability of mistakes and omissions would be high; and changes in the information language would necessitate re-extraction of the data from original sources. Data entry in the English language would be convenient for human input aides, but inconvenient for the computer. Translation into the Relational Information Language would require routines whose design would be a considerable research project.

Therefore, a special notation has been developed for entry of data into the Relational Data File. The input aides express the data in this notation and the computer translates from the notation to the information language. Changes in the information language demand only a change in the translation program and a new computer translation of the input data--the input data do not have to be collected anew.

Data Input Forms

The notation that has been adopted has five major properties.

1) The input aides enter all data onto forms.
2) A specific form is designed for each class of data encountered at the input. Forms to collect bibliographic, organizational, and citation data are among those needed for the cybernetics corpus. The specific design of such forms must be done anew for each new subject corpus, although many forms will serve several corpora.
3) All classes of forms for all corpora are structurally similar, so that a single set of input programs can be written to handle input forms of

any class. Thus, the basic programming has to be done only once; and the design of new forms and application to new subject corpora does not require any change in programs. The form designer has only to provide the input program with the values of certain parameters for each specific class of forms.

4) Each separate item of information on each form is explicitly named, variable in length, and atomic. That is, the name of each primitive datum enters the computer along with the piece of information; the entire entry is bounded by parentheses so that it can be of any length; and the string of symbols that constitutes the entry is always treated as a whole, never broken into constituent words or phrases. Consequently, design of forms is simplified, since fixed-length areas do not have to be established for certain kinds of data, and new entries can be inserted without redesigning the entire form. And similarly, the programming of operations for handling forms is simplified, since all the data are explicit in the input and not implicit in the location of an item on a card or tape.

5) Within a form, all the primitive data related to a specific aspect of the subject item are grouped together. For example, the bibliographic data form groups its primitive data items into six broad categories concerning the subject publication. A bibliographic data form is filled out for each publication; it contains information relevant to as many of the six broad categories as are present in the particular source.

Bibliographic Data Form

Each bibliographic data form is composed of a set of data units called vectors which may be any one of six vector types, one for each aspect of the data. The six types and the aspects of the data that each is intended to capture are as follows:

- SOURCE--the input aide's actual source of the data entered on the remainder of the form (e.g., an abstract or the item itself);
- PUBLICATION--the immediate publication context of the subject item (e.g., the journal in which an article appears);
- ITEM--the subject of the form (e.g., an article, book, or collection);
- PERSONNEL--the individuals associated with the subject item (e.g., authors, editors, translators);
- PRESENTATION EVENT--any event or meeting at which the subject item was presented (e.g., conferences, seminars, classes);
- ORGANIZATION--the organizations associated with the subject item (e.g., sponsors, group authors, contractors).

For convenience, the vector types have been given literal names: A = Source, B = Publication, C = Item, D = Personnel, E = Presentation Event, and F = Organization. A form may contain more than one vector of a particular type; a vector is filled out for each separate entity of a type. For example, if a publication has three authors, three personnel-type vectors will be entered on the form referring to that publication. Each vector is given a name formed from the name of its type and an index. The three personnel vectors, for instance, would be called D1, D2, and D3. A form need not contain vectors of every type, but a bibliographic data form must contain a Source, a Publication, and an Item vector.

Each vector consists of a set of primitive data units called components. Each component has the form

(nn...n = xx...x)

where "nn...n" is the name of the component (e.g., Publication Date) and "xx...x" is a string of characters, the datum (e.g., August 1965). For brevity, the components of the bibliographic data form vectors have been given numerical names, for the most part. For example, each personnel vector has a component, called "1", that identifies the role of the person it concerns with respect to the published item, and a component, called "2", that gives the name of the person. Thus, a form concerning a publication of which Norbert Wiener is the author would contain a personnel vector whose first two components have the form

(1=Author) (2=Norbert Wiener) .

Vectors may have any number of components, but entries are made on the form only in those components for which information is specifically at hand. For example, personnel-type vectors include a component for the residence location of the person, but it is not entered unless that information is explicitly present in the source. The vectors in the bibliographic data form average about 20 components, each intended to accept a specific kind of data.

ENTRY OF DATA

The production of data forms is only the first step in the input process. The data must subsequently be extracted from the forms and translated into the Relational Information Language for storage in the file. This subsection will discuss the steps in executing that translation. But first, we describe a programming system for manipulation of the

data forms that is the basic tool for carrying out the translation, as well as a valuable auxiliary to the input process.

Form Manipulation Programming System

The purpose of the Form Manipulation Programming System is to enable a) the selection of input forms from the complete file on the basis of their properties, and b) the performance of specified operations on the selected forms. For example, the system would be able to select all bibliographic forms that have the "2" component in a personnel-type vector equal to "Norbert Wiener" and that also have any event-type vector. Such forms would concern published items with which Wiener was associated in some respect (author, editor, reviewer, etc.) and that had been presented at some meeting.

Having selected a set of forms according to some criterion, the system will be able to execute on each form operations such as: printing the entire form; printing selected vectors or components of the form; sorting the component values before printing; deleting or correcting the form; and constructing a sentence in the Relational Information Language from components in the form. The last operation, of course, is the basis for the data translation procedures.

Besides its use in translation, the Form Manipulation Programming System is a valuable tool for the input aides. It can be used, for example, to obtain lists of all names appearing in the input--a first step toward producing a thesaurus of equivalent names. It can be used also to

verify the completeness of data collection from a specific journal or of an individual's publications. It can be used to produce bibliographies without having to employ the main data file. These uses show that the Form Manipulation Programming System, which works on any data forms having the vector-component structure, is by itself a small-scale, but flexible, data retrieval system.

Translation Into Information Language

Once a data form has been filled in by an input aide, it is keypunched and given to the computer. It is then operated upon by routines written for the Form Manipulation Programming System to produce sets of relational sentences. This is the process of translation. How is the translation routine produced?

A new translation routine is prepared for each class of data forms. That is, there is one for bibliographic data forms, another for biographic data forms, still another for organization data forms, and so on. The translation routine is prepared at about the same time as the form, for translation requirements can affect the type of data collected and the format used.

The translation routine consists of a set of commands for the Form Manipulation Programming System that are precise descriptions of such rules as the following: If the first component of the first personnel vector (vector D1) is "author," then establish a sentence

D1'2 AUTHOR OF C1'2

where D1'2, the second component of the first D-vector, is the name of an author, and C1'2, the second component of the first item vector (vector C1) is the title of the publication.

The Form Manipulation Programming System executes the translation routine on each data form, producing a set of sentences in the information language for insertion in the file. If the information language changes or the data in a form is analyzed differently, a modified translation routine is prepared; but once that is done, the computer easily re-translates old data forms into revised data sentences.

VIII. OUTPUT PROCEDURES

THE PROBLEM

Section VII considered the problems of entering data in the file. This section considers the opposite problem, retrieving data from the file in response to the requests of the user--the output problem.

We begin with the kinds of output requests that the file should satisfy. Then we discuss the process of communicating those requests to the file, and describe the means of processing them.

KINDS OF REQUESTS

The output procedures for a data retrieval system must be evaluated relative to the needs of the prospective users. Therefore, we shall indicate briefly the kinds of use and the types of requests that the file is intended to serve.

Kinds of Use

We anticipate two main uses of the file: a) to continue and deepen research on its subject matter; b) to satisfy a current interest in some question falling within its domain. The requirements for these two uses are so different that it is necessary to provide two, almost distinct, kinds of output procedures.

Uses of the first kind, for example, are likely to require an extremely wide range of intricate searches, and the system designer probably cannot anticipate many of them. At the same time, a researcher should devote time

and effort to learning how best to employ the file. Consequently, the requirements of this kind of use can best be met by providing a repertoire of basic retrieval operations from which the user can construct procedures for his research problems.

Uses of the second kind, however, require direct replies to simple questions, many of which the system designer can anticipate. It is infeasible to expect researchers to devote much time or effort to learning how best to employ the file for such ends. Consequently, the necessary requirements can most satisfactorily be served by building into the file the capacity to analyze and answer questions directly.

Types of Requests

The file should be able to satisfy two types of requests:

- o Execute a procedure,
- o Answer a question.

Procedures. Some of the kinds of procedures that the file must be able to execute have been identified in Secs. V and VI:

1) It should be able to carry out the elementary retrievals, mentioned in Sec. V, that either test for the presence of a given sentence in the file or, if it is a sentence with variables, identify the values of the variables for which instances of the sentence are present;
2) It should have the capacity to execute procedures that when applied to retrieved data can produce counts, compute correlations, indicate trends, etc.;

3) It should be able to execute strict and plausible inference schemes of the kinds mentioned in Sec. VI.

Questions. An important distinction among questions may be made on the basis of what kinds of procedures must be executed in order to find their answers.

1) The answer to a question may be explicitly contained in the data file. In such a case, answering it requires only the execution of one of the elementary retrieval procedures.

2) The answer to a question may be available only after some preliminary processing of the available data. For example, such processing might include execution of counting or comparison procedures.

3) The answer to a question may not be explicitly contained in the data file, although it is implicit (in the sense mentioned in Sec. VI) in what is there. Consequently, obtaining its answer would require the execution of an inference scheme.

Many questions, of course, will fall into several of these classes; e.g., their answers might be partly explicit and partly implicit in the data file.

The feasibility of automatically obtaining an answer to a question depends largely on the classes to which it belongs: It is greatest if its answers are all explicitly in the file; least if its answers are mostly implicit (in which case automatic derivation of an inference scheme would be required).

COMMUNICATING REQUESTS

The two classes of use--continuing and current--impose different requirements for communication with the file.

During continuing and deep research, the greatest importance attaches to specifying precisely and concisely

the sometimes intricate procedures that must be executed. To do this, a language is necessary whose basic statements describe the operations and decisions to be performed: the Procedural Language.

During current use, it is most important to be able to state a question as easily and simply as possible. For this, a language is necessary whose basic statements are as close as possible to English language sentences: the Query Language.

Procedural Language

The Procedural Language has a vocabulary that consists of the following commands and decisions.

Extraction. These commands specify operations that take a relational request sentence with variables (e.g., x SUBORDINATE TO y, which has variables x and y), search the file for sentences identical with the request sentence except that the variables are replaced by constants (e.g., Jones SUBORDINATE TO Smith), and retrieve the constants as values of the variables (e.g., x is Jones and y is Smith). Often the values of only certain variables will be of interest, so the command statement should identify these. For example, the command

EXTRACT (x) FOR (x SUBORDINATE TO y)

would result in the retrieval of the values of x alone.

Manipulation. Commands of this kind specify the basic manipulations of the values obtained by the extraction command: e.g., the arithmetic commands required to specify

counts, correlations, and trend analyses; commands that specify ordering of the values, formation of the union or intersection of sets of values, determination of the first, last, or i^{th} member of an ordered set, etc.

Insertion. These commands specify processes that are the reverse of extraction. Given a relational sentence with variables (e.g., x SUBORDINATE TO y) and a set of values of the variables (e.g., x is Jones and y is Smith), they form sentences by replacing the variables with the specified values (e.g., Jones SUBORDINATE TO Smith). The primary role of the insertion operation is in the execution of an inference scheme; it is used to form the consequent statement, employing values for which the antecedent is satisfied.

Decisions. Commands of this kind, specifying possible branches in the retrieval procedure, have the form:

IF α, THEN GO TO σ,

where α is a testable condition and σ is the name of another command in the procedure. The command states that if condition α is satisfied (e.g., if a particular sentence is in the file), then the next command to be executed is σ; otherwise, the next command in order will be executed.

Input-Output. These commands specify the operations needed to establish a convenient interface between user and machine. Output especially is important; it is desirable to provide a flexible complement of print, plot, and display operations to assist the user to the fullest possible extent.

* * *

This then is the basic structure of the Procedural Language that will be used in continuing and deep researches Its building blocks can be combined to form simple procedure for retrieving a few pieces of explicit data or the intricat processes needed to execute complex inference schemes. More over, the language is independent of the subject of the file its commands and routines apply to any Relational Data File.

Query Language

The vocabulary of the Query Language is a subset of English and a grammar that permits simple English sentence forms.

It is developed evolutionarily. Vocabulary and acceptable grammatical forms are added as procedures for interpreting them are provided. At each step the user may have difficulty determining whether the question he wishes to ask will be interpreted correctly. To avoid error, therefore, the file feeds its interpretation of the specified query back to the user before seeking the answer.

PROCESSING REQUESTS

Regardless of the form in which a request enters the data file--procedure or question--it must be translated into a program in machine language that will perform the appropriate searches and manipulations and print out the desired results. Evidently, the translation process will be far simpler when the request is already a precisely defined procedure than when it is a freely phrased question.

Procedures. The Procedural Language must be accompanied by a compiler that translates a procedure into a program in computer language.

When a procedure request enters the file, therefore, it is compiled and executed directly. No further logical or linguistic processing is necessary.

Questions. This situation is entirely different. In this case, the preprocessor must accept questions phrased in an English-like language and produce procedures described in the Procedural Language. It is necessary, in other words, to treat explicitly the problems of accepting natural language input and of determining the procedure that will provide an appropriate answer to the question.

The first role of the question-accepting preprocessor is to translate as much as possible of the question from English to the Relational Information Language. For example, the question "Who wrote 'The Theory of Cybernetic Systems'?" should be transformed into a sentence like:

(Who x) (x AUTHOR OF The Theory of Cybernetic Systems).

The second role of the preprocessor is to specify the retrieval procedure that will provide an appropriate answer to the question. For example, the transformed sentence

(Who x) (x AUTHOR OF The Theory of Cybernetic Systems)

must give rise to a procedure such as:

1) EXTRACT (x) (x AUTHOR OF The Theory of Cybernetic Systems);
2) PRINT (x).

Somewhat greater difficulties arise when the question posed requires some manipulations of the extracted data for its resolution. For example, the question "How many papers has John Jones written?" must first be translated into

(How many x) ((x BELONGS TO CLASS papers) AND
(John Jones AUTHOR OF x)),

and that must, in turn, give rise to a procedure such as:

1) EXTRACT (x) ((x BELONGS TO CLASS papers) AND
(John Jones AUTHOR OF x));

2) SET y TO COUNT OF (x);

3) PRINT (y).

At the same time, a class of answers remains that seems to be inaccessible to an automatic question-answering system: those only empirically implicit in the data stored in the file. In order to obtain such answers, a plausible inference scheme must be employed. Construction of such inference schemes depends frequently on the use of background data and processes of reasoning that cannot, at present, be duplicated in a computer. Consequently, access to certain kinds of implicit data can be gained only by requesting the execution of a procedure that embodies the user's inference scheme and not by requesting the answer to a question.

Of the two means of access to the file that have been described, Procedural Language and Query Language, the former is the basic one for two reasons: first, because it provides the only access to an important class of implicit data;

second, because it is the language in which the question-accepting preprocessor must communicate with the file. Thus, although the foregoing discussion has paid equal attention to the Procedural and the Query Languages, the file development will devote primary attention to the Procedural Language. The Query Language will be developed as it appears useful and feasible, but the file will be usable in its full power with the Procedural Language alone.

IX. STORAGE AND PROCESSING PROCEDURES

THE PROBLEM

Sections VII and VIII described the interrelations between the Relational Data File and its two classes of users: those who enter data and those who extract it. This section examines the internal organization of the file: the structure of stored data and the programs that operate on those data.

STORAGE

While the basic content of the Relational Data File is a collection of binary relational sentences, in order to interact conveniently with its users and keep the size of the file within practical bounds it must also contain a number of other kinds of data. This subsection describes each of the kinds of data that will appear in the file and indicates the form in which they will be stored.

Relational Data

A file may hold data about a relation R in one of two forms: extension or intension.

Forms of Relation Storage. A relation is stored in extension as a set of sentences of the form, aRb, where each sentence denotes an instance of the relation. The LOCATED IN relation would be stored in extension, for example, as a number of sentences like:

Los Angeles LOCATED IN California
California LOCATED IN United States
Los Angeles LOCATED IN United States
.
.
.
and so on.

A relation is stored in intension, however, by specifying a procedure through which sentences of the form aRb may be derived. For example, the EMPLOYED IN relation between persons and geographical locations could be stored in intensional form through use of the following strict inference:

If (x EMPLOYED AT z) and (z LOCATED IN y),
then (x EMPLOYED IN y)

where x is a person, y is a geographical location, and z is an organization. The inference implicitly specifies a procedure for verifying whether (Jones EMPLOYED IN London) is true with respect to the contents of the file, which is to be searched for a pair of sentences of the form:

(Jones EMPLOYED AT z)
(z LOCATED IN London).

If such a pair with the same z in each exists, the sentence is verified; otherwise, it is not.

Thus, a relation may appear in the file in one of two ways: in extension, as an explicit set of relational sentences; in intension, as a procedure through which sentences of the relation may be derived. The file contains two

separate stores: an extensional store of relational sentences and an intensional store of derivation rules.

Extensional Store. Each sentence in the extensional file is an entry in a binary relation, so it must contain at least three components, one for each of the two arguments of the relation and one for the name of the relation. In addition, a number of other items of information should be associated with each primitive sentence: the source of the data described by the sentence, the dates on which the relationship denoted by the sentence is known to have held, the date that the sentence entered the file, the number of times it has been used, and so on. Yet, reserving components in each primitive sentence to represent each of those information items would unnecessarily clog the store and complicate data processing. So instead, each sentence is assigned a name, say S_i, and the file holds all source, date, and usage information through relational sentences in which "S_i" is one of the arguments. Each primitive sentence in the file consists of four components: the names of the sentence, its domain element, relation, and range element.

Intensional Store. The intensional store can be organized in much the same way as the extensional store. It may be viewed as expressing relations among the relations in the information language.

Other File Data

Besides the primary file of relational data, a number of other forms of information must be included in the data

store: A code dictionary to perform the translations between external and internal names; a thesaurus to identify the several different names for a single individual; and a natural language dictionary for interpretation of natural language queries.

Code Dictionary. Data appear in sources in forms inconvenient for computer storage. Names are not as compact as possible, wasting valuable storage space; and they vary in length, complicating storage allocation, file search, and data transmission. These difficulties may be overcome by mapping external data representations into compact, fixed-length internal codes. For example, a single 36-bit computer word may contain the coded name of any of $2^{36} \approx 69$ billion different objects, and clearly the needs of all feasible data banks could be served by assigning a single 36-bit code to each entity. Thus, with effective coding, the basic relational sentence can be stored in a handful of computer words. Without coding, an article title alone might occupy 10 or 15 computer words.

To carry out the translation between external and internal coded representations, a coding routine and code dictionary is needed. The code dictionary, which has to be consulted during each entry to and exit from the file, occupies part of the basic data file.

Thesaurus. In the data sources, many items may be called by several different names. For example, in different contexts the same person may be referred to as "J. V. Jones," "John Jones," "J. Vincent Jones," or "John Vincent Jones." Each datum entered in the file uses the name as it appears in the specific source. But when the

data file receives a request for information about J V. Jones, all the data concerning him, no matter under which form of his name they were entered, should be retrieved. The vehicle for achieving this identification is a thesaurus a dictionary that identifies the synonymous forms of each data name. Thus, a request for information about J. V. Jones first goes to the thesaurus, where it is expanded to include the other forms of Jones' name.

Besides identifying the several forms of names of the entities that enter relationships, the thesaurus can be used to recognize the various names for relationships themselves. For example, all the phrases "author of," "writer of," "wrote the publication," "author of the publication," "authored," and so on, may be used to denote the relationship AUTHOR OF between persons and publications. If the thesaurus is used to identify all these phrases, the user has greater freedom in phrasing his requests.

Natural Language Dictionary. The translation of simple natural language requests to procedures for retrieving their answers demands some analysis requiring a natural language dictionary. The nature and form of the information in the dictionary depends, of course, on details of the translation process. However, it must perform functions such as identifying which words are question words, names of objects, names of relations, and auxiliary words; determining the part of speech of specific words; and noting the tense of verbs.

PROCESSING

The descriptions in Secs. VII and VIII emphasized the processes of input and output as they might appear to the external observer. This subsection briefly discusses these internal-processing steps as they appear to the system programmer.

Input

The primary input to the file is data forms. The task of the input programs is to transform the information on these forms into relational sentences for storage. There are actually four steps in the process.

Forms to Expanded Forms. The input aides employ a number of conventions in order to avoid repetitive typing. Before relations may be extracted from the forms, these conventions must be interpreted and the repeated information inserted in the appropriate places. The result is called the expanded form. Its formation is the first step of the input process.

Expanded Forms to Relational Sentences. The next step employs a routine, provided by the designers of the input data form and specified in the Form Manipulation Programming System, to extract relational sentences from the data form.

Relational Sentences to Coded Relational Sentences. As noted in the section on storage, data are coded before being entered in the file. The coding program must scan each sentence, find the code for each entry, and, if a new code has to be assigned, update the code dictionary appropriately.

Coded Relational Sentences to Data File. The final step in the input process inserts the extracted and coded relational sentences at the appropriate locations in the extensional relation file. This routine also includes housekeeping functions that check for duplicated sentences, construct appropriate cross-indexes, reorder file structures, etc.

Other Input Data. The steps outlined above must be followed to enter extensional data sentences. Routines must also be provided to enter new intensional sentences, thesaurus entries, and natural language dictionary entries. The details of those routines must await further specification of the structures of these data files.

Output

The fundamental request to the file is a procedure expressed in Procedural Language. The task of the principal output programs is to accept such a request and produce the appropriate output data. The process has two steps.

Procedure to Retrieval Program. The procedure, expressed in a form convenient for the user, is translated into a form convenient for the computer. The commands are translated into computer instructions and the relational sentences are coded (through the code dictionary) to conform to the internal data format.

Retrieval Program to Retrieved Data. The computer program is executed and the retrieved data is made available to the user. The steps required to execute the EXTRACT command described in Sec. VIII illustrate this step.

For concreteness, consider the command:

EXTRACT (x): (J. V. Jones WROTE x)

(The variable and the relational sentence would be in internal code, but for clarity we shall leave them in their external form.)

The first step is to "normalize" the relational sentence through use of the thesaurus. Entity names are replaced by all their synonymous forms; relation names are replaced by their preferred form. The command then would appear as follows:

EXTRACT (x): (J. V. Jones AUTHOR OF x)
OR (John Jones AUTHOR OF x)
OR (John V. Jones AUTHOR OF x)
OR (J. Vincent Jones AUTHOR OF x)
.
.
.
etc.

The second step is to search the extensional data file for sentences satisfying one or another of the request sentences and as they are found to add the appropriate values of x to a list.

If the values of x were to be used in a subsequent search of the file, the thesaurus would be used to find all of their synonyms. For example, if

AND (x PUBLISHED IN Proceedings IEEE)

were part of the request sentences, the list of x's would be expanded to include all alternative forms of the titles. Then for each x in the list, a search would be made to see if (x PUBLISHED IN Proceedings IEEE) appeared in the file. If it did, that value of x would remain on the list; if it did not, the value would be removed.

When the list of x's is complete, a check of the thesaurus is made to remove all but one of the synonyms for each object. However, all appropriate answers to the request may not appear in the extensional file. In that case, the intensional file is consulted to determine means of evaluating the AUTHOR OF relation.

For example, separate relations may be stored in the file for PRIMARY AUTHOR OF and SECONDARY AUTHOR OF. In that case, a derivation procedure based on the following strict inference would appear in the intensional file:

(w AUTHOR OF x) IF ((w PRIMARY AUTHOR OF x)
OR (w SECONDARY AUTHOR OF x))

In this procedure, the file would be searched for sentences of the form:

(J. V. Jones PRIMARY AUTHOR OF x)
(J. V. Jones SECONDARY AUTHOR OF x)

and so on. The values of x obtained would be added to the list of x's found in the extensional file.

Natural Language Questions. Simple questions, phrased in English, must undergo an additional processing step in order to be translated into a procedure, phrased in Procedural

Language, for execution by the file. The additional step is performed by a self-contained preprocessor that, however, employs the natural language dictionary contained in the file.

The preprocessor must identify the words in the question that name relations and entities, are logical connectives, specify unknowns, and indicate the type of question concerned. It then must modify the sentence until it has only well-formed relational sentences and question words. Finally, it uses the question words and relational sentence to identify the appropriate retrieval procedure.

X. LITERATURE SEARCHING

THE PROBLEM

For two reasons, the Relational Data File may be applied to the problem of literature searching.

First, the data file lends itself directly to literature searches based on conventional bibliographic or subject indexes. Second, and more important, the flexibility and breadth of the data file enable information about the entire context in which research in a given discipline is conducted to be stored. The data file, therefore, lends itself to searches that employ a wide variety of clues about the persons, organizations, and journals concerned with a given subject. It is believed that by employing such clues the process of machine-assisted literature searching can be made more effective and convenient.

This section first shows how a Relational Data File can be applied to literature searching employing the usual bibliographic and subject data, and then demonstrates how far more complex searches employing context data may be executed with the file's help.

LITERATURE SEARCHING USING BIBLIOGRAPHIC DATA

Description

The most direct means of reference to a publication is through one or more elements in its bibliographic description. In a literature search, the user may know the author's name or the title and date of a publication, and wish to

know enough of the remaining bibliographic facts to enable him to find the publication.

Application of Data File

A data retrieval system is adaptable to such a task. The information language must include relations such as AUTHOR OF, PUBLISHED IN, and TITLE OF and the file must include a set of sentences expressing the relationships between authors and their papers, between papers and their journals of publication, and so on. Data about the context of cybernetics research naturally include such sentences; thus the experimental corpus for the present system may easily be turned to bibliographic retrievals.

For example, a researcher interested in the papers of Norbert Wiener would be able to find them through the file by means of a request of the form:

EXTRACT (x) ((Norbert Wiener AUTHOR OF x) AND
(x BELONGS TO CLASS papers))

The file would produce a list of all the publications "x" appearing in both such sentences.

LITERATURE SEARCHING USING SUBJECT DATA

Description

Often a researcher knows nothing about the documents he seeks except their subject. He may wish, for example, to find documents most relevant to pattern recognition. Since it is infeasible to identify all subjects at the

time of each request, documents are generally indexed when they enter the collection. The problems arising from such an approach are well known, but difficult to avoid: the assignment of subject indexes is laborious, requires the kind of knowledge indexers do not usually have, demands a similarity of view and vocabulary between the indexers and the requesters, and can become quickly outdated by advances in science or technology.

Application of the Data File

In order to use a data retrieval system as a subject-indexed literature-searching system, both bibliographic and subject information about the desired publications must be added to the file. To characterize a publication's subject, a single relation in the information language might be sufficient: SUBJECT INDEXED UNDER. Then, for each publication, one or more additional sentences of the form "x SUBJECT INDEXED UNDER y" would be inserted in the file. A request for all publications relevant to pattern recognition might take the form:

EXTRACT (x) (x SUBJECT INDEXED UNDER pattern recognition)

The response would be a list of all publications whose name "x" appeared in a sentence of that form.

Storing a Thesaurus

One problem in subject indexing arises from the hierarchical nature of subject areas. Pattern recognition

is a subdiscipline of artificial intelligence, which is a subdiscipline of cybernetics. The interrelations among subjects are so complex that it is difficult, if not impossible, to identify at every level all to which a document may be relevant. Consequently, an attempt is sometimes made to identify the most specifically relevant subjects. To serve general requests, it is necessary to provide some means of explicating general subjects. This may be done by a thesaurus stored within the literature searching system. The thesaurus may also be used to reduce the mismatch between index terms and request terms by identifying synonyms, and to expand the area of search into closely related areas by means of "see also" entries.

A thesaurus may easily be included in a Relational Data File. The information language must contain relations such as SUBDISCIPLINE OF, IDENTICAL TO, and RELEVANT TO among subject areas; also, a subject specialist must enter into the file sentences such as "Pattern Recognition SUBDISCIPLINE OF Artificial Intelligence," "Switching Theory IDENTICAL TO Theory of Switching," and "Switching Theory RELATED TO Automata Theory"; and finally, that the data file be provided with inference schemes such as "If (x SUBJECT INDEXED UNDER y) and (y SUBDISCIPLINE OF z), then (x SUBJECT INDEXED UNDER z)."

Subject Index Assignment

Because the assignment of index terms to a publication is a difficult and time-consuming task, an efficient computer-based literature-searching system should provide easier and faster means. While the Relational Data File

can accept index terms produced by any source, it also has unique capabilities for automatically indexing publications. This section will discuss the possible outside sources of index assignments, and some possibilities for automatic assignment within the file.

Outside Sources of Indexes. There are three principal external sources of index statements.

- At the time that bibliographic information is being prepared for entry into the file, the input aides could manually assign index terms to the publication.
- Any technique for automatically assigning index terms could be employed at the input to the data file. A number of such techniques employing keyword or frequently occurring words in titles, abstracts, or text have been suggested. The only changes needed to employ existing or future techniques would be those necessary to produce the output in the form of relational sentences.
- Extensive use could be made of indexing that is done independently of the input procedures and under different auspices. For example, review and abstract journals provide subject indexes; subject-indexed bibliographies frequently appear in the technical press; journals frequently divide their contents into subject-area departments; and libraries assign classification numbers to books.

Sources of Indexes within the File. As an example of the techniques that can be used within a data retrieval system to assign indexes to publications automatically, let us consider one employing information about citations.

The technique might work as follows: During input, as each publication is being scanned for bibliographic data, it is also checked for citations. For each publication "x" that is cited in document "d", a relational sentence of the form "x CITED BY d" is entered in the file. (The file

is also assumed to include subject index information.) Now, the following inference may be made: If a publication "d" refers to a publication "p", and "p" has previously been assigned the subject index "s", then publication "d" is also relevant to subject "s". This is not a strict inference, of course, but one might assign a moderate degree of plausibility to it; and even such a simple approach could be useful in some cases. Thus, the inference schema "If (p CITED BY d) and (p SUBJECT INDEXED UNDER s) then (d SUBJECT INDEXED UNDER s)" could be used to assign subject indexes automatically.

But in a data file containing research-context information, it is not necessary to stop with simple inferences. The above example can be improved by bringing more of the contextual information in the file to bear: e.g., one might decide that the hypothesis that a citing paper is relevant to the same subjects as the papers it cites would be more valid if a simple check were made to insure that the subject is one in which the author of the citing paper is known to be interested. A file on the context of research in a given scientific field would naturally contain sentences of the form "a PROFESSIONALLY INTERESTED IN s", where "a" is the name of an individual and "s" is the name of a subject. Thus, the antecedent of the citation inference could be expanded by addition of the phrase "and (a AUTHOR OF d) and (a PROFESSIONALLY INTERESTED IN s)."

Another check could be based on the recognized practice of professional journals to publish only articles on certain specified subjects. The inference scheme could be expanded to include a phrase insuring that the index terms assigned

are only those known to lie within the journal's area of interest. Many other improvements of this kind could be made. The value of the Relational Data File is that it enables inference schemes to be altered and elaborated easily and it contains the wide store of data essential for the construction of sophisticated inferences. Consequently, the file can serve as a valuable tool in the design and testing of automatic subject-indexing techniques.

LITERATURE SEARCHING USING CONTEXT DATA

The value of a data retrieval system for literature searching is that the research can proceed in a myriad of ways adaptable to the searcher's individual demands and knowledge. He is limited neither to bibliographic nor to subject clues, but may work within the entire context of research, and go beyond the traditional approaches to literature searching.

Description

Manual literature searches frequently employ clues other than the bibliographic or subject characterizations. A scientist's search, for example, will employ many kinds of information about the people, organizations, journals, and locations associated with studies of a specific subject area; i.e., almost the full range of information about what we have called "the context of research." Moreover, his search will continually adapt and evolve in response to the information received in progress.

Context-guided, adaptive searches can be extremely effective when done manually, but their quality depends

upon the quantity of available information and time. Consequently, computer assistance can improve their average quality. At the same time, their mechanization could lead to a comparable improvement in the quality of computer-assisted literature searches.

Application of the Data File

In order to use a data retrieval system as a literature-searching system employing context clues, it is necessary to include, in addition to bibliographic and subject information, a wide variety of data concerning the relevant people, organizations, journals, and locations. Relations such as CONDUCTS RESEARCH ON, SPONSORS RESEARCH ON, STUDENT OF, AFFILIATED WITH, LOCATED IN would be included in the information language. As mentioned in Sec. III, this kind of data is being collected for the corpus on the context of cybernetics research. So the Relational Data File is well-adapted to literature searching using context data.

Take, for example, the situation in which the searcher would like to use clues about persons or organizations concerned with, say, pattern recognition, to lead him to relevant publications. He need not only rely upon his own awareness of such persons and places, he can employ the file as follows:

EXTRACT (y) ((x CONDUCTS RESEARCH ON pattern recognition) AND (x AUTHOR OF y)) OR ((z SPONSORS RESEARCH ON pattern recognition) AND (z SPONSORS PUBLICATION OF y))

where x is a person, y is a publication, and z is an organization.

Or, suppose the searcher has heard that an electronics conference held in Boston had some especially good papers on pattern recognition. He could employ the file to find them as follows:

EXTRACT (z) (x HELD IN Boston) AND
(x CONCERNED SUBJECT OF Electronics)
(x SPONSORED PUBLICATION OF y) AND
(z PUBLISHED IN y) AND
(z SUBJECT INDEXED UNDER pattern recognition)

where x is the conference, y is the proceedings of the conference, and z is a paper in the proceedings.

The intricacy with which context-based searches can be made in the Relational Data File is delimited only by the available input data and the ability of the user to construct complex inference schemes.

Context Hypothesis

Until the Relational Data File becomes operational, there is little evidence on the effectiveness of computer-assisted, context-based literature searches. The assertion that use of context data will improve literature searches is a hypothesis--the context-hypothesis. However, some informal arguments justify it.

The context-hypothesis appears to be reasonable because:

- Scientists seem to conduct their own manual searches according to a wide variety of context clues;
- Manual contextual searches can undoubtedly be improved in thoroughness and complexity through machine assistance;

- o Context searches employ both bibliographic and subject data, thus they contain as special cases both of those kinds of searches;
- o The flexibility inherent in keeping a file of data relevant to the entire context of research and in providing the ability to perform a wide range of searches can only improve literature searches.

Data Retrieval and Literature Searching

Before leaving the subject of literature searching, two points should be emphasized:

First, because of the flexibility inherent in the Relational Information Language and the ease of defining procedures to work on relational data, the Relational Data File may be employed to test a wide variety of possible literature-searching schemes.

Second, literature searching is only a specific application of the Relational Data File, a natural by-product of our choice of experimental corpus. The file, however, may be used to store and retrieve data about subjects totally unrelated to literature searching. (Section XI mentions some possibilities.)

XI. EXTENSIONS AND APPLICATIONS

THE PROBLEM

The preceding sections have described the implementation of the Relational Data File. The design choices have been made with the objective of quickly achieving a useful file, avoiding difficult problems not intimately related to the process of data retrieval. However, once an initial Relational Data File is achieved, it will probably be both desirable and feasible to add refinements. This section discusses some of these. In addition, a number of potential applications of the file to corpora other than the context of cybernetics research will be described.

INPUT REFINEMENTS

The data input process can be improved by allowing input in natural language, by introducing the means for direct, on-line entry of data, and by establishing feedback from the file about correspondences between new and previously stored data.

Natural Input Language

Use of a natural input language for at least a portion of the data base should pose few problems beyond those that have to be resolved at output. The translation of natural language queries into acceptable procedures involves as a subtask the translation of the simple natural language statements into the Relational Information Language.

On-Line Data Entry

The attractive prospect of a natural language input ability is that of enabling a user with an on-line console to make direct entries to the file. Once such consoles are available, access to the file will be convenient to a great number of individuals possessing much information that they can file without having to learn special procedures. An on-line console would have further value: even if entry were made on forms by input aides, the direct connection would enable them to execute their task more quickly, effectively, and accurately.

File Feedback

The direct interaction between input user and file through an on-line console could be used to detect and correct imperfections in data during the input process. Among the imperfections for which this might be useful are ambiguities (e.g., two or more individuals may have the same name), incomplete information (e.g., a book's date of publication may have been left out), and repetitive or inconsistent information (e.g., two birth dates may be associated with the same person).

STORAGE AND PROCESSING REFINEMENTS

The Relational Data File will store vast quantities of data. Current technology gives no hope of keeping any but the immediate requirements in the internal computer store. Much data will be kept on auxiliary random-access devices (e.g., discs). Still other data will have to

reside on auxiliary sequential-access devices (e.g., tapes). Consequently, obtaining a specific item of information can take a long time unless great care is spent in designing store access procedures and structure. One way to expedite access is to make the whole data structure adapt to the demands of the user.

Adaptive Storage

The techniques of adaptation could include a transfer of data items from one form of bulk storage to another, a rearrangement of sentences in the file, and a modification of the form in which a relation is stored. Two forms of relation storage were described in Sec. IX: extensional and intensional storage. The choice between those forms for a given relation will depend in part on the frequency with which the relation is used, the number of entries in the relation, and the access speed and volume of the several storage media.

On-Line Processing

Another set of desirable refinements to storage and data-processing procedures are those associated with the introduction of on-line, time-shared access to the file. There are problems associated with 1) executing the required operations in real-time, and 2) servicing multiple simultaneous users. The real-time requirements should be satisfied through flexible and adaptive store organization and efficient search strategies. The multiple-user requirements should yield to the buffering and store-sharing techniques currently being developed for computational applications.

OUTPUT REFINEMENTS

An eventual objective for a computer data file might be the following output process: without special instruction the user approaches and communicates with the file directly in English. The file responds with output at the user's pace. In addition to providing answers to di ect queries or executing complicated retrievals, the file provides assistance in posing requests, generates and carries out many inferences without external help, and stores and reports the relevant experience of previous users. Once an initial file is achieved, work can begin on extending its capabilities. One line of research could be concerned with improving the man-machine interaction at output; another with mechanizing more of the inference-making tasks.

Improved Man-Machine Interaction

Once an on-line console and an English-like request language make access to the file convenient, it becomes desirable to provide assistance to the inexpert user through feedback from the computer. For example: The user indicates that he is interested in studying organizations concerned with pattern recognition research. The file responds by listing all the relations between organizations and subject areas that appear in the file. The user phrases his request in terms of the relation that seems most appropriate. Should his request call for a very large reply, the file indicates its size and verifies that the user desires it. If he does not, the file suggests

the form of an additional requirement to narrow the field of appropriate responses.

Mechanized Inference Making

Still more assistance could be provided the user in the field of inference making. A simple form of help could be based on storage of inference schemes constructed by previous users. Should the user enter a request for which a stored inference scheme would be relevant, the file would display it to him and, should the user desire, execute it. But the file could also be provided with the capacity to generate some inference schemes on its own.

The machine construction of arbitrary plausible inference schemes poses a difficult, if not impossible, problem. Usually such inferences are constructed on the basis of a wide variety of empirical knowledge that a specialized data file would not be expected to contain. There may, however, be techniques that can be used to construct a number of types of strict inference schemes, and, possibly, some plausible ones.

Strict Inferences. A number of strict inference schemes, for example, follow from the basic properties of relations. If a relation R is known to be transitive, then an inference scheme of the form

IF (aRb) AND (bRc), THEN (aRc)

is justified. If each stored relation is characterized in terms of the properties of transitivity, reflexivity,

symmetry, and so on, design of routines which generate certain strict inference schemata becomes possible.

Plausible Inferences. One way the file could construct plausible inference schemes is by generalizing schemes presented by users. For example, if a user had employed the following:

IF (x AUTHORED PAPER y)
AND (y SUBJECT INDEXED UNDER pattern recognition)
THEN (x CONDUCTS RESEARCH ON pattern recognition)

where x is a person and y is a paper, the file could generalize it by substituting a variable z, ranging over the set of subject areas, for the constant "pattern recognition"; or by replacing the variable y, ranging over the set of papers, with y', ranging over the set of publications; or by replacing the relation "AUTHORED PAPER" with "ASSOCIATED WITH PAPER," where the latter relation subsumes editorship, sponsorship, presentation, and so on.

Though the file itself can be used in several ways to construct inferences, implementation of an inference-constructing capacity will have to wait until experience has been gained with an operating file. And the collection of manually prepared, formally expressed inference schemes from various users will not be the least of the file's values. Examination of these actual schemes should suggest fruitful techniques for machine inference construction, as well as provide insight into the process of human construction.

POTENTIAL APPLICATION OF THE FILE

In order for application of the Relational Data File to be of value, a subject must satisfy three conditions.

1) A large body of data must be stored and retrieved, and it should probably be rapidly accumulating. If these requirements are not satisfied, then some simpler system is probably preferable.

2) There should be a need for extensive, unpredictable manipulation of the data. One of the Relational Data File's major advantages is the ability to command a wide variety of extensive manipulations of primitive data. If there were no such need or if all the manipulations could be specified in advance, a manual or simple computer file might suffice.

3) The body of data to be stored must lend itself to expression in a Relational Information Language. The limits that this requirement imposes have not been clearly identified. About all that can be said at present is that expressing data about the context in which research is conducted appears considerably easier than expressing data about the content. The relations among the scientists, organizations, and publications active in cybernetics research lend themselves more obviously to expression in the form of binary relational sentences than do the basic ideas of cybernetics.

The experienced reader may be able to suggest promising areas of application that satisfy the foregoing requirements The following three areas are candidates.

- The context of research--While the context of cybernetics research has been used as the experimental corpus, the ideas and most of the information language developed may be easily extended to any other field of research or development. As a matter of fact, the file could almost as easily be adapted to keep track of the people, organizations, and activities associated with any common enterprise (e.g., medical, legal, industrial, criminological, etc.).

- The management of organizations and projects--Much of the data collected in the course of managing a large organization or project could be expressed in a Relational Information Language. Such data might include the responsibilities of groups and individuals, completion times of subprojects, results achieved, and so on. The Relational Data File would enable the manager to find knowledgeable individuals, to keep track of progress, and to consider the effects of alternative policies. Adaptation of the File to a data base of this kind would require construction of new input forms and a new information language.

- Research in the social sciences--A number of the social sciences concern one aspect or another of the relationships among people, organizations, power groups, principles, and events. A Relational Information Language appears to be an appropriate way to store such data. Where the quantity of data, intricacy of analysis, and research support warrant it, a Relational Data File could profitably assist research in history, sociology, anthropology, archaeology, or political science.

XII. SUMMARY AND EVALUATION

This section summarizes objectives and principal features of the Relational Data File and compares it with alternative approaches to data retrieval.

OBJECTIVES

The research described in this Memorandum has two principal objectives:

1) To develop a computer-based file for the storage of data consisting of statements of fact and to develop procedures for retrieval and logical processing of and inference execution over the file data;

2) To collect and process a corpus of data of sufficient size and interest to permit an effective test of the data file techniques.

While the first objective is paramount, we believe for two reasons that it cannot be achieved except in unison with the second: a) many important theoretical problems become evident only through practical applications; b) many apparently satisfactory theoretical solutions break down in practice.

PRINCIPAL FEATURES

One convenient way to summarize this research is to list the principal features of the Relational Data File.

Linguistic Features

Formal Information Language. Data are stored in the file as precise, unambiguous statements in an artificial

information language. The basic unit of information in the file is the sentence. All more complicated types of information are decomposed into sentences for storage in the file.

Relational Information Language. The information language has a simple syntax; every sentence has the form aRb, where a and b are arbitrary entities and R is a relation that holds between them.

The semantics of the information language is defined by the choice of relations, R, that may appear in stored sentences. The choice depends, of course, on the corpus.

Extensional Representation. A relation, R, may be explicitly stored in the data file as a list of sentences, aRb. It is then said to be stored in extension.

Intensional Representation. A relation, R, may be stored as a set of rules whose application to the extensional file will derive sentences, aRb, of R. It is then said to be stored in intension.

Logical Features

Access to Explicit Data. The file may be used directly to answer two types of questions about data stored explicitly in the file: a) it can verify whether or not a request sentence, aRb, is in the store; b) it can identify the values of specified variables for which a request sentence with variables, say xRy (x and y are variables), is in the file. A request of the first type, for example, would be to verify that

John Jones AUTHOR OF On Cybernetics.

A request of the second type, for example, would be to identify the values of x (authors) in

x AUTHOR OF On Cybernetics.

Access to Implicit Data. The conjunction of stored explicit data confers a degree of plausibility on a large amount of data not in the file. Such data are implicit in the file. The process of deriving implicit from explicit data is called inference. The vehicle for inference is called an inference scheme; it takes the form

If A, then C

where A, the antecedent, is a sentence (possibly a logical conjunction of many simple sentences) in the Relational Information Language whose truth may be determined by searching the file; and C, the consequent, is a sentence in the Relational Information Language that is formed if A is true. If the truth of C follows logically from the truth of A, the inference is said to be strict. When the truth of A only confers a degree of plausibility on C, the inference is said to be plausible.

Data implicit in the file may be obtained through execution of a strict or plausible inference scheme over the file contents.

Manipulation of Data. Data retrieved from the file may be manipulated by arithmetic, statistical, or symbolic processes to obtain counts, correlations, trend analyses, ordered lists, greatest and least members of sets, etc.

Output Features

Human Control of Retrieval Processing. For the most part, the user performs the empirical, logical, and data processing analysis needed to pass from a question to the data processing routine that will retrieve the answer to the question.

Query Language. Questions that can be answered satisfactorily through elementary retrievals and simple manipulations will be posed to the system in an English-like Query Language.

Procedural Language. Questions whose answers demand complex retrievals, or manipulations, or the evaluation of inference schemes, will have to be answered through use of a user-provided retrieval routine, written in a formal Procedural Language, that commands the appropriate operations of the file. Thus, the process of constructing an inference scheme remains with the human user; the computer performs the laborious, but mechanical, searching of the available data to identify the specific consequences of the inference scheme.

Input Features

Structured Data Forms. Data become available for input in many instances in fixed, recurring groupings.

Biographic data about individuals, bibliographic data about publications, organizational data about institutions, and so on, fall into predictable categories and occur frequently enough to warrant special treatment. Data forms are designed to capture each such distinct grouping for the experimental corpus. The forms all belong to a single class, related in basic structure, format, notation, etc.

Form Manipulation Programming System. A Form Manipulation Programming System is used to write procedures to select stored forms from a collection on the basis of their contents. Procedures can also be defined to delete or insert data into the forms, to select data for printing, or to construct binary relational sentences for entry into the data file.

Experimental Corpus

Large Quantity of Real Data. The context of cybernetics research corpus consists of data concerning a subject of real and immediate interest. The data stored are chosen on the basis of their relevance to an understanding of the subject, and not for reasons of convenience in developing the file. The vast quantity of available data are more than enough to justify the use of computers and to provide a meaningful test of their ability.

Meaningful Testing of Logical and Linguistic Concepts. The use of a real, diverse, and large data base enables meaningful tests of the logical and linguistic ideas underlying the structure of the proposed data files. In attempting to treat a large corpus of real data formally, problems

arise that might go unnoticed in a purely theoretical investigation.

Testing of "Context Hypothesis." Since the stored data base will concern the context in which cybernetics research is conducted, it will enable a trial of the ideas underlying the "context hypothesis" for literature searching.

COMPARISON WITH ALTERNATIVE APPROACHES

How does the Relational Data File compare with alternative techniques for achieving the same ends?

We can categorize the alternative techniques for storing and retrieving data into three major groups: manual and semi-manual, simple computer files, and question-answering systems.

Manual and Semi-Manual

The most basic data storage and retrieval routines depend on some combination of the human mind and such simple aids as handbooks, notes, manual card files, and basic punched card processes.

In comparison, the Relational Data File offers the following advantages.

- More data can be stored and retrieved conveniently and compactly through the use of computer bulk-storage devices.
- Speed and accuracy of retrieval of selected data increases through the use of a high-speed digital computer.
- More index keys exist for access to data, since each relation in the file essentially defines a new key.

- The intricacy of feasible retrieval procedures increases through the programming of a high-speed digital computer to do the highly repetitive, time-consuming searching.

Each of these advantages is essentially one of quantity, an increase in performance of a task that could, if necessary, be done manually. But the quantitative changes are large enough to become qualitative, enabling the performance of tasks impossible with a manual file.

Simple Computer Files

Some of the deficiencies of manual and semi-manual files can be overcome through the use of simple computer files. Typically, such a file might contain standard, preselected items of data arranged in groups in fixed formats and stored on tape or discs. Each different kind of data retrieval requires the construction of an appropriate program and, in most cases, the services of a programmer.

In comparison, the computer-based Relational Data File offers the following advantages.

- Greater ability to vary the kinds of data stored during use without affecting already stored data and existing retrieval routines. Such variations would be achieved in the relational file simply through addition of new relations to the information language; in the simple computer file, they would require the design of new record formats or alteration of old ones and possible rewriting of existing retrieval programs.

- Easier access to the file through natural language for simple retrievals or a convenient procedure language for complex retrievals. Moreover, the

Relational Information Language provides a more natural vehicle for expressing inferences than the record formats of simple computer files.

- Greater flexibility and adaptability in the kinds of requests that the user may conveniently make of the file. His data needs and questions do not have to be anticipated in detail in the system design.

- Programs that are independent of the file subject matter. All subject-matter-specific items are concentrated in the information language and input-form design; in changing subjects, only those two portions of the Relational Data File have to be altered; the programs remain invariant. In a simple computer file, however, programs are usually dependent on record format. Consequently, a complete new file must be designed for each change in subject matter.

Question-Answering Systems

Development of question-answering systems has been intended, in part, to overcome the problems of inconvenient access and inflexibility that arise with simple computer files. The primary objective has been to enable the user to pose questions in natural language. Several such systems have included some mechanization of the logical analysis needed to pass from a question to the process of answering it by indirect evidence. The Relational Data File can be considered a question-answering system. However, it differs from most such systems in the following respects.

- It includes, and in fact emphasizes, the use of plausible as well as strict inference in obtaining the answers to questions.

- It leaves to the user the problem of deciding which retrieval procedures and inference schemes are appropriate. Thus, for the most part, logical analysis is the role of the human being, while mechanical searches and retrievals are assigned to the computer.

- Its procedures are completely subject-independent. They apply equally well to any data corpus expressed in a Relational Information Language.

- It replies to questions without having to anticipate them and store appropriate reply programs. The user may enter his own reply program or, if he has a simple question, phrase it in English.

- It is being developed to handle a large body of real data used as an experimental corpus during the file's development.

REFERENCES

1. Simmons, R., "Answering English Questions by Computer: A Survey," Comm. ACM, Vol. 8, No. 1, January 1965, pp. 53-70.

2. Uspenskii, V. A., "The Problem of Constructing a Machine Language for an Information Machine," in Problems of Cybernetics II, A. A. Lyapunov (ed.), Pergamon Press, New York, 1961, pp. 356-371.

3. Carnap, R., Introduction to Symbolic Logic and Its Applications, Dover Publications, Inc., New York, 1958.

4. Kochen, M., Adaptive Man-Machine Non-Arithmetic Information Processing (Final Report), Air Force Cambridge Research Laboratory, AFCRL TR 62-01, June 1962.

5. Levien, R., and M. E. Maron, Cybernetics and Its Development in the Soviet Union, The RAND Corporation, RM-4156-PR, July 1964.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY	2a. REPORT SECURITY CLASSIFICATION
THE RAND CORPORATION	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

RELATIONAL DATA FILE: A TOOL FOR MECHANIZED INFERENCE EXECUTION AND DATA RETRIEVAL

4. AUTHOR(S) (Last name, first name, initial)

Levien, Roger and M. E. Maron

5. REPORT DATE	6a. TOTAL NO. OF PAGES	6b. NO. OF REFS.
November 1965	102	5

7. CONTRACT or GRANT NO.	8. ORIGINATOR'S REPORT NO.
AF 49(638)-1700	RM-4793-PR

9a. AVAILABILITY/LIMITATION NOTICES	9b. SPONSORING AGENCY
	United States Air Force Project RAND

10. ABSTRACT

A description of the background and status of a current project on automatic data storage and retrieval. The research emphasizes the development and testing of logical techniques for data retrieval and inference-making. The techniques are being implemented in the form of computer routines, and tested on a large body of facts concerning the field of cybernetics. Various sections present the theoretical base of the proposed system, a summary of the theory of relations, typical data retrieval requests, the key problems of inference, techniques for practical realization of the data file, output problems, storage and processing problems, the data file as a whole, the question of literature searching, and future steps to be taken to extend the capability of the system.

11. KEY WORDS

Information storage and retrieval
Data processing
Cybernetics
Computer programs