COMPUTER SYSTEM SUPPORT FOR DATA ANALYSIS

Thesis by

Norton Robert Greenfeld

In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

California Institute of Technology

Pasadena, California

1972

(submitted March 6, 1972)

ACKNOWLEDGMENT

Professor Frederick B. Thompson has influenced every part of this thesis. His insight, understanding, innovative depth, and friendship have been essential to this work and the author.

ABSTRACT

This thesis is an investigation into the nature of data analysis and computer software systems which support this activity.

The first chapter develops the notion of data analysis as an experimental science which has two major components: data-gathering and theory-building. The basic role of language in determining the meaningfulness of theory is stressed, and the informativeness of a language and data base pair is studied. The static and dynamic aspects of data analysis are then considered from this conceptual vantage point. The second chapter surveys the available types of computer systems which may be useful for data analysis. Particular attention is paid to the questions raised in the first chapter about the language restrictions imposed by the computer system and its dynamic properties.

The third chapter discusses the REL data analysis system, which was designed to satisfy the needs of the data analyzer in an operational relational data system. The major limitation on the use of such systems is the amount of access to data stored on a relatively slow secondary memory. This problem of the paging of data is investigated and two classes of data structure representations are found, each of which has desirable paging characteristics for certain types of queries. One representation is used by most of the generalized data base management systems in existence today, but the other is clearly preferred in the data analysis environment, as conceptualized in Chapter I.

This data representation has strong implications for a fundamental process of data analysis -- the quantification of variables. Since quantification is one of the few means of summarizing and abstracting, data analysis systems are under strong pressure to facilitate the process. Two implementations of quantification are studied: one analagous to the form of the lower predicate calculus and another more closely attuned to the data representation. A comparison of these indicates that the use of the "label class" method results in orders of magnitude improvement over the lower predicate calculus technique.

CONTENTS

LIST OF ILLUSTRATIONS

Introduction

The development of data analysis has paralleled the rise of empirical science itself. Modern science is founded upon the idea that theory should be verified against the data obtainable from reality. This inclusive view of the analysis of data has tended to be submerged by the successful development of the theories of probability and statistics, which have turned data analysis into a relatively confined sub-branch of mathematics. The advent of the electronic computer, however, with its great flexibility and liberating power has caused the rediscovery of data analysis as a field in its own right that has much wider goals and fewer restrictions than either mathematical statistics or probability theory.

To get a feel for the rapid changes in attitude that have occurred recently, listen to the pioneer John W. Tukey (1962, p. 1):

> For a long time I have thought I was a statistician, interested in inferences from the particular to the general. But as I have watched mathematical statistics evolve, I have had cause to wonder and to doubt. And when I have pondered about why such techniques as the spectrum analysis of time series have proved so useful, it has become clear that their "dealing with fluctuations" aspects are, in many circumstances, of lessor importance than the aspects that would already have been required to deal effectively with the simpler case of very extensive data, where fluctuations would no longer be a problem. All in all, I have come to feel that my central interest in is data analysis, which I take to include, among other things, procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of mathematical statistics which apply to analyzing data.

This statement, first presented in 1961, is still closely bound to the traditional notions of statistics, as it is a description of what statisticians did as opposed to what they said they did. A short time later Tukey recognized (1966, p. 695) the generality and independence of data analysis and had progressed far beyond the narrow confines of conventional statistics:

> The basic general intent of data analysis is simply stated: to seek through a body of data for interesting relationships and information and to exhibit the results in such a way as to make them recognizable to the data analyzer and recordable for posterity.

CHAPTER I

THE ESSENCE OF DATA ANALYSIS

> Data analysis is that coordination of continuing observation and developing theory which produces information.

Data analysis is the activity of interrelationship between ongoing theory and ongoing data: it is neither the theory-changing nor data-gathering process. Modern trends in the philosophy of science match this view that the existence of reality, and with it the notion of truth, is irrelevant. Data analysis does not result in true theories, only informing ones.

This use of "information" is non-standard. Both theory and data are required to produce information. Theory without data is so unsubstantiated as to be empty. Data without theory is meaningless. In tying data to theory, data analysis gives the confirmation of data to theory and the interpretation of theory to data, and creates information.

Now a theory is linguistic in nature: a set of statements in some language. One might prefer "conceptualization" instead, but this is illdefined and unmanageable. Theory is the tangible, manipulable form of conceptualization, insight, understanding, and explanation.

And what is data? Data is not linguistic, not sentences of a language. Data is a structured body of facts, a tabular listing of terms. Today data is epitomized by the computerized data bank.

The distinction between extension, the data structure, and intension, the theory, is the first fundamental duality. Data analysis is the bridge between them. It resembles the double helix, that foundation of life as we know it. If one strand is the activity of observation and the other is the unfolding process of conceptualization, then data analysis bonds the two, holding them together and conveying their recriprocal influences.

The Role of Language

When one is faced with a body of data, one conceives his task to be finding relationships which are substantiated by that data. One searches for those models, or sets of structural relationships, which best reflect the data. Some would like to think that the data analyzer has at his disposal all possible structures or models - this is not the case.

"All possible models" is far too large a class and in fact is philosophically treacherous. In any particular case the analyzer is limited, limits himself, to a much more detailed and circumscribed set of models. These are the ones compared with the data. Thus another aspect of the task is to determine the modelspace, the set of models, to be considered.

Equivalently, since a theory is embedded in a language, one must determine the language in which to express the theories to be given attention. The division between language and theory,

or modelspace and model, is the second fundamental distinction of data analysis.

An illustration can be formed from the relationships given by a family tree. When a researcher knows that his data is about family relationships he will use such terms as father, mother, and grandparent, and will state such particular data as "John is the father of Mary," fully understanding the meaning of these terms. These phrases, together with some knowledge of their meanings and interrelationships, form a language which he uses to describe certain worlds.

Tacit Knowledge. Which models are available in this language? It is clear that using such terms one cannot describe any model whatsoever. Since the words of the language include tacit knowledge, we find that language delimits the set of models we can consider and this very restriction adds knowledge which would otherwise not exist. Thus if we assume, either apriori or by explicit statement, that parent and child are related in the normal fashion, then from the data "John is the father of Mary" we can know that "Mary is John's child." This new understanding is attainable only because we have eliminated many possibilities and thus have some restrictions on the models involved. This technique of gaining information by restricting the possibilities considered is tremendously powerful and ubiquitous.

The position of tacit knowledge is generally misunderstood. Consider "John is the father of Sue." Even if the level of implicit meaning that relates father to child is ignored, there remains a basic understanding of the structure of the sentence itself: it establishes that a relationship, namely father, exists between John and Sue. It is only in terms of these understandings of language and language structure that data is in any sense meaningful. Even when the data is given in the form of tabulations, without some prior understanding of how the forms of these tabulations are to be interpreted, of the significance of the symbols used, and so on, the data would be complete nonsense.

Let us examine in greater detail the implicit knowledge tacit in language itself. One's ontology - what types of things one believes can exist - determines to a large extent what things one looks for, pokes and examines, or considers errors in measurement rather than data. To see that these metaphysical assumptions affect our perceived reality reconsider family relationships. We know that a father is male and a mother female, and every person has one of each. Yet in certain primitive cultures a person might have two female parents, one the mother and the other the father. Further, in our own society, artificial inovulation makes it possible for a person to have two "real" parents and a third woman for a mother.

It can be argued that the above is merely a change in the meaning of the words. This is a change of language (in the broad sense of language) and as such is a very definite change in the assumptions and knowledge we bring to a situation. The informativeness of data is affected rather directly by this kind of change.

The linguist Benjamin Whorf expresses (1956, p. 212) the role of language quite forcefully:

> When linguists became able to examine critically and scientifically a large number of languages of widely different patterns, their base of reference was expanded; they experienced an interruption of phenomena hitherto held universal, and a whole neworder of significances came into their ken. It was found that the background linguistic system (in other words, the grammar) of each language is not merely a reproducing instrument for voicing ideas but rather is itself the shaper of ideas, the program and guide for the individual's mental stock in trade. Formulation of ideas is not an independent process, strictly rational in the old sense, but is part of a particular grammar, and differs, from slightly to greatly, between different grammars. We dissect nature along lines laid down by our native languages. The categories and types that we isolate from the world of phenomena we do not find there because they stare every observer in the face; on the contrary, the world is presented in a kaleidoscopic flux of impressions which has to be organized by our minds-and this means largely by the linguistic systems in our minds. We cut nature up, organize it into concepts, and ascribe significances as we do, largely because we are parties to an agreement to organize it in this way - an agreement that holds throughout our speech community and is codified in the patterns of our language.

The language as a whole encapsulates tacit knowledge. Statements in the language, "theory," extend this in an explicit way. While even the level of meaning assumed in the language

may go beyond or be inconsistent with the data, presumably we start with a language which does not do this. But the theories which extend the tacit meaning may well.

One can think of the sentences that make up a theory as specifications of certain aspects of the world. As such, each statement further restricts the class of possible models. In general, one would hope to have a theory which so restricts the possibilities that there would be one and only one left - this, then, would be the "true" theory of reality. Unfortunately, no data is complete enough to confirm such a theory, thus theory must be weaker.

Data as Theory. There is at core a language in terms of which the data is stated. But the languages we use to deal with data are far richer than that minimally necessary for the statement of the data itself. We can describe further, more complex relationships that may or may not exist in the data. We can account for processes that reduce the data into other forms. Moreover, the language could have potentially stated items of data incompatible with those that may have been given, or which may extend or modify the original data.

Any set of statements in a language is a theory. The data translated into statements form a theory, but a terribly weak one. Dr. Richard Feynman, Nobel Laureate in physics, puts the matter (1965, p. 76) this way:

> How is it possible that we can extend our laws into regions we are not sure about? Why are we so confident that, because we have checked the energy conservation here, when we get a new phenomenon we can say it has to satisfy the law of conservation of energy? Every once in a while you read in the paper that physicists have discovered that one of their favorite laws is wrong. Is it then a mistake to say that a law is true in a region where you have not yet looked? If you will never say that a law is true in a region where you have not already looked, you do not know anything. If the only laws that you find are those which you have just finished observing then you can never make predictions. Yet the only utility of science is to go on and try to make guesses. So what we always do is to stick our necks out, and in the case of energy the most likely thing is that it is conserved in other places.

It is evident that data as theory is too weak. However, theory that goes far beyond data is too unsubstantiated. The "proper" theory is in an intermediate position between the two. We seek a theory that provides the greatest insight adequately confirmed by the data.

Data as Submodel. Another view, complementary to the above notion, is that we seek that model or set of structural relationships which best reflects the data and its interconnections. We begin with some assemblage of models, the modelspace, from which we can choose a model on the basis of our data. The modelspace cannot be the set of all models: the limitations are identical to those imposed by the tacit knowledge underlying a language. For example, social scientists doing regression analysis have confined themselves to linear models of their data. Thus the modelspace limits our alternatives in exactly the same way and for the same purpose as does a language.

Since the data specifies structural interrelationships one can view it as a submodel. As we have seen, the data-submodel does not form a complete model in itself, but only a partially specified configuration of the universe. The modelspace, then, consists of models which extend that partial specification--models which contain the data as a submodel. These are the models which are compatible with the data.

The relationship between a language and a modelspace is quite close: one can derive the modelspace from the language, though not quite the reverse. Consider the models of set theory as "all possible models", at least from a meta-level vantage point. We can say that two of these models are equivalent, to us, if no sentence in a given language can distinguish between them. Thus, no sentence of our language is true in one model and false on the other, or vice versa. In this case the language simply cannot express those features that differentiate the two models. As an example, suppose our language talked about flipping coins. We can express whether a coin lands with either heads or tails showing. What we cannot express or distinguish is the difference between landing heads up on the table or landing heads up on the floor or landing heads up after spinning exactly 101 times. All of these events are identical to our simple heads/tails language.

Therefore, a language clusters models together. The language can be used to distinguish any two models from different

clusters, and cannot be so used on models within the same cluster. Formally, the language has partitioned the models into a set of elementary equivalence classes. These clusters, or equivalence classes, form the modelspace we see when using that language.

This modelspace represents exactly the possible states of the universe--as seen by a particular language or conceptualization. It mirrors the implicit understandings, knowledge, and structural relationships which are tacit in that language.

A language corresponds to a modelspace. A theory, or set of statements, within that language will select one or more of the equivalence classes as being the set of models compatible with that theory. We can correspond theories and models in this way, with a "complete" theory selecting only one equivalence class, or one model in the modelspace.

We will label the language/theory approach as intensional and the modelspace/model approach as extensional. While the two are complementary, their differences are meaningful and will be discussed further in the section below on computer system techniques.

The fabric of data analysis, then, can be torn in two ways by the fundamental distinctions expressed in the diagram below:

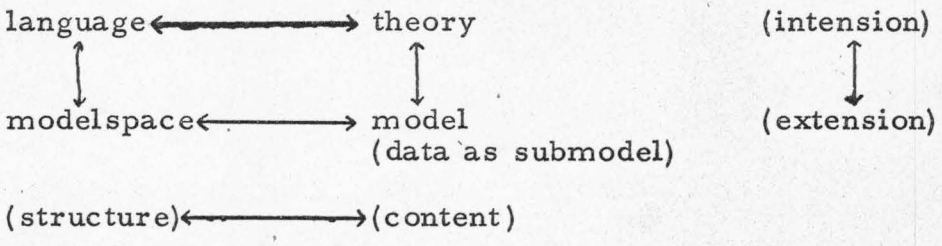


Fig. 1. --The Two Dimensions of Data Analysis

Just as the first distinction can be considered intension vs. extension, the second can be thought of as structure vs. content. The difficulty is that there is no adequate definition of structure. As used here it means the commonalities found in a set of models, or the organization abstracted from some particular set of data. It is in this sense that we utilize a set of models, for the language/modelspace gives us a means to manipulate structure. Marshall McLuhan pursues a somewhat similar idea in distinguishing media from message.

The importance of structure is now being realized. If the milestones of computing history were to be enumerated, most computer scientists would agree on (1) the notion of a stored-program machine and (2) the notion of list-processing techniques. Information scientists, however, would subsume list-processing under the idea of structure processing in general, for we are becoming aware that the limitations of our programs are set by the structures we utilize much more than by any other factor. Furthermore, by designing programs to handle some particular structure rather than a very specific set of data, we acquire more widely applicable programs. One can in fact go to the extreme (but logically correct) position that all our computers do is convert from one structure to another and therefore should be called structure-processing machines rather than data processing machines.

We impose structure on our universe. But the structure of our observations is too weak to be of any use; enormous structure far beyond our data has too low a confirmation. We need theories and models which are in between.

What we do is build theories which are informing: which are compatible with our data and which go beyond the data in delimiting alternatives. This notion that our theories are not totally implied by our data disturbs people, for it insists that "totally objective science" does not exist. It means that in all human endeavors we impose our own subjective views on our perceptions and that if we wish to be informed we must be artists. But artists and scientists combined, for there are the two aspects to information: the side which compares theory to data in order to maintain compatibility, and the side which adds subjective structure in order to delimit the alternatives to be considered.

This imposition of cognitive structure on observation means that one can no longer believe in the primitive scientific ideal: one merely looks at nature (in this case some data) and all will be revealed, for all scientific laws are inherent in the data waiting for us to elicit them. This naive view has been supplanted by one in which scientific laws are the product of our perceptions and of our own thinking process, and are informing at the moment.

Information, Language, and Data

The notion of information has been woven throughout the preceding discussion; with it has been the assumption that information is a function of both language and observation. While the formalization is beyond the scope of this work, it is possible to outline some characteristics of this function.

The word information reminds us immediately of the existence of information theory, as communication theory has come to be known. This branch of probability theory, founded in 1948 by C. E. Shannon, is concerned with the likelihood of the transmission of messages when they are subject to certain probabilities of transmission failure, distortion, and accidental additions called noise. The notion of information quickly appeared as workers in the field tried to express what it is that is communicated, and was just as quickly given a mathematical definition which fits the context of communication theory.

The technical definition of information in communication theory is an attempt to measure the worth or value of receiving any particular message from some fixed set of messages (Pierce 1961, p. 23):

> In communication theory we consider a message source, such as a writer or a speaker, which may produce on a given occasion any one of many possible messages. The amount of information conveyed by the message increases as the amount of uncertainty as to what message actually will be produced becomes greater. A message which is one out of ten possible messages conveys a smaller amount of information than a message which is one out of a million possible

> messages. The entropy of communication theory is a measure of this uncertainty and the uncertainty, or entropy, is taken as the measure of the amount of information conveyed by a message from a source. The more we know about what message the source will produce, the less uncertainty, the less the entropy, and the less the information.

This tremendously successful conception of information has one important point: the amount of information depends heavily upon the characteristics of the set of alternatives from which the message is drawn. In fact, the amount of information conveyed by a message is defined as the difference before and after its receipt of our uncertainty about the message space. Thus the central concept of information is the space of alternatives and its probability distribution.

What is the alternative space in a given situation? Communication theory was first applied to telegraphy, whose space was obviously the alphabet, numerals, and a few punctuation characters. These few characters were encoded into dots and dashes for transmission, and one could determine the amount of information a particular sequence of dots and dashes represented.

One must be careful however. Morse originally devised a coding of words from a dictionary into dots and dashes - a radically different space of alternatives. One can receive a sequence of signals and compute many different amounts of information represented by that sequence, one for each alternative space or even one for each probability distribution on the same alternative

space. Communication theory limits itself to a known and fixed alternative space and probability distribution.

What has all this to do with data analysis? First, one can certainly think of data as a message, perhaps received from Nature over a noisy channel. One would obviously like to know how much information that data contained. If we had an alternative space the whole of communication theory would be applicable, and presumably we could compute the information.

The problem, of course, is the space of alternatives. Here, as should be guessed by now, is the function of language. A language determines a modelspace, as shown previously, which is exactly the set of alternatives needed.

Therefore, a language and a set of data together determine the amount of information. Given a body of data, one can search for that language which maximizes the information associated with that data. Given a fixed language, one can search for that data which is most informing within that conceptualization. We maximize our information by adjusting both language and data as necessary.

The case of a single, fixed language is exactly that covered by communication theory. More interesting is the extension of the notion of information into the realm of many languages, conceptualizations, alternative spaces. We will refer to a conjecture concerning this area, enunciated by F. B. Thompson, as the Fundamental Theorem.

Consider the case of a fixed set of data and a linear chain of languages. The languages form a "proper" chain, that is, any language in the chain is a proper ramification of the languages to its left. More formally, we can induce a partial ordering on the set of all formal languages by this definition: if L_1 and L_2 are formal languages, then $L_1 \leq L_2$ if the modelspace associated with L_2 is a refinement of the modelspace associated with L_1. That is, some model possible in L_1 has been ramified into several distinguishable models in L_2.

Two properties of such chains of languages are worth noting: every proper chain has a right-hand end, and none have a left-hand end. The right-hand end language is one which creates a modelspace with only one model - it cannot distinguish between any states of the universe. Such a language might consist, for example, of the one word "wow." The fact that there is no most powerful language is essentially Tarski's theorem on truth: for any formal language L_1, there is a more powerful language L_2 which can express things not expressible in L_1.

Thus, if we make our chain of languages the horizontal axis of a graph, and a measure of the amount of information given by a fixed set of data the vertical axis, we should at least be able to see the shape of the curve even if we cannot give explicit formulae for its computation. There is one point worth noting on the language axis. We will assume that there is a least powerful language in

which all aspects of the given body of data can be expressed, and mark that L^*.

We know that the information provided by our one-word language is zero, since no data affects what we can do with it. As for the rest of the curve, the standard expectation assumed in the literature is that information increases until L^*, at which point everything knowable is known, and is thereafter constant since one does not lose information already gained by being able to express more. This curve is depicted below.

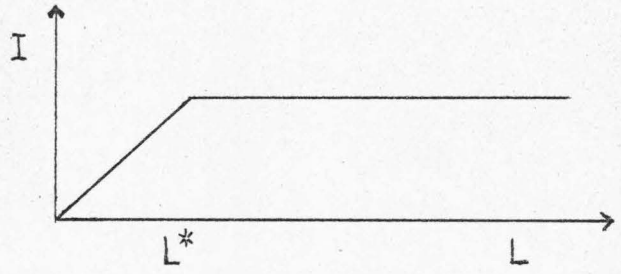


Fig. 2. --"Objective reality" information curve

This view encourages the use of low-level languages and conceptualizations - at least as ramified as L^*. More importantly, it says one can go much lower without loss of information; thus biologists and psychologists should be thinking in the same terms as atomic physicists, for example. This view of the information curve supports a reductionist philosophy.

The Fundamental Theorem has two parts:

1) if one considers more and more powerful languages, in the limit the information obtained is zero;

2) there are languages to the left of L^*, i. e. abstractions from L^*, which maximize the information across the chain of languages.

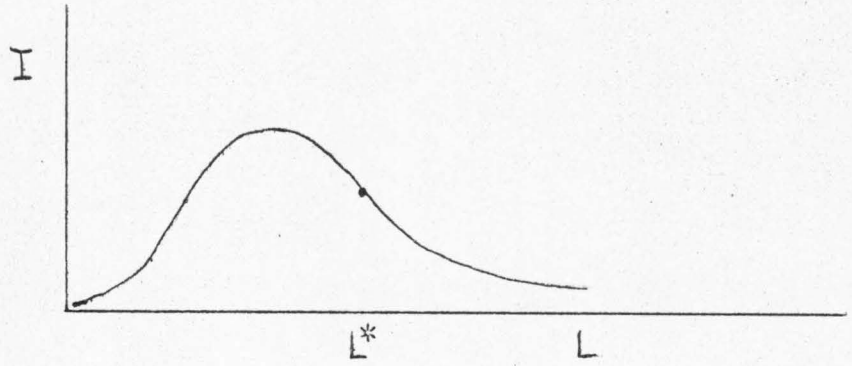


Fig. 3. --The information curve of the Fundamental Theorem

While this theorem has not yet been formally proved, there are good reasons for expecting it to be true. For more details, see Thompson (1969) and Randall (1970).

The Fundamental Theorem implies that we are most informed when working at a fairly high level of conceptualization - more abstract than the level of the raw data, and certainly not at some extremely ramified common, basic language. At the same time one cannot get too far above the data.

One must search for an informative conceptual view. But all languages are informative to some degree. The importance of the Fundamental Theorem is that it tells us to search for the most informative conceptual view, that in fact one exists. This view, furthermore, is not at the level of our sensory impressions or some other "objective" level. It is a view in which we have gone beyond the data, made inferences, and imposed our own will.

What is data analysis? On one side is the activity of finding an informative language and theory within that language. On the other is the continuing activity of perceiving and data gathering. Both of these processes affect the other enormously, and data analysis is the bridge that intervenes. Data analysis should tell us when to move to a more informing theory, and when to gain information by changing the data we work with to bring it closer to current language and theory.

Data analysis is that connection between advancing cognitive structuring, on the one hand, and continuing perception, on the other, which produces information.

Statics of Data Analysis

If information is a function of language, one might well ponder the use of theory. Theories are necessary for the process of confirmation: one can compare data to a theory, not to a language. Theory is our bridge between data and conceptual view.

Given a language and a body of data how does one choose which theory within the language will be used as representative and compared to the data? We would like to choose the best one, the theory which fits the data most closely, out of the possibilities provided by our language. In most languages, however, there is hardly a notion of fit, and no apriori meaning for "best fit."

Mathematical Statistics. Fortunately, for a very few simple languages we can define "fit" and elicit the best-fitting theory. Mathematical statistics is that subject which describes these languages, a notion of fit, and procedures for finding the best theory in the language. The languages involved are all numeric, and in general are the ones which are mathematically tractable.

For example, one of the most frequently used languages talks about lines: linear functions of one real variable. All sentences in the language are of the form "Y = < number$_1$ > + < number$_2$ > * X"; any such sentence can be considered a theory. The associated modelspace is the set of all non-vertical lines in a coordinate plane--every line corresponds to a sentence and every sentence specifies a line.

For this language and a set of data (pairs of numbers $< x_i, y_i >$) one can define the best-fitting theory. It is that theory whose values of <number$_1$> and <number$_2$> are such as to minimize the function $E(a, b) = \sum_i (a + bx_i - y_i)^2$

This definition is then used to find the best-fitting theory - the coefficients which minimize the error function. Obviously, this procedure is curve-fitting with a least-squares criterion, and in this case finds the regression line.

The point here is that most of descriptive statistics can be rephrased into the following form: "if the data is of type such-and-such, and we consider only a particular language, then (1) a good

notion of fit is _____, and (2) using this notion, the best-fitting theory can be found by _____".

Thus mathematical statistics is an important part of data analysis and is limited only by the languages and modelspaces it considers.

The Uses of Fitting. Suppose we step back a little and consider what people think they do. We find two rather distinct groups. The first, consisting mainly of statisticians, advocates the use of models to analyze data. The second group, the simulation users, champion the use of data to analyze models. Are these opposing philosophies?

The people who use models to analyze data talk in terms of "fit": how well does the model fit the data? The viewpoint here is that it is the data which is important; they desire techniques and tools that summarize the data and display the interesting relationships in the data. Models, from this point of view, are simply structures that guide data analysis. They are assisting tools, and one should never completely believe in them. Tukey (and Wilk 1966, p. 796) puts the matter this way: Data analysis "can only bring to our attention a combination of the content of the data with the knowledge and insight about its background which we must supply. Accordingly, validity and objectivity in data analysis is a dangerous myth."

The main tools of this group are fit and exposure. They fit a model to the data, then consider the residuals - those places and instances of lack of fit. Again according to Tukey (1966):

> The iterative and interactive interplay of summarizing by fit and exposing by residuals is vital to effective data analysis. Summarizing and exposing are complementary and pervasive. . . . The single most important process of data analysis is fitting. It is helpful in summarizing, exposing, and communicating. Each fit (1) gives a summary description, (2) provides a basis for exposure based on the residuals, and (3) may have the parsimony needed for effective communication.

In this type of data analysis, while the focus is on the data, we use models and theories as tools to get at the relationships that hold between the various elements in the data. Thus our eyes are on the data and our hands can be manipulating theory.

The other school says that one uses data to analyze models. In this case, people generally have some theory and wish to verify the correctness of that theory against some "real world" data. This is the problem of verification of theory to increase the credibility of theoretical construction.

The view that science proves theories to be true by verifying them has passed its day. The question has instead become "how much should one believe in a given theory?" This is one of the main concerns of the people who design and experiment with simulations:

> A simulation or game is the partial representation of some independent system. Usually we are interested in simulation as a means for increasing our understanding of the system it is intended to copy.

> Therefore, the representativeness of a simulation or game becomes extremely important in assessing its value. The process of determining how well one system replicates properties of some other system is called validation. In experimental research, validity is the goodness of fit or the correspondence between phenomena produced by two sets of properties. (Hermann 1967, p. 216)

> To gain confidence in his simulations, the social scientist may check them against scholarly work in general. Further, he should compare his constructions with "realities" - empirical descriptions of the world of nation-states and international organizations . . . . However, a simulated construction is but theory. It provides no shortcut or magical route to the "proof" of the validity of the verbal and mathematical components it contains. Thus, there is a need for a systematic examination of the extent of the congruences between empirical analyses of world processes and simulations of international relations. (Guetzkow 1968, p. 202)

While these two viewpoints seem to be in opposition, it should be clear that both are sub-processes of what we call data analysis. They are both involved in the relationship of data to model, data to theory. The difference is that one side emphasizes data as being more important, the other side emphasizes the model. This unbalanced attention determines and is determined by the researcher's relative reluctance to change one or the other.

If one looks closely enough, of course, one can see the two schools overlap: Tukey, primarily a data man, says, (1966, p. 698): "Even when used for confirmation alone, data analysis is a process of first summarizing according to the hypothesized model and then exposing what remains, in a cogent way, as a basis for judging the adequacy of this model or the precision of this summary, or both."

Clearly data analysis encompasses both viewpoints and more: it is the dynamic balancing of the activities of perception and cognition.

Dynamics of Data Analysis

One of the criticisms of statistical decision theory is that the set of alternatives open to a decision maker is assumed to be fixed. The importance of this static nature is only now being recognized:

> Much of the impetus for the computerization of managerial decision making came from operations researchers who saw the power of certain optimizing techniques and recognized that most managers could not hope to find the best answers to their problems without the assistance of certain sophisticated mathematics. . . .
>
> However, an answer can be "optimal" only if the range of choices considered by the manager is restricted. Let me illustrate: A manager who is being "eaten alive" by carrying costs on his inventory might be told by a bright young operations researcher (or a computer printout) that he should order items into his inventory in optimal lot sizes. There is a nice little square root formula that tells him how to determine the optimum. Suppose a lot size of 162 is the optimal answer to the mathematicians question, "What is the optimal lot size?" This may solve one facet of the problem, but it is not necessarily the best answer to the manager's question, "What should I do about my high inventory carrying costs?"
>
> The best answer for him might be: (a) hold a fire sale; (b) put a new roof on the warehouse to stop parts from rusting; (c) hire new design engineers who can standardize the parts; (d) fire the accountant who treats this account as a place to dump other costs; (e) change the reorder points; or (f) instruct the inventory clerk on corporate goals! (Jones 1970, p. 76)

Enlarging the space of alternatives is one of the means we have for changing a situation in order to gain information. The possibilities for change form the dynamics of data analysis and have too long been ignored.

There are three types of change which can be considered from our conceptual vantage point: (1) one can change his data; (2) one can change theories within the basic framework of some fixed language; or (3) one can change languages.

That one might change his data seems improper and is often referred to as unscientific. Historians of science, especially Kuhn (1970, p. 135), have investigated scientists at work and have actually found enormous amounts of selected purging of old data, usually in times of revolutionary science. Further, the gathering of new data is always under the guidance of the current conceptualization, including when and how. In statistical analysis there are special techniques which justify the elimination of unwanted data by labeling it error or "outlier."

The dynamics of the situation are such that at times the current conceptualization is more valuable or more believed than data which raises questions about it, and so that data is ignored or dropped. This may be used to increase the information associated with that conceptualization/data pair.

Change of theory is a relatively well-understood phenomena; statistical decision theory is applicable, for example. We wish to choose that theory from among the possibilities created by our

language which is best confirmed by our data, which fits best, or which has the best expected payoff. There may be technical difficulties in finding such theories, but for a wide variety of theories the techniques of curve fitting, the calculus of variations, or dynamic programming are effective. The problems intensify, of course, as the theories involved become more complex.

The conceptual problem, and our lack of understanding, of language change is greater. We can identify several instances of the general notion of language change.

If some part of a theory becomes very highly confirmed, it is usually more informative to shift the explicit structure of this subtheory to implicit structure within a language. That way one assumes something that was once questioned and considered. An example is the belief that physical laws can be stated mathematically. This notion was once as controversial in physics as it is today in the social sciences.

Other types of language change can have even greater effects. There is change which admits the existence of new conceptual entities. The existence of forces-which-work-at-a-distance was a revolution in physics, as was the emergence of aristocrats in social philosophy.

There is also language change which adds new alternatives, exactly as in the above example of the manager making decisions. This kind of language change ramifies structure - creating several

alternatives where before there was only one. Abstraction has the opposite effect; it consolidates many alternatives into one by ignoring differences. The concept of "people" ignores many individual differences in favor of certain commonalities. The concept of sex subdivides the class of people by emphasizing certain differences while excluding others. The dynamics of conceptualization is often a pattern of alternation between abstraction and ramification.

Note that change of language implies theory change as well. A theory, as a set of sentences within some language, is interpreted only with reference to rules contained within that language. Even if the explicit statement of a theory does not change, its meaning can.

Think of the theory of physics, part of which is contained in the statement, "all of the properties of the world can be accounted for as interactions between atoms." When atoms were defined as indivisible, basic particles, physicists conducted certain experiments to determine their characteristics, for example the implication that chemical reactions occur with small-integer weight relations.

Now, however, when an atom is a collection of further particles and forces, the operational meaning of the above statement is quite different. "Atom smashers" were self-contradictory in pre-subatomic days.

Language change is more difficult than theory change since it entails the latter. It therefore occurs less frequently and with greater effort and attendant confusion. This more basic kind of change affects the unspoken assumptions of a field so that communication may be disrupted. In data analysis this relative difficulty also holds. A social scientist doing correlations and regressions is working within the language of linear models. To switch to general polynomial models requires major adjustments in technique, interpretation, and theory.

The importance of language change in the dynamic aspects has already been recognized. Thomas Kuhn, in his work on the nature of scientific progress (1970), distinguishes normal science from revolutionary science. We can identify normal science as theory change and revolutionary science as language change.

Kuhn's thesis is that normal science means working within a "paradigm", while revolutionary science changes paradigms. Kuhn's paradigm is our notion of language. Paradigms are works which (p. 10) "served for a time implicitly to define the legitimate problems and methods of a research field for succeeding generations of practitioners. They were able to do so because they shared two essential characteristics. Their achievement was sufficiently unprecedented to attract an enduring group of adherents away from competing modes of scientific activity. Simultaneously, it was sufficiently open-ended to leave all sorts of problems for the

redefined group of practitioners to resolve." Although Kuhn is concerned with major upheavals, his notion is close to our conception of language, within which there are many theories.

On revolutionary science, or important language change, Kuhn (p. 84) writes:

> The transition from a paradigm in crisis to a new one from which a new tradition of normal science can emerge is far from a cumulative process, one achieved by an articulation or extension of the old paradigm. Rather it is a reconstruction of the field from new fundamentals, a reconstruction that changes some of the field's most elementary theoretical generalizations as well as many of its paradigm methods and applications.

Conceptual Frictions. A discussion of dynamics would not be complete without some thought given to the frictions which are inhibiting conceptual change. The following is superficial, yet does represent a beginning on this complex subject.

We can classify the inertias into three broad categories: informational, psychological, and technical. Psychological resistance to change is the best documented and studied. In this domain, anxiety is a major cause for mental rigidity. An anxious person seems to lose the ability to move in the abstraction/ramification dimension, to a degree dependent on the level of anxiety. Psychologists are investigating this aspect of anxiety now.

Another psychological friction is reluctance to change solely because of the previous level of investment. The investment could be in terms of money, time, mental effort, or any such scarce resource. When one has a lot invested in some conceptual view,

one tries to retain that view if at all possible. These views are usually abandoned much later than they should have been, only when their use is a total catastrophe. This effect is visible today especially in societies, government structure, and computer systems.

Under technological frictions are classified all inertias imposed on us by our use of current technology. There will always be technological friction, since one's technology forms a part of one's reality. Some forms of technology are less limiting than others, though. The electronic computer has the potential to enormously facilitate our conceptual movement. The present usage of computers, however, does not. Chapter II of this dissertation provides the details on the current computer practice in data analysis systems.

Informational frictions are those related to the nature of information and the conceptualization process. First, suppose that we attempt to find the most informing conceptual view. That is, given some starting view, we move in the direction of increasing information: information hill-climbing. But there is a trap here: we may find a language which provides a local maximum, in terms of information. All of its neighboring languages have less information, even though some other languages provide more. One would be reluctant to change conceptual views if it meant a loss of information immediately and a possible gain later.

The second type of information friction is related to the need for communication. Communication between two individuals

can only take place if they share some conceptualization. Remember that we equate conceptualization to language: the individuals communicating must be talking the same language. What happens if one person changes his language? Either the amount of communication drops, to that part of the language still held in common, or the other person must adjust his conceptualization to match. The painfulness of this process is evident, and one can cite many examples of its effects. A simple one is the frantic effort to standardize programming languages such as FORTRAN, COBOL, or BASIC.

These inertial forces in the dynamics of conceptual change constitute an interesting and important area of research for the behavioral sciences. A much deeper understanding of them is essential to a thorough treatment of data analysis.

CHAPTER II

THE CURRENT STATE OF DATA ANALYSIS SYSTEMS

A data analysis computer system is certainly a repository for data, but it is also something more: a medium for the articulation of conceptualizations. Since data analysis is the interaction between cognition and perception, the primary goal of these computer systems is to encourage and provide support for this interaction.

Data analysis systems must aid both sides: the ongoing processes of data collection and theory building. Furthermore, these two processes must be in harmony--neither can be neglected or overshadowed.

There is an important point to be made about computer aids for conceptual developments. Computer systems are always a resistance to conceptual movement. They are, after all, only recursive mechanisms. Beyond this, however, various types of systems have their own rigidities. These restrictions exist because of the incorporation of meta-level conceptualizations (the system designer's) and current technological limitations. Any particular system represents a balance between the conceptual and technological efficiencies obtainable by imposing limitations and the inhibition of conceptual freedom that such limitations require.

The basic questions to be asked about current systems include: (1) what range of user conceptualizations does the system allow; (2) how does the system facilitate the user's movement through that conceptualization space; (3) in what ways does the

system aid the process of data adjustment; and (4) does the system balance data and theory?

Today there are five identifiable system types being used for data analysis:

1. data management
2. statistical analysis
3. question-answering
4. reference retrieval
5. simulation

Data Management Systems

These are the systems designated by some combination of the terms data, information, file, retrieval, management, and generalized. There are currently around 200 distinct systems in existence; the system type is being studied intensively by a CODASYL committee (1969).

Data management systems are an evolved form of the 3 x 5 card file. This extremely useful device is typified by the card catalogue in a library. The catalogue consists basically of a file of cards, one for each book in the library. Each card contains all the data pertaining to one book, such as its title, call number, author, etc. There are also auxiliary files, containing such things as cross-references which facilitate certain types of searches.

In current data management systems a data base consists of a set of files, each a sequence of records. All records in a file have

a similar, fixed format and all contain data about a single type of entity. Each record contains essentially all of the data about an entity, and ideally there is only one record per entity. Finally, some of the newer systems have added auxiliary files of indexes, using the notion of the "inverted file," in order to facilitate certain kinds of searches.

Using these data management systems, one could, theoretically, display the record of a single, particular entity. Instead, one usually produces a "report," a display of a specified portion of the record for all those records satisfying some selection criteria. An example of such a report, and of these systems, is shown below in Figure 4. An understanding of the nature of data management systems requires a look at the restrictions placed on the selection criterion. The decision of whether to include record X must be made on the basis of data contained only in record X. That is, the selection criterion is a recursive function of data in the given record exclusively.

This limitation enforces a worldview that each entity, i. e. record, is basically independent of every other entity. What sorts of user conceptualizations are allowed in this environment? The only theories permitted are those that state that some entities really are related and are interesting: all those which pass some stated selection process. Thus the space of theories is generated by the set of allowable selection functions, which are limited as described above.

The data management systems facilitate the user's movement through this conceptual space both by providing such a simplified space and by making it relatively easy to describe the selection function desired. The tremendous proliferation of such systems provides proof of the effectiveness of their conceptualization and implementation technique.

The limits of their applicability are equally clear. These systems assume a basically static language, that is in this case a basic set of data attributes. Modification of data in existing records is tolerated, as is the addition of new records within an existing file. Barely tolerated, since, as a typical example of the common use of such systems, a change-of-address on a magazine subscription will take six to ten weeks.

As for more fundamental changes, the addition or deletion of an attribute across an entire file for instance, these require major upheavals in conceptualization as well as considerable time and effort. Data management systems are counterproductive in a dynamic or highly interrelated world.

Fig. 4.--An example of data management systems

> Suppose that one wanted to create a file with a record for each member of the computing center staff, giving his name, date of employment, and principal programming language. Suppose that one also wanted a list of all the PL/I programmers on the staff. The following ASAP program would accomplish this (Conway, Maxwell, and Morgan 1971, p. 13):

```
)) ASAP     START RUN:   NEW, DEFINE
))  ASAP    'PASSWORD'
)) DEFINE RECORD:    STAFF
))          NAME 30 KEY
))          DATE_OF_EMPLOYMENT 8
))          PRINCIPAL_LANGUAGE 20
))  DEFINE INPUT:     STAFF CARD
))            COLUMNS 2-31 = NAME
))            COLUMN 1 = NEW RECORD
))            COL 32-39 = DATE_OF_EMPLOYMENT
))            COLUMNS 40-59 = PRINCIPAL_LANGUAGE
))  DEFINE END
))
)) FOR ALL STAFF SELECTED BY KEY
))  IN INITIAL_DATA, FORMATTED BY STAFF_CARD,
))    UPDATE RECORD.
))
))  DATA BEGIN INITIAL_DATA
*JONES, WILLIAM              11/23/68FORTRAN
*WILSON, MALCOLM             01/20/69COBOL
*STEWART, PAUL               07/01/65FORTRAN
*HOPKINS, PAULA              10/15/68PL/I
*ABELSON, PETER              02/01/66ASSEMBLER
*CHAMBERLAIN, H. G.          03/01/64PL/I
))  DATA END  INITIAL_DATA
))
))  FOR ALL STAFF WITH
))       PRINCIPAL LANGUAGE = 'PL/I',
))       PRINT A LIST OF NAME, PRINCIPAL_LANGUAGE.
))  ASAP END, ASAP END RUN
```

Statistical Analysis Systems

These systems have been designed to support the mathematical statistics view of data analysis. These packages usually include a primitive data management system with a simple, rigid data structure, and place emphasis on the processes available for analysis and summary. Some of the current systems are OSIRIS (Inter-university Consortium for Political Research), SPSS (University of Chicago), PSTAT (Princeton University), and BMD (UCLA).

The statistical analysis systems make the assumption that their data is a random sample from some much larger (i. e. infinite) population. In this conceptualization only the broad, statistical view is relevant and analysis of individuals is meaningless. Thus these systems have a data structure which can be described as rectangular: a fixed set of entities, a fixed set of attributes (either numerical or character-valued), and each entity is characterized by all attributes. There is no cross-linking of entities.

In fact, in a random sample one does not expect the individuals to be interconnected, and most statistical processes assume independence of individuals. Having related entities implies that the sampling technique was faulty.

Thus the world view presented is one of having a small amount of data taken from a large population. One wishes to discover broad, generalized characteristics of the total population.

In order to do this, the statistical systems provide a set of primary tools. Each tool is a process which imposes some particular conceptualization on the data and summarizes the data accordingly. These basic theoretical views are the usual ones found in mathematical statistics, simple random variables with known probability distributions, for example.

In these systems a user can also express his theoretical view by transformations of the data or by some recursive selection process. Thus a user's conceptual space consists of some fixed set of basic views applicable to recursive transformations of the data. The overriding limitation is that the data must be considered a random sample collected from a large total population.

Fig. 5. --An example of statistical analysis systems (Nie, et al. 1970, p. 54)

RUN NAME	SAMPLE RUN OF THE SPSS SYSTEM
FILE NAME	EXAMPLE2, THIS IS THE FILE LABEL
VARIABLE LIST	AGE, SEX, RACE, INCOME, EDUCAT N
INPUT MEDIUM	CARD
# OF CASES	10
INPUT FORMAT	FREEFIELD
MISSING VALUES	AGE TO RACE (0, 8, 9)/INCOME(7)/ EDUCATN(0)
VAR LABELS	AGE, AGE OF THE RESPONDENT/SEX, SEX OF THE RESPONDENT/INCOME, YEARLY FAMILY INCOME IN DOLLARS/EDUCATN, EDUCATN OF HEAD OF HOUSEHOLD
VALUE LABELS	SEX(1)MALE(2)FEMALE(3)NOT ASCER-TAINED/RACE(1)WHITE(2)NEGRO(3) ORIENTAL(4)OTHER(9)NOT ASCERTAINED/ EDUCATN(1)NONE(2)PRIMARY OR LESS(3) SOME SECONDARY(4)SECONDARY GRADU-ATE(5)SOME COLLEGE(6)COLLEGE GRADUATE(7)GRAD SCHOOL(8)OTHER(9) DON'T KNOW(o)NOT ASCERTAINED

```
PRINT FORMATS     AGE TO EDUCATN (o)
CROSSTABS         RACE BY INCOME BY EDUCATN/INCOME
                  BY RACE BY SEX
OPTIONS           1, 3
STATISTICS        1, 4, 6

READ INPUT DATA

74  1  2  8999  7  64  2  1  7463  4  24  3  1  5000  6
41  3  1  4756  2  87  1  2  2746  3  55  2  4  8468  5
57  2  3  9999  7  25  3  4  5472  1  37  2  3  2757  4
28  1  1  7000  1

PEARSON CORR      AGE TO EDUCATN WITH SEX TO INCOME
OPTIONS           1, 3
FINISH
```

Question-Answering Systems

W. Cooper (1964) first described what is now the standard view of question-answering or fact-retrieval:

> There are two propositions which are plausible in themselves, and which, when viewed in conjunction, focus attention on what we believe to be the fundamental problem of Fact Retrieval.
>
> Proposition I. A Fact Retrieval system must normally accept most of its information to be stored, and also its queries, in the form of natural language sentences (e. g. English) rather than in some artificial language selected for the purpose.
>
> Proposition II. A Fact Retrieval system must possess the capability of performing logical deductions among the sentences of its input language. . .
>
> Together these propositions suggest that the central theoretical problem of Fact Retrieval is to develop a system of logical inference among natural language sentences.

We can categorize question-answering systems into two types. Corresponding to an intensional view are the deductive systems, to an extensional view are the relational systems. A third type, the semantic net system (Quillian 1969), is an interesting and novel attempt to combine an intensional view with an extensional structure.

Deductive Question-Answering Systems

Deductive systems have evolved from artificial intelligence research on finding deductive proofs of mathematical theorems. The research has been generalized to deductions in a predicate calculus environment, usually only the first-order calculus. The question-answering systems, then, add to this work a translation of the English input sentences into the predicate calculus, but otherwise use the same techniques.

These deductive systems all assume the intensional view, that is, they manipulate sentences and theories. The approach to deduction is essentially syntactic: new theorems are added to a growing store by grammatical manipulations of the previously existing set. The most efficient current techniques, the resolution methods, do work extensionally by trying to construct models. If an appropriate model cannot be constructed, it proves the falsity of some sentence: usually the negation of the theorem one is trying to prove. However, the elements of these models are sentences

and clauses; one uses the linguistic entities as elements of models in order to construct manipulable models.

There are several interesting aspects to the intensional approach. The first deals with the notion of atomistic completeness. The concept of completeness, taken from logic, is the property of a theory or model that all parts of that theory or model can be derived from some basic set of primitive elements--the atoms of that theory or model. Applied to a data base, this means that all of the data can be derived by application of recursive functions to the atomic elements of the data base.

A good example is the grandparent relation. Suppose that the parent relation is a primitive in some data base. Then one can define the grandparent relation as the composition of parent with itself: "grandparent" means "parent of parent." In this case the grandparent relation is totally dependent on the parent relation and derives all of its characteristics from it; for example, the fact that every person has four grandparents. At this point the grandparent relation has added nothing new, and all instances of the term could be replaced by its definition.

Suppose, however, that one added the datum "the grandparent of Mary is John" and that our data does not include Mary's parents. Now grandparent is de-coupled from parent; it has more properties and relationships to the rest of the world than is implied by the parent relation. This data base no longer has the property of atomistic completeness.

A theory or model can be atomistically complete only if all of its elements are of the primitive, atomic kind, with recursive definitions added for higher-level structures. In an atomistically complete model, with a primitive parent relation, the grandparent relation can be either 1) defined solely as "parent of parent" and thus completely coupled, or 2) defined primitively also, thus completely uncoupled.

All of the current extensional systems are atomistically complete, and it is only the intensional, deductive systems which are not so restricted.

The ability to handle meta-level data gives these systems their great power. The logic of the deductive systems is explicit, and therefore can be manipulated instead of implicit in the processing routines as is the case for other types of systems. An example of this power is the fact that these systems can comprehend data containing quantifiers as primitive items. "At least ten people live in Boston," as data, makes certain kinds of deductions and answers possible, even if we are uncertain exactly who is in Boston.

The cost of this power is clear; deductive systems use a recursively-enumerable search procedure, rather than the recursive procedures found in the extensional systems. By this we mean that the set of theorems in a formal language is recursively enumerable and not recursive. Thus one cannot determine the

truth or falsity of any given sentence directly, but instead one must list all theorems and see if the given sentence is among them. This enumeration technique, the only effective procedure for a recursively enumerable non-recursive set, has been shown time and again to be much slower than a direct approach where that is possible. This relative inefficiency limits the complexity of query and the size of data base allowable. For example, a recently developed system (Biss, Chien, and Stahl 1971, p. 303) works with a data base consisting of 2000 English sentences, claimed to be "larger than any other data base currently being used for natural language [deductive] question-answering systems." This fundamental limitation on efficiency may be bypassed to some extent by a judicious combination of both intensional and extensional approaches, which is the long-range promise of the semantic net systems.

Fig. 6.--An example of deductive question answering.

Suppose the system (Biss et al 1971, p. 305) receives the question: Do cars always have to yield to pedestrians? and it has at its disposal the facts 1) Pedestrians not in a crosswalk must yield to cars and 2) If x must yield to y, then y does not have to yield to x.

The syntactic analysis of the question produces the form: always(must(yield(car, pedestrian))). The semantics of the word "always" converts this statement into:

$\forall$ y(y$\rightarrow$ must(yield(car, pedestrian))), where y is a variable ranging over situations. This is further converted into: $\forall x_1 \forall x_2$(must(yield(x_1(car), x_2(pedestrian)))), where x_1 is a variable ranging over situations on car and x_2 is a variable ranging over situations on pedestrian.

The relevant data has been stored as must(yield((not(in (crosswalk))) (pedestrian), car)) and $\forall x \forall$ y(must(yield(x, y))$\rightarrow \sim$must (yield(y, x))). The system tries to prove the question true by showing that its negation contradicts the relevant axions. This it will not be able to do, and so will eventually try to prove the question false. In this case the system can prove that the question statement itself contradicts the axioms, and so can be answered "no."

To do this the R2 system first rewrites the second axiom as $\sim$must(yield(x, y))$\vee \sim$must(yield(y, x)) since A$\rightarrow$B is equivalent to $\sim$A$\vee$B. Then, $\sim$must(yield(x_2(pedestrian), x_1(car))) follows from this and the question statement by recursively applying high-order resolution. This statement resolves with the first axiom if we let x_2 = not(in(crosswalk)) and x_1 = $\emptyset$ (the empty substitution), generating a contradiction.

The system can also output those situations in which a car does not have to yield to a pedestrian, i. e. the instantiations of x_1 and x_2: when the pedestrian is not in the crosswalk.

Relational Question-Answering Systems

Relational data systems take the extensional approach in order to reach a usable level of efficiency. All of these systems use a single type of modelspace, a relation algebra--a set of entities and a number of relations among those entities. Such systems are exemplified by the Relational Data File (Levein and Maron 1967) and Converse (Kellog et al. 1971).

The notion of a relation algebra is a very general mathematical concept. It is general enough to be used as the basis for mathematical model theory, which underlies the use of the term model in this thesis. One can also consider set theory to be the theory about a particular relation algebra, one with a specified binary relation.

Thus a relation algebra has a wide scope. At the same time its primitives, both entities and processes, are surprisingly simple and few in number. This implies that it should be possible to implement this type of system relatively easily and with a great deal of attention to efficiency. Such implementation details will be considered in a later section of this thesis.

The relational data systems therefore allow rich interconnection among the entities of the model, in contrast to the data management and statistical analysis systems discussed earlier (on this point see Codd 1970). This type of modelspace, however, seems to require atomistically complete models. What, then, are its capabilities for deduction?

One can distinguish two types of capability, that for global deduction and that for local deduction. Global deduction, another term for the usual type of deduction, is an intrinsically recursively enumerable process as discussed above. The set of provable theorems is in general not recursive, but is recursively enumerable. Local deductive capability, i. e. the relational data systems, is the ability to work with recursive subsets of theorems. Local deduction denotes a recursive set of theorems and obviously is more restricted than a full deductive capability.

Some examples are in order. First, suppose one had the following two items of data, "Joe arrived in Los Angeles in 1960," and "Joe left Los Angeles in 1970." What can one say in regard to Joe's whereabouts in 1965? On the basis of the data alone, nothing. One can, however, include in the logic of the language enough assumptions and rules of inference to be able to answer "Joe was in Los Angeles in 1965." These assumptions and rules take the form of recursive functions of the data, built specifically for particular cases.

For a second example of local deduction, consider the ancestor relation (i. e. "transitive parent"). With an explicit logic and relation algebra one could define the properties of transitive relations with axioms and then deduce Joe's ancestors from the data contained in the parent relationship. Local deduction here implies that the meaning of "transitive" is defined by a specific

recursive function rather than by axiom. One could still find Joe's ancestors.

As a final example consider the most important aspect of deduction--quantifiers. We will try to answer the question "Are all men mortal?" by recursive methods. Obviously, if the number of men is finite, one can simply generate each man in turn and ask the appropriate question. But even where we wish to allow the possibility of an infinite number of men, it is sometimes a recursive problem. We might have the class of men a subclass of the class of mortal things, and thus merely re-phrase the question into a simple one about subclass relationships.

The point to be made here is that the relational data systems, at least in their present completely extensional implementations, are limited to local deduction. While local deduction is restricted, it may provide enough power and efficiency for some areas of application.

Thus the relational data systems represent a compromise. They allow a fairly richly-interconnected universe--much more than the data management systems, for example. Yet they are efficient enough to handle reasonably large data bases. A description of such a system constitutes Chapter III of this thesis.

Reference Retrieval Systems

These systems are designed to automate the search of a library's card catalogue, and in general facilitate the search for books and articles dealing with some particular subject matter. Reference retrieval systems are specialized versions of question-answering systems; the restriction of purpose and data is made for the purpose of more efficient operation. Gerald Salton, a leading exponent of these systems, identifies (1968, p. 393) the restrictions this way:

> When comparing reference retrieval and data retrieval systems, the main complications present in the latter (and absent from the former) are caused by the more detailed analysis of the stored data necessary to operate a fact retrieval system. Whereas, for reference retrieval, it is normally considered sufficient to isolate the main objects or entities useful for the specification of the subject content of each stored item (the keywords, concepts, descriptors, etc.), in a question answering system it is necessary also to identify a large variety of functional relationships between entities. Thus, the semantic analysis must be much more thorough, and it must notably include the identification of a majority of the relations indicated in the language by verbs and function words, such as conjunctions, prepositions, and quantifiers.
>
> Furthermore, a reference retrieval system is expected to cope with only one type of question, expressed in terms of a document set considered closed at any given instant, namely "Does the stored collection include items dealing with such and such a subject matter?" On the other hand, a data retrieval system must handle a much larger variety of queries, including also queries for which an explicit answer may not be stored but may first have to be generated from the information actually available.

The goal of reference retrieval is to display all those documents deemed relevant to the subject matter contained in a query. Relevance, of course, is scarcely understood, and so the central notion used in these systems is that of a "concept." If one has an operational definition of concepts, and some way to measure distance between concepts, one can define relevance as a measure inversely proportional to this distance function. The difficult work on reference retrieval consists of defining "concept" and "distance between concepts."

The usual operation of such systems is over some identifiable universe of discourse. First the appropriate concepts are decided upon, and then all documents in the collection are rated on their distance to each concept. Finally, a query is entered into the system and also rated on each concept. Then the correlation coefficient between the query ratings and document ratings are computed for every document in the collection, and the ones with the highest correlations output.

This type of operation, typified by the Smart system of Salton, assumes a rather fixed set of data and certainly a static data structure. In fact, the data structure involved is a sequential file of vectors, one vector per document. Each vector contains the rating of each concept for that particular document and can be (and is) considered to locate a point in n-dimensional space. The query also represents such a point, and relevance is defined by

some n-dimensional distance function. A great deal of work is being done on clustering, that is, representing a group of closely-related documents by one representative description. These techniques are aimed at improving efficiency and especially search times, and lead naturally into other file structures, such as inverted or multilist.

The subject of indexing (i. e. what are concepts?) has been active, breaking into two camps: clustering, where all documents are on the same level, and hierarchical indexing, where abstracted categories are combined in a tree-like structure. Hierarchical indexing appears to be winning in both efficiency and acceptability, especially as the systems become interactive. In fact, the future points obviously toward more general concept structures as the index attempts to mirror our own conceptualization, and therefore toward the convergence of these reference retrieval systems with the more general question-answering systems.

This projected assimilation of reference retrieval systems is caused by 1) the emergence of efficient question-answering systems; 2) the existence of economic interactive computer systems; and most importantly 3) the growing awareness that the user must have a great deal of freedom and control in his conceptualization and search processes.

Fig. 7. --An example of reference retrieval systems

The document collection being searched in this case consists of the 405 abstracts published in the IEEE Transactions on Electronic Computers for March, June, and September, 1959. The collection covers all fields of the computing literature. Sixteen abstracts were manually judged to be relevant to the request. (Salton 1968, p. 467)

The search request:

Give algorithms useful for the numerical solution of ordinary differential equations and partial differential equations and partial differential equations on digital computers. Evaluate the various integration procedures (try Runge-Kutta, Milnes method) with respect to accuracy, stability and speed.

answer	correlation	identification
384 stability	0.8567	Stability of numerical solution of diff. eq.
360 simulate	0.7741	simulating second order equations
386 eliminati	0.7457	elumination of special functions from diff. eq.
392 on comput	0.6571	on computing radiation integrals
200 solution	0.6443	solution of algebraic and transcendental eq.
85 note on an	0.6372	note on analogue techniques for resolving
387 boundary	0.6171	boundary contraction solution of Laplace
103 Runge-Kut	0.5874	Runge-Kutta methods for integrating diff. eq.
102 On the so	0.5648	On the solution of Poisson's difference eq.
390 Monte Car	0.5448	Monte Carlo solutions of boundary value problems

Simulation

A simulation is a tangible, manipulable model that corresponds to a theory about some relevant aspects of the world. Simulations are usually dynamic models, that is, explicitly time dependent, and are therefore represented by processes which operate on some basic structural model. The execution of a simulation calls into play each of these "events," which modify the model in some predetermined way. The main use of such simulations is to unfold the dynamic aspects of a model, especially those models too complex to be adequately handled by formal mathematics.

There are two types of simulations currently receiving attention. The continuous simulations reflect the view that time is a continuous real variable and the processes involved operate continuously and often simultaneously. These models are very often translated into sets of differential equations and solved numerically. Typical application areas might be electronic circuit design, neural network research (e. g. the Hodgkin and Huxley nerve membrane equations), and atmospheric pollution studies.

The second type is called discrete simulation. Here the individual events are considered more important, and are usually distinguishable from each other and are quite complex. The relevant times are only those at which events happen - a discrete sequence of ascending instants. Examples of such simulations

abound in the social and behavioral sciences, such as the formation of coalitions in international politics or the flow of traffic through a city.

The purpose of a simulation is the same as that of any theory. It is used to gain insight into the phenomena under study. Simulations also have the same characteristics as theory: they can be more or less generalized, their primitive entities may or may not be well-chosen for the subject area, they may fit the data more or less closely, etc. The importance of simulation is that they are tangible theory, and thus can be studied, manipulated, and changed.

While simulation is an important vehicle for conceptual development, as a data analysis tool it has one important defect: it underemphasizes data. Simulation is totally overbalanced on the side of theory; any data produced by a simulation, and any data compared to these outputs, are to be utilized by some external process. Simulations merely produce data - what happens to it after that is left to the imagination. What this means, of course, is that a combination of theory-building simulations with a data-oriented analysis system could be extremely powerful. The conceptual pressure for such a combination is increasing, so that it will not be too many years before it exists.

Fig. 8. --An example of a discrete simulation (Gordon 1969, p. 125)

Consider the example of a simple telephone system. The system has a number of telephones, connected to a switchboard by lines. The switchboard has a number of links which can be used to connect any two lines, subject to the condition that only one connection at a time can be made to each line. It will be assumed that the system is a lost call system, that is, any call that cannot be connected at the time it arrives is immediately abandoned. A call may be lost because the called party is engaged, in which case the call is said to be a busy call; or it may be lost because no link is available, in which case it is said to be a blocked call. The object of the simulation will be to process a given number of calls and determine what proportion are successfully completed, blocked, or found to be busy calls.

Suppose each line is treated as an entity, having its availability as an attribute. A table of numbers is established to show the current status of each line. It is not necessary that a detailed history be kept of each individual link, since each is able to service any line. It is only necessary to incorporate in the model the constraint imposed by the fact that there is a fixed number of links. Under these circumstances, the group of links is represented as a single entity, having as attributes the maximum number of links and the number currently in use.

Each call is a separate entity having as attributes its origin, destination, and length. There is a list of calls in progress showing which lines each call connects and the time the call finishes. It will be assumed that the call is equally likely to come from any line that is not busy, and that it can be directed to any line, other than itself, irrespective of whether that line is busy or not.

The simulation proceeds by executing a cycle of steps to simulate each event. The event of disconnecting a call merely updates the status information, while the event of an arriving call must check to see whether the call can be processed, and if so updates records and schedules the disconnecting event. Arrival times, source, destination and length of call are all random variables. Statistics are collected throughout the simulation and after some predetermined elapsed time or number of calls the simulation is stopped and the results output.

The Boundaries of the Practicable

What are the real problems that data analysis systems designers face? It is not in the area of data collection, for our current ability to collect and communicate data overpowers our ability to find appropriate conceptual frameworks for the data (consider the 96,000 reels of magnetic tape holding social security data). Thus our real need is to improve the aid we give to the conceptualization and analysis process.

Static limitations are the size and variety of the models our systems can handle. There are some extremely large sets of data available today, such as the individual-level raw census data, for which no analysis systems exist. It is only at the large size, say the census data aggregated to the census tract level, that either the data management or the statistical analysis systems became useable. Both of these system types permit only simple models in their conceptual space, and so large amounts of data can only be viewed in simpleminded ways.

As the amount of data decreases systems with more complex conceptualizations become effective. A complicated simulation, for example, might have from several hundred to several thousand entities or items of data - a fairly small amount. The deductive systems usually can handle only a few - up to a thousand - axioms and theorems. This inverse relationship between data base size and complexity forms an important boundary on the scope of present activities. Figure 9 attempts to depict this relationship. It is a coarse estimate of the size and complexity capability of each of the contemporary system types. For comparison, the raw United States census data should contain about 10^{10} items of data, and at least have kinship-type interrelationships between the entities.

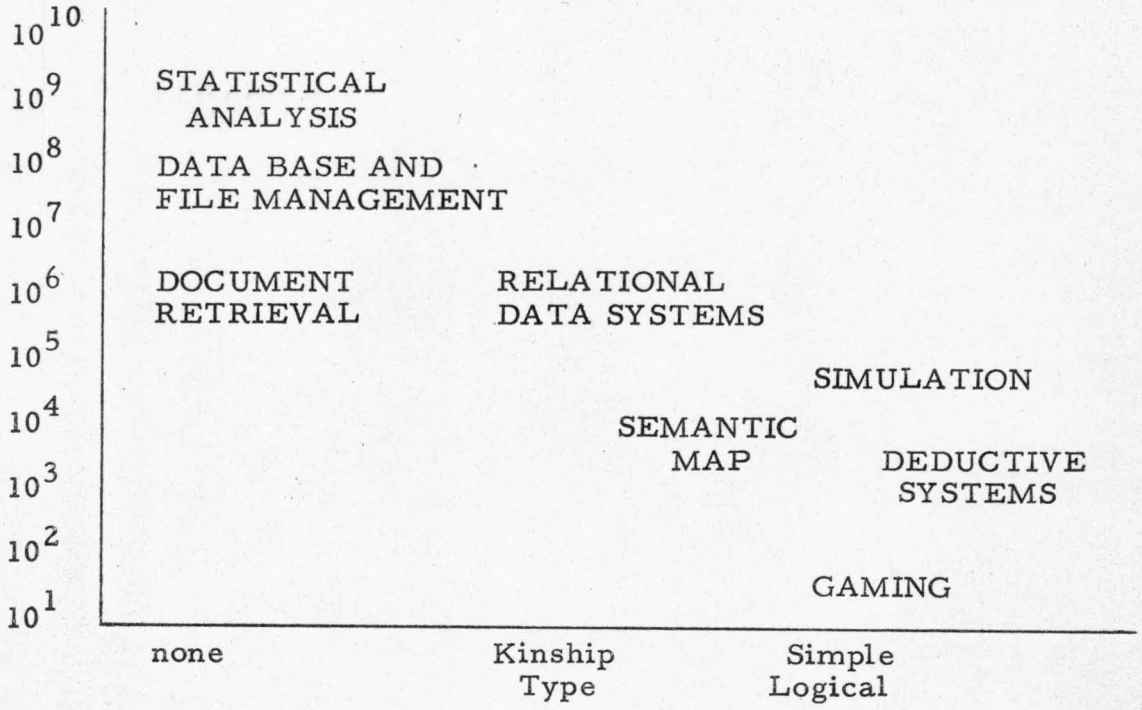


Fig. 9.--The contemporary relationship between data base size and complexity

We have stressed throughout this thesis the importance of the dynamics of conceptual adjustment. The fundamental problem facing systems designers is how to aid such adjustment, not hinder it as do most present systems. The goal: computer systems which help their users find insightful conceptualizations.

Computer systems could help in two ways. Their limitation to some modelspace/language means that we have fewer models to consider as relevant. This pushes the common features of the models into the background, since they are pre-determined. We can concentrate on the differences among the set of models or theories. The system could help us explore these theories, by

making it easier to express them, by aiding in the process of matching theory to data, or even by applying some known optimizing technique.

More difficult and more important are computer systems which facilitate language change. This is what we really need in our very dynamic world. The conceptual rigidity of our computer systems is the significant boundary.

Keeping these boundaries in mind, one can ask where the current thrust of research is heading. The answer, unfortunately, is simple: computer scientists are busily attempting to find a universal programming language in which all problems are to be solved (the old UNCOL ideal), and a universal data structure or data structure mechanism.

The analysis of information given previously shows this to be a misdirected effort, except possibly in one instance. While it is not possible to have a universal language, it is worthwhile to seek a generalized language useful as a system designer's language, or meta-language for other users. With this much narrower goal in mind, the current research becomes practicable. Here, however, the most difficult part becomes finding a language which is extremely efficient in implementation, since we already have many generalized-enough languages (e. g. set theory, graph theory, machine language, or PL/I).

In a similar response to the need for a multiplicity of data structures, some computer scientists have been attempting to

handle totally unstructured data (a contradiction in terms) or, failing that, to find a universal data structure (Earley 1971). This notion loses by the same criticism of a universal programming language: even though a terribly generalized and abstract structure might be able to handle almost all known applications and conceptualizations, it would simply not be very informative in most contexts. A parallel can be found in mathematics. All theories and entities in mathematics can be expressed in set theory and the predicate calculus. Yet analysis talks of real and complex numbers, and algebra of groups, rings, and fields. The level with which they deal effectively is not the lowest level of conceptualization possible.

A current approach to the need for idiosyncracy is that of providing a generalized mechanism which is capable of being specialized as necessary. An awareness of this situation in the domain of programming languages has led to the extensible languages:

> There are two basic premises which underlie the development of ELF. The first of these is that there exists a need for a wide variety of programming languages; indeed, our progress in the understanding and application of computers will demand an ever widening variety of languages. There are, in fact, "scientific" problems, "data processing" problems, "information retrieval" problems, "symbol manipulation" problems, "text handling" problems, and so on. From the point of view of a computer user who is working on one or more of these areas there are certain units of data with which he would like to transact and there are certain unit operations which he would like to perform on these data. The user will be able to make effective use of a computer only when the language facilities provided allow him to work toward a desired result in

> terms of data and operations which he chooses as being natural representation of his conception of the problem solution. That is, it is not enough to have a language facility which is formally sufficient to allow the user to solve his problem; indeed, most available programming languages are, to within certain size limitations, universal languages. Rather, the facility must be natural for him to use in the solution of his particular problem. . . .
>
> It is our contention that the most reasonable approach to providing the desired variety of language facilities is that of providing an extensible language supported by an appropriate compiling system. We do not, however, suggest that we can now devise a single universal core language which will adequately provide for the needs of the whole programming community; the diversity in "styles" of languages and translation mechanisms will probably always be sufficient to encourage several language facilities. ELF, which is the subject of this paper, provides a facility in the "style" of such languages as ALGOL-60, PL/I, and COBOL. (Cheatham et al 1968, p. 937)

More generally, there are developments such as the REL system, described in detail below. This is a generalized language system, designed to handle a large variety of specialized languages, which need not be related to each other and can indeed be extensible themselves.

These advances portend the proliferation of "natural" languages and made-to-order conceptualizations. This shift will force attention away from computer techniques toward information techniques. We are facing the beginning of a real information science and with it, an information engineering.

CHAPTER III

THE REL DATA ANALYSIS SYSTEM

Chapter I developed the theoretical position of data analysis as an informational activity. Chapter II presented a descriptive taxonomy of computer systems for the support of data analysis and an assessment of the present boundaries of their application. In this chapter, we turn to consideration of a particular data analysis system - the REL (Rapidly Extensible Language) System. The architecture of REL reflects both our theoretical understanding of system requirements and our practical understanding of present capabilities.

Development of the REL system is based upon two goals: (1) to bring into concrete realization the theoretical view of Chapter I; and (2) to reach operational status with such a system at the earliest possible time.

The need for operational status on real applications derives from the lack of experience with these advanced systems, and how they affect information processing and data analysis in particular. In a sense, the REL System is a vehicle for testing our conceptualization of data analysis. We have little empirical evidence of a form that could be called scientific (namely from controlled experiment or planned intervention) of the conceptual processes. The view presented herein, namely that the task of "knowing" is finding the most revealing conceptualization, is only one of several,

and is by no means the most widely held doctrine of information processing. A popular, and contrasting position is that the structure of reality is to be discerned in the data taken from that reality, rather than imposed by the researcher as a way of giving meaning to observational evidence.

REL involves mechanisms for accommodating conceptual change and extension, for experimenting with the imposition of structure on data. The observation of serious applications of the system to actual data analysis tasks is expected to reveal much concerning the dynamics of information processes, by charting the use and evaluating the effectiveness of these mechanisms. In this way we believe it will reflect on the efficacy of our theoretical position.

Since our interest in REL is based upon these considerations, experiencing the actual operation of the system on real data becomes an important goal. How has this constraint influenced the design specifications of REL? The data bases available today prejudice the choice of system type. In terms of size, most current data bases contain about 10,000 to 1 million items. As one example, 98% of the 65 data bases archived by the Inter-University Consortium for Political Research in 1970 were within that size range (ICPR 1970). Near the end of the last chapter, Figure 9 related system type to the data base size which could reasonably be handled. On this rough graph we find that (1) the data management systems can

comprehend far larger amounts of data; (2) the relational data systems fall exactly in this range; and (3) both the deductive and complex modeling systems are unable to cope with this many items.

As we look deeper, we see two interacting effects. First, consider the computing times associated with tasks typical of each of the data system methods. We can state broadly, though not precisely, that (1) the data management systems' processing is simple and thus extremely fast; (2) the relational data systems are slightly more complicated and slower; and (3) the processing of the deductive or modeling systems is rather complex and time-consuming. Although the data management systems alone can handle the extremely large data bases, these bases have become so huge as to be unuseable in any case. With smaller data bases the relational data systems cost very little more and are enormously more powerful. For the kinds of applications where data management systems are useful, other types of systems can do much better.

Suppose we now consider the computing time and cost for some fixed analysis task, in data bases typical of each method. Here we find that, on comparative tasks and system specific data bases, the deductive and complex modeling systems are so powerful that they can accomplish given tasks easily, and hence have a small cost. The relational data systems are more restricted in capability, and thus will cost more to do the same task on data bases specific to their application. The data management systems

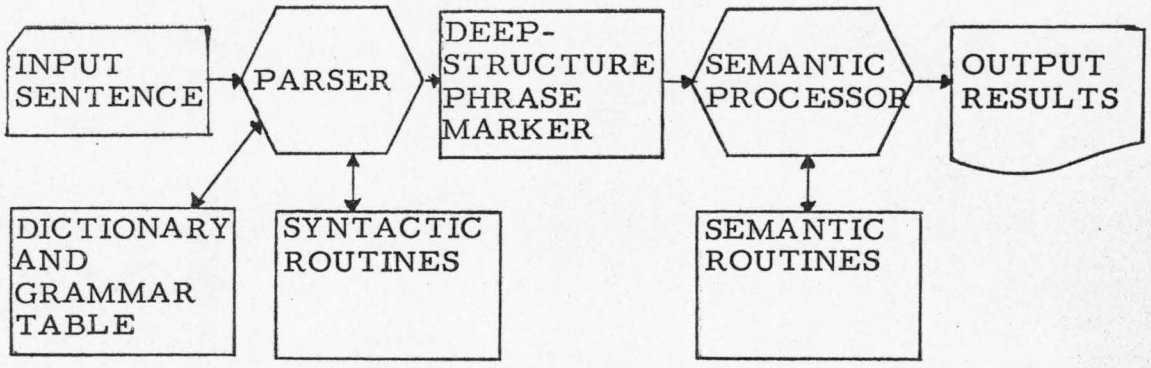
are extremely restricted - standard analysis tasks on their typical data bases are prohibitively expensive. One pays a low price for the restriction from deductive methods to relational methods in terms of analysis capability, and receives a very high payoff in terms of data base capability. For the areas in which deductive techniques are applicable, one can still perform a major portion of the task with somewhat less capability, (namely the local, rather than global, deductive ability). Thus the relational data systems are in exactly that compromise position today which promises a significant advance in operational capability.

The REL System

The REL System is predicated on the view that: (a) the central human information process is to seek the appropriate conceptualization; and (b) one's language is both the articulation of that conceptualization and the media for molding that conceptualization. It is a generalized computer system that supports a large variety of languages each specialized - by grammar, data structure, and processing algorithms - to some problem area. The system encourages the development of these "natural" languages and facilitates their implementation and extension. The REL System, then, is a maximally supporting environment in which "natural" computer languages are implemented. It puts only minimal constraints on possible languages, allowing the most general grammars, data structures and processing algorithms. Minimal system constraints mean that each language can seek its own efficient implementation, tailored and extended in response to the conceptualization of the particular user.

The System provides strong supportive resources. The REL System is a sentence driven, syntax directed interpreter. After a sentence has been input, it undergoes syntactic analysis by a parser. This produces a complete deep-structure phrase marker which in turn is used to direct the semantic processing of the sentence. Conceptually, therefore, the system can be described by the following diagram:

Fig. 10.--Syntax-directed interpretive systems



The interpretation of a sentence depends solely upon the grammar, the syntactic routines, and the semantic routines of a particular REL language. Thus an REL language is defined by exactly these three elements. The REL System consists of the total framework in the above diagram which integrates these elements and applies them to the syntactic and semantic analysis of the input sentences.

This overall REL System framework can be broken down into four major parts:

a) the language processor, including as major subparts the parser and the semantic processor;

b) the programming environment, consisting of two major components-the list processor and the paging mechanism;

c) the language extension component, namely the language building routines and language extension utilities;

d) the operating system components (over and above OS/360 itself) - the input/output components, job control language catalogued procedures, master routine, etc.

(a) The heart of the REL System is the language processor, consisting mainly of a parser and a semantic processor. The properties of these two programs, their efficiency, and how they are integrated determine to a large extent how the system works. The language processor is incomplete until provided with a grammar and corresponding interpretive routines; with these it becomes a total language system. The range of languages is determined mainly by the power and generality of the parser and semantic processor. REL uses a bottom-to-top general rewrite rule parser.

(b) Language processing and the stack organization of syntax directed interpretive routines, in present state of the art systems, make dynamic use of memory through list processing schemes. Such a scheme underlies the REL language processor and provides the media between the language processor and the syntactic and semantic routines. This general list processing mechanism is also made available to the syntactic and semantic routines themselves.

In a parallel fashion, the paging mechanism is a general resource used by both the REL System and the interpretive routines underlying any given REL language. These interpretive routines access pages as tabula-rosas. Thus they can organize and access data on these pages at the design discretion of the language programmer. Therefore REL accommodates any data structures (including, of course, programs themselves if so desired). Further, the interpretive routines have control of individual pages and the paging area, thus are in a position to optimize their own page

referencing algorithms and data organization. At the same time I/O, dynamic address relocation, etc., are handled automatically by the paging mechanism.

(c) Extensible languages are those which can change, by adding new syntax, during the course of conversation. REL has been designed particularly to facilitate the development and use of this type of language. There are utility programs that build the three language ingredients - grammar, syntactic routines and semantic routines - into an integrated language, producing the necessary grammar table and link-editing grammar table and interpretive routines into internal forms which can be efficiently applied by the language processing system. There is a second family of utility routines which manipulate the grammar table and organize the paging of definitional structures. These utilities are available to each REL language, providing the mechanisms of language extension.

(d) Finally, REL is implemented on top of OS/360 through a series of eight catalogued procedures, and the master routines that organize access to the relevant data sets, initialize list processing and paging, handle query and answer input and output, and schedule the successive steps of language processing.

The REL Data Analysis System

REL consists of (a) the REL operating environment, and (b) REL languages built within that environment. The last section discussed the environment; we now turn to the application of that system. This dissertation is not directly concerned with all of the various language developments that are presently underway, e. g. the REL Animated Film Language and the REL Applied Mathematics Language. It is concerned only with the REL Data Analysis System, based upon the REL English language. Moreover, our particular concern is even more narrowly defined. The syntax of REL English, i. e. its capability to be queried in what is ostensibly natural English, is not the subject matter of this thesis, both because it has not been a part of this thesis research and also because the central remaining operational problems of building an effective data analysis system do not lie in the areas of syntax or language processing.

The limitations on current question-answering systems lie mainly in the semantics, especially the problems of efficiency which occur for any reasonably large sized data base. Such data bases will not fit into the main memory of a computer, but instead must be stored on much slower, secondary memory devices (typically magnetic disk). The bottleneck today is the amount of access to this secondary memory, for its relative slowness dominates all other processing time. The implications of this

problem of access to secondary memory and the development of effective solutions to it constitute the core problems of this thesis research.

These problems are central to data management and data analysis systems. They are approached here in a specific context, namely the REL Data Analysis System, with all that implies for a rich but restrictive programming and operating environment. Nevertheless, our discussion of these problems is directed toward contributing to a general understanding of these problems and the tactics for their solution. The fact that we work within the REL environment serves largely to give concrete specificity to our results.

Consideration of the problems of secondary memory access naturally divides into two specific technical areas:

1) data structures and the algorithms for processing them, and

2) the organization across a sentence (or program) of the quantification of variables.

Each of these will be considered in detail.

Data Structures and Processing

We shall attempt to minimize the number of accesses to secondary memory in a paging environment. The environment will be unusual, however, in that semantic routines will be able to exercise complete control over the transmission of pages, rather

than being dependent on some generalized system page replacement algorithm. This will enable us to attempt to find true minimums.

In any relational data system, ubiquitous to the semantic processing of nearly every query and embedded deep within that processing is a central routine, namely: to find the image of a given class under a given relation.

Examples are:

A: parents of people

B: prices of stocks

C: salaries of employees

D: allies of countries

Besides being the basic operation in a relational system, one can see that the internal processing will be rather similar to that for almost all of the other large data operations, such as the intersection of two classes. Most of what can be said on the image problem is directly applicable to the other important processes in the system.

For this single task, then, we will see the effects of data structure and processing algorithms on the number of page transmissions, and therefore on overall efficiency. The coordination of data structure and algorithm is important, for there are many documentations of the catastrophic failure of either not coordinating the two or entirely ignoring the properties of a hierarchical memory (e. g. Brawn and Gustavson 1968; McKellar and Coffman 1969).

We therefore turn directly to the analysis of obtaining the image of a class under a relation. The class can be thought of as a set of pages, equivalent to a file, containing some identification of the members of that class. We define c to be the number of pages covered by the class, and c* to be the number of members in the class. Generally, c and c* are roughly proportional, depending on the number of elements which can fit on a page (usually 100 to 1000).

We now consider a number of alternative methods to store and process data, and the implications of these methods on finding the image of a relation. As each method is considered, it will be illustrated in terms of the following four examples:

Example A: "parents of people"

In this example we assume a data base which includes family relationship information. Such a data base could be from anthropological field data concerning an ethnic group or primitive tribe. We shall assume that there are 1000 people and that each has two parents.

Example B: "prices of stocks"

There are 2000 companies listed on the New York Stock Exchange whose prices vary over time. This data base will cover 50 time periods, containing the price of each stock at each time period (e. g. weekly price data for one year).

Example C: "salaries of employees"

A typical personnel file for a large industrial firm contains data such as the current salary level for each employee. This example assumes 10,000 employees and that each has a single salary figure.

Example D: "allies of countries"

The United Nations has about 150 member nations. Over the lifetime of that organization, both the membership composition and the web of alliances has been changing. We will postulate an average of 25 allies for each country.

Method I: Fixed Format. This method embodies a fixed-format data structure together with a direct accessing scheme. Each individual in the data base has associated with it a page or set of pages. All data related to that individual are kept there, and corresponding to every relation is a fixed location in that data file in which the value of that relation is stored. The identification associated with an individual is the page address of its data file.

With this data structure the algorithm for finding the image of a given class under a specific relation becomes: (1) get a member identifier from the class; (2) read the data file addressed by that identifier; (3) go to the fixed location in that file specified by the relation and find its value; and (4) save that value in an output class and repeat the algorithm (execute step 1).

The analysis of the paging behavior for all of the methods discussed in this section will be standardized in two ways. We

will assume that all data is on secondary memory at the start of the algorithm (an assumption we will reconsider later in the case of repeated applications of relations). We will also ignore the page transmissions required for the image class - the output of the algorithm. This is done because we do not know the size of the output class, and also because the number of page transmissions will be the same for all methods. Thus it does not affect their relative efficiency.

The fixed format method requires that we read at least one page for each member of the class, that is, in step 2 of the algorithm. We will assume exactly one page per member, since the fixed location for our given relation should enable us to directly address the right page. Add to this one-page-per-member the reading of the class itself and we find that the number of page transmissions required by method I: fixed format is (c* + c).

We will now consider the meaning of this figure in each of our examples. A constant factor in these calculations is the number of member identifications which can be placed on a single page. This number determines c as a function of c*: we shall use the REL Data Analysis system figure of 253[1].

---

[1] The REL Data Analysis system has a page size of 2048 bytes, a class element size of 8 bytes, and a 24 byte header at the top of each page.

Example A: "parents of people"

For 1000 people c* = 1000, and c = 4. The total number of page transmissions will be: 1004

Example B: "prices of stocks"

There are 2000 companies, and therefore c* = 2000, c = 8, and the total is: 2008

Example C: "salaries of employees"

The company has 10,000 employees: c* = 10,000 and c = 40 10,040

Example D: "allies of countries"

We have 150 countries, thus c* = 150 and c = 1, for a total of 151

The fixed format method does not distinguish among our examples, except on the basis of the size of the class. Secondary random-access storage media today consist of either fixed- or moving-head magnetic disks. The fixed-head disk can access any page in about 20 milliseconds, or 50 pages per second. Our examples thus have the following, more easily interpreted, elapsed times:

A. 20 seconds

B. 40 seconds

C. 3 minutes

D. 3 seconds

Example C clearly approaches the size limit for interactive response for the fixed-format, direct-access method.

Method II. Ring Structure. In this data structure each individual and relation consists of a linked list of elements, circularly closed. An element contains space for the link and an extra space for a cross-link. A primitive item of data such as "Robert is the father of Sue" is maintained by creating a cross-connecting ring. This ring links an element of the "Sue" ring to an element of the "father" ring, and then to an element of the "Robert" ring.

The representation of this structure on pages places each individual or relation ring on a page (or list of pages). The cross-rings are then represented by pointers connecting elements on each of the rings involved.

With this data structure we have two algorithms for finding the image - one for the relation and another for the converse of that relation. To simplify matters we will assume that every element contains the identification of the cross-linked ring along with the pointer into that ring. This means that we do not have to load the ring to see which ring it is.

The algorithm for finding the image of a primitive relation is: (1) get the identification of a class member; (2) load its associated ring and search it for an element containing the identification of the given relation; (3) when such an element is found, walk to the ring element of the relation by loading that page, and pick up the identification of the image; (4) place that identification in the output class and repeat from step 1.

For the converse of a primitive relation we use the following procedure: (1) get the identification of a class member; (2) search the relation ring for an element containing that identifier; (3) when such an element is found, walk the cross-ring to the range element and pick up the identifier of the domain; (4) output it and repeat.

An early such use of ring structures can be found in F. B. Thompson's classic DEACON work (Craig et al. 1966). DEACON used "referent rings" and "connective rings" and contended that "ring structures are adequate for storing a wide range of richly interrelated data that is pertinent to such functions as intelligence analysis, management planning and decision making." (p. 366). The data structure described above was actually implemented in an earlier version of REL English (Thompson et al. 1969).

The analysis of paging behavior for ring structured data is slightly more difficult. For primitive relations, we must load the ring corresponding to each individual in the class (step 2 of the algorithm) plus some number of pages for the relation. We now need three more parameters: r, the number of pages taken by the relation; r*, the number of elements in the relation; and K, the number of page frames in main memory available to our algorithm.

The number of relation pages which must be loaded can be estimated by consideration of the following two cases. First, if the relation is small enough to fit into main memory ($r << K$), one

need never load more than r pages. If the relation is large, however, one may be in a position where every access to the relation requires another page load. This would happen if the particular page holding the relation element was never in main memory when needed. Thus the number of page transmissions lies between (c* + c + r) and (c* + c + c*) - always greater than the (c* + c) for the fixed format method. These figures also assume that each individual ring is only one page long.

The analysis for the converse relation algorithm is similar: the number of page loads is dominated by c*. Here we must load the page of the range element for each class member identifier found in the relation.

Example A: "parents of people"

In a data base consisting of 1000 people we will have a parent relation with 2000 elements. With good packing a ring element will fit in 12 bytes, or 168 elements per REL page.

Thus, c* = 1000
c = 6
r* = 2000
r = 12

Since there are only 12 pages containing the relation and we can expect about 20 page frames, the total number of page loads will be: 1018

Example B: "prices of stocks"

Here there are 2000 companies and 50 prices for each, so that the relation becomes large: 100,000 elements.

c* = 2000
c = 12
r* = 100,000
r = 600

If we have but 20 page frames available, there is only a small chance that a relation page needed is already loaded. Thus, we will need essentially 2 (c*) pages: 4012

Example C: "salaries of employees"

The sheer size of c* dominates:

c* = 10,000
c = 60
r* = 10,000
r = 60

There is a one-third chance that a relation page will be in memory when needed, and so the expected number of page loads is $(c^* + c + \frac{c^*}{3})$: 13,360

Method III. Relational Data Structure. The preceding two methods were limited by the need to bring in a page for each member of the class. The relational data structure overcomes this difficulty by rearranging the data to be local, a property that data which must be accessed in a group is physically near also. In this data structure a relation consists of a list of pages whose elements are ordered pairs - the identifier of an argument and the identifier of a value. The relations contain all of the data in the data base; there is no longer any need for pages associated with individuals.

A simple algorithm for finding the image of a given class is the following: (1) get the identifier of a class member; (2) search the entire relation for ordered pairs with matching first element; (3) when one is found, output the second element of the pair; (4) repeat from step 1. The converse of a relation can be found by matching on the second half of an element.

This method is not useable because of its paging characteristics. If the relation is small enough to fit into main memory, we can load it and then read the pages of the class one at a time. With K available page frames, we must have $r \leq K-2$ so that the relation will fit alongside one input class page and one output class page. In this case, we will have read the relation once, and then the class once, for a total of $(r + c)$ page transmissions.

Suppose, however, that $r > K-2$, that is, the relation is too large to be contained in available memory. Now for every class member all r pages of the relation must be loaded, since the cyclic nature of the accessing of relation pages always finds that the next page needed is on secondary memory. Thus in this case the algorithm loads $(r*c*)$ pages.

Example A: "parents of people"

In this data base of 1000 people and 2000 parents we have:

$c* = 1000$
$c = 4$
$r* = 2000$
$r = 12$

For K available page frames, if K ≥ 14 we have: 16

if K is smaller, then we need: 12,000

Example B: "prices of stocks"

Here c* = 2000
c = 8
r* = 100,000
r = 596

We can assume that the relation does not fit into main memory. Thus the total number of page loads is: 1,200,000

Example C: "salaries of employees"

c* = 10,000
c = 40
r* = 10,000
r = 60

For K ≥ 62 we have: 100

For K <62 we need: 600,000

Clearly this algorithm collapses when the relation is large, though with enough main memory it is more efficient than the methods depending on c*. The next method is a modification of this one, which attempts to overcome this difficulty.

Method IV Generated Relational Data. The primary tenet of good programming practice in a paging environment is that one should utilize as much data as possible from a page once it has been loaded. This method attempts to achieve efficiency with the relational data structure by manipulating the sequencing of page loads and identifier comparisons.

Suppose that the algorithm knows the value of K, the number of available page frames. It can then consider the relation to be composed of a sequence of sub-relations, each small enough to be held in main memory. Now the algorithm can form the image of the given class under each subrelation in sequence, using the simple Method III, and concatenate the results. The fact that the sub-relation can be loaded in its entirety means efficient processing for each segment.

This algorithm, which we will call GEN-R, is: (1) load the next K-2 pages of the relation; (2) read through the entire class, one page at a time, and form the image of the class under that subrelation; (3) repeat the process until the relation is exhausted.

There is a dual to this algorithm, called GEN-C, which breaks the class into small sub-classes: (1) load K-2 pages of the class; (2) read through the relation, one page at a time: (3) for each relation page in memory, form the image of that sub-relation and subclass; (4) after the entire relation has been read, get the next subclass and continue.

For these algorithms the analysis of paging is quite simple. The GEN-R algorithm structures the relation as $\left\lceil \frac{r}{K-2} \right\rceil$ subrelations, each, except possibly the last, (K-2) pages long. The algorithm reads through the class once for each subrelation, for a total of $C \cdot \left\lceil \frac{r}{K-2} \right\rceil$ page loads. The relation itself is read only once. Thus

the GEN-R algorithm requires $r + c \cdot \left\lceil \frac{r}{K-2} \right\rceil$ page transmissions, and GEN-C, since it is entirely dual, requires $c + r \cdot \left\lceil \frac{c}{K-2} \right\rceil$.

The relational data structure, with no further organization, requires a minimum of (r + c) page loads. This number means that each class page and each relation page is loaded once and only once. When $r \leq K-2$ the GEN-R algorithm achieves this minimum; when $c \leq K-2$ the GEN-C algorithm does. These algorithms in general are sensitive to the relative sizes of K and r or c. The examples below are therefore presented with varying values of K, representing between 10 and 50 available page frames.

Example A: "parents of people"

Since there are relatively few people, the number of pages involved here is small. The algorithms will be at the minimum values quickly.

c = 4
c* = 1000
r = 12
r* = 2000

K(number of page frames)	GEN-R	GEN-C (number of page loads)
10	20	16
20	16	16
30	16	16
40	16	16
50	16	16

Example B: "prices of stocks"

In this case the relation is large, yet the class is small. Under these circumstances the GEN-C algorithm minimizes the

number of page loads immediately; the GEN-R algorithm needs more space but is not too inefficient.

c = 8
c* = 2000
r = 596
r* = 100,000

K	GEN-R	GEN-C
10	1196	604
20	868	604
30	772	604
40	724	604
50	700	604

Example C: "salaries of employees"

Neither the relation nor the class will fit in main memory until K is fairly large. Yet the numbers of page loads are only a few times the minimum.

c = 40
c* = 10,000
r = 60
r* = 10,000

K	GEN-R	GEN-C
10	380	340
20	220	220
30	180	160
40	140	160
50	140	100

Example D: "allies of countries:

The class is so small that this has become an extremely easy case.

c = 1
c* = 150
r = 23
r* = 3750

K	GEN-R	GEN-C
10	26	24
20	25	24
30	24	24
40	24	24
50	24	24

Method V. Sort/Merge. The technique of sorting data has been used extensively, and sometimes unthinkingly, by the data processing community. We shall consider the implications of sorting the relational data structure. The power of the sorting technique stems from the situation in which both the class and the relation are properly ordered. In this case one can read through both class and relation simultaneously, keeping synchronized by use of the sort order: a merge process. This requires that each page in both the class and relation be loaded once and only once for a total of (r + c) page loads.[2]

Thus, on the assumption that the relation and class are already sorted, the number of page loads is at the minimum for the relational data structure. However, since we cannot

---

[2]The mathematical purists might argue that not all r pages of the relation need be loaded, since once the class is exhausted the merge process can stop, and vice versa. However, suppose one assumes that the individuals in the data base are numbered from 1 to N, and the class and relation contain random samples of individuals. Then the expected value of the maximum individual, i. e. the last, in the relation and class is $\frac{r^*}{r^*+1}$ N and $\frac{c^*}{c^*+1}$ N, respectively (Feller 1950, p. 212).
This means that for sizeable r* and c* we can expect to load every single page in both relation and class- hence this factor is ignored in the page transmission calculations.

guarantee that pre-ordering, this absolute minimum does not tell the whole story. A sort, if needed, can easily do more paging than some more sophisticated algorithm.

Sorting can be necessary under several conditions. First consider the relation. A binary relation can be ordered on either its domain or its range. One order is needed for the relation and the other for its converse. The relation could be duplicated and ordered both ways. This has been done, in fact, for small data base systems (Levien 1969), but this solution wastes expensive secondary memory. Further, the use of n-ary relations ($n > 2$) means that the relation must be replicated many times. One can instead keep the relation sorted one way and re-sort whenever necessary. A small, and certainly insufficient, study of queries put to a relational system revealed that this means sorting approximately one-half of the time for binary relations.

It may be necessary to sort the class also. The classes created during the process of sentence analysis may not be sorted, even when the classes in the permanent data base are sorted. In our image task, if the input class is assumed sorted then the output class must be sorted, for it may become the input of another application of the process. A further complication arises in that a class may have a subclass structure rather than simply members. An example is the class of "people" consisting of the two subclasses "male" and "female," each of which contains individuals.

Some amount of paging must be done to ensure a simple ordering on the classes involved.

The sort/merge algorithm will assume that in the data base all relations are ordered on their domain, and that no classes are sorted. This last assumption will make our estimates of paging activity overestimates, but not too much on the average. Thus, the algorithm is simply stated: (1) sort the class; (2) if we need the converse relation, sort the relation on its range; (3) merge the class and relation, producing the image.

The paging behavior of this algorithm can be estimated analytically for large data bases. Suppose we have a file which covers n pages and n is large enough so that the file cannot be contained in main memory. A simple, standard sort/merge technique to order that file works as follows: (a) subset the file into fragments of K pages each (except possibly the last), and sort each fragment while in main memory; (b) perform the required number of (K-1) - way merges, until all fragments have been merged into one, ordered, file. The sort phase will require 2n page transmissions, as each page is read and written once. A simple merge algorithm will require $\lceil \log_{K-1} n \rceil - 1$ merge steps with 2n page transmissions in each. Thus to sort an n-page file requires $2n \lceil \log_{K-1} n \rceil$ pages. Assuming that the relation requires sorting one-half of the time, the total number of page transmissions is $r(1 + \lceil \log_{K-1} r \rceil) + c(1 + 2 \lceil \log_{K-1} c \rceil)$.

Improvements can be made in this simple sort/merge which will improve on this formula slightly and these techniques have been incorporated into the REL sort/merge algorithm. In obtaining the numbers given in the examples below, we have used a simulation of the actual technique employed by REL.

Example A: "parents of people"

Both the relation and class are small enough so that significant savings can be made by working entirely in main memory. In fact, the absolute minimum is achieved for 25 available page frames.

c = 4
c* = 1000
r = 12
r* = 2000

K	SORT
10	44
20	32
30	16
40	16
50	16

Example B: "prices of stocks"

In this example the relation is so large that the paging required for its sort dominates. This is exactly the kind of situation in which the sort is relatively inefficient.

c = 8
c* = 2000
r = 596
r* = 100,000

K	SORT
10	2674
20	2180
30	1804
40	1804
50	1804

Example C: "salaries of employees"

Another example of files large enough to force a multiple pass sort, causing three times the minimum number of page transmissions.

c = 40
c* = 10,000
r = 60
r* = 10,000

K	SORT
10	380
20	380
30	380
40	380
50	380

Example D: "allies of countries"

Even though a rather small amount of data, the relation is large enough to cause excess paging until K is 50 or larger.

c = 1
c* = 150
r = 23
r* = 3750

K	SORT
10	71
20	71
30	48
40	48
50	24

Method VI: Others. There have been other suggestions for the implementation of relational structures which should be mentioned, and then rejected. One of the favorite techniques for searching a table in main memory is the binary search. If our relational data structure is ordered, we can use a binary search to find the value corresponding to any given argument. For any single argument we would expect to make $\log_2 r^*$ comparisons, or at the very least one page load. For a class of arguments we must repeat this process, and can save nothing from the full paging requirements. Thus a binary search will need c^* page loads at least - always worse than the direct access method I.

Another possibility which has been suggested and implemented (Feldman and Rovner 1969) is the use of hash coding the relational data. This clever implementation places the data for a given relation on a single, variable length "page" and hash codes the argument to find its location on that page. If the relation "page" fits in main memory this technique is fast; on the other hand, a relation which is larger means essentially c^* page accesses again. (Assuming that the relation is p times larger than available memory and that the hash function distributes uniformly, the probability that the current needed "page" is already in main memory is $\frac{1}{p}$. Therefore the expected number of page loads is $(1-1/p)c^*$.)

Summary of paging behavior. The methods considered above can be segregated into two categories: those which require a page load for each individual, and those which can group individuals. Fixed formats, ring structures, and hash coding are all in the first category. The number of page loads needed by these methods is proportional to the number of individuals in the class. Consequently, if the number of individuals is small these are extremely efficient; a large size class makes all of them break down catastrophically.

These methods have other virtues, especially the possibility of finding the values of several relations for a given individual at the cost of that same page load. This is the reason why they are used in the data management systems which produce telephone-book-like reports. The Fundamental Theorem discussed in Chapter I implies, however, that we are more informed if we step back from the absolute lowest level of detail. We need to be able to produce generalizations of our data.

Abstractions can be generalizations across a set of relations or across a set of individuals for a given relation. The latter problem is attacked by the second category of methods. They structure the data in such a way as to facilitate abstraction over sets of individuals, in particular collecting all the data concerned with a relation into physical proximity for efficient access.

Of the methods studied, the two generator algorithms and the sort/merge, each has its own range where it is the most efficient.

For very large data bases the sort/merge is superior: its paging is approximately $r \cdot \log(r)$ while the generators page about r^2 (or c^2). On smaller data bases, or smaller questions on large data bases, the generator algorithms are more efficient.

A rather nice solution has been implemented in the REL Data Analysis System. It is a simple matter to keep the values of r and c in each relation and class respectively. Then every invocation of the image-producing routine can be locally optimized by computing the number of page loads required for each algorithm and selecting the best algorithm for the particular input parameters. This dynamic minimization of paging has dramatic effects on the overall processing of a query.

Our four examples show why one should not naively use the sort/merge algorithm everywhere:

Example A: "parents of people"

c = 4
c* = 1000
r = 12
r* = 2000

K	SORT	GENR	GENC
10	44	20	16
20	32	16	16
30	16	16	16
40	16	16	16
50	16	16	16

Example B: "prices of stocks"

c = 8
c* = 2000
r = 596
r* = 100,000

K	SORT	GENR	GENC
10	2674	1196	604
20	2180	868	604
30	1804	772	604
40	1804	724	604
50	1804	700	604

Example C: "salaries of employees"

c = 40
c* = 10,000
r = 60
r* = 10,000

K	SORT	GENR	GENC
10	380	380	340
20	380	220	220
30	380	180	160
40	300	140	160
50	300	140	100

Example D: "allies of countries"

c = 1
c* = 150
r = 23
r* = 3750

K	SORT	GENR	GENC
10	71	26	24
20	71	25	24
30	48	24	24
40	48	24	24
50	24	24	24

More on paging. A further consideration is whether one can better optimize by taking a wider context. The succeeding section discusses the relationship between quantification and paging. Here we examine the implications of the common situation of composition of relations. Our paradigm example will be the phrase "locations of parents of people."

The most straightforward method for handling this phrase consists of applying some technique to "parents of people" to obtain the class of parents, then repeat the process independently for "locations" and that class. Thus the composition of relations is reflected in the composition of processes for finding the image of a single relation and class. This method has the advantages of simplicity and the use of an already needed procedure. The possibility remains, however, that a specialized routine might be more efficient. Fortunately, no - the straightforward method is also the most efficient in this case.

The simple composition method has the disadvantage that a temporary class must be created, and paged, which holds the output of the first application of the image procedure. In a procedure designed expressly for the composition case one can hope to eliminate that temporary class and thereby become more efficient. We can assume the relational data structure in which the relation consists of pairs < domain element, range element> . If both relations fit entirely in main memory one can proceed directly from argument to "relation of relation of argument" without an intermediate class. This can be done in our "locations of parents of people" example by (a) take a person, say Sue; (b) find her first parent, say Robert; (c) output all locations of Robert; (d) continue searching for other parents of Sue and repeat from (c) when one is found; (e) when there are no more parents of Sue, repeat the process from (a).

Complications arise when the relations are too large to fit into the available main memory. This could be handled by viewing the relations as sets of subrelations and the class as a set of subclasses, such that two subrelations and one subclass will all fit into main memory. One would then need to work through all combinations of the subrelations and subclasses, taking one piece from each of the three main sets of data, in order to find the composition image. Thus if "location" were broken into 2 parts, "parent" into 3, and "people" into 4, we could have $2 \cdot 3 \cdot 4 = 24$ combinations to consider. This means that the number of page transmissions becomes multiplicative (in the number of relations), as opposed to additive for the straightforward composition method. We thus have reason to stay with the simple technique.

Quantification

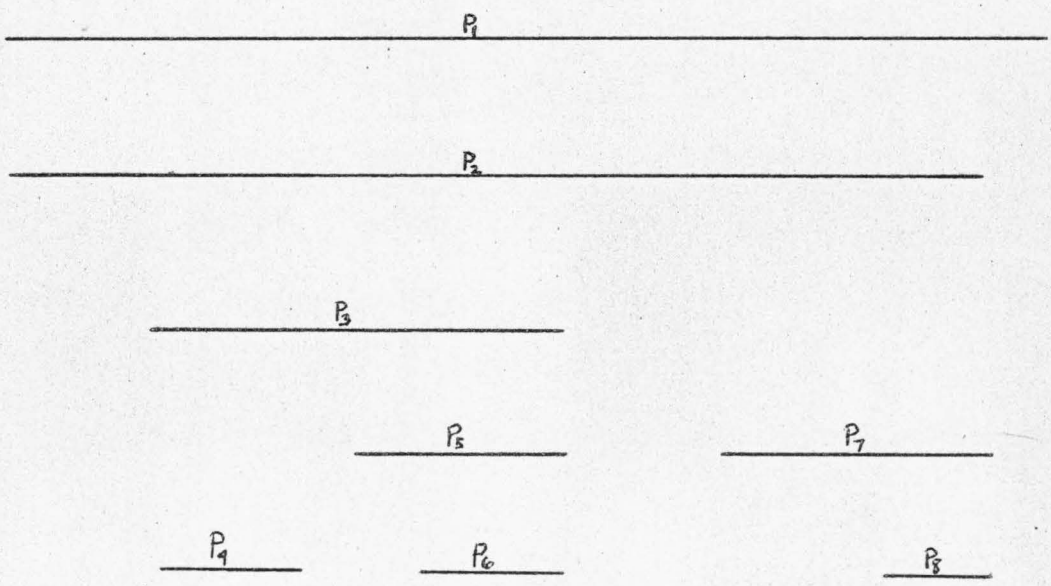
Despite the fact that quantification is basic to our intellectual endeavors, it has been relatively ignored by the designers of computer information processing systems. Quantification is one of our primary tools for abstraction and generalization, and the Fundamental Theorem implies that we gain information by moving from the level of detail of our data to the more abstract.

Quantification in English is exemplified by such phrases as:

all boys
at most seven books
which countries
each student

Note the use of such phrases in abstracting overall characteristics of classes of objects from details concerning each member of these classes. Thus, in the sentence "All Harvard students have at least one girlfriend at Radcliffe," a property of Harvard students as a class is derived from data relating individual Harvard students to individual girls, some of whom attend Radcliff.

The techniques of quantification will be illustrated by a single example: "Have the locations of all senators included at least 3 nations?" This in-depth examination provides the concreteness necessary for an understanding of a complex process. The parse of this example is below, with unimportant details omitted:



Have the locations of all senators included at least 3 nations?

Fig. 11.--Parse of quantification example

The example will be discussed in terms of its phrase marker, which is a set of phrases portraying the structure of the sentence and thus revealing the processing necessary to unravel the meaning of that sentence. We will use a LISP notation to express these phrase markers. Each phrase consists of two lists, a phrase list and a phrase information list. The phrase list - indicated (POS, F, PI) - contains a part of speech, syntactic features, and the name of the phrase information list. The features will be omitted when they do not affect the semantic processing. The indirection to the phrase information is made to facilitate the execution of the phrase marker, for the result of a semantic transformation is a new phrase information list which is then named by the old phrase element.

Phrase information lists can be of several types, identified by the first element:

1. (ROU, C, T) postfix routine: C is a list of the constituent phrases, and T is the name of a semantic transformation.
2. (GEN, C, T) prefix routine: (used mainly in generating situations).
3. (DATA, D) data: D is some data such as a number or a page in the data base, indicated by α_{location}.

4. (VAR, R, TY) primitive variable: R is a phrase which is the range of that variable, TY is the type of quantification.

5. (OUT, STR) output string

The "variable" technique. The "variable" technique for handling English quantifiers turns each quantified noun phrase into a "variable," in the REL sense. This variable then propagates upward through the parse during the syntactic processing of the sentence, and gets bound at the appropriate level of analysis.

The quantified noun phrase qua variable contrasts rather sharply with arithmetic expression or predicate calculus variables. These latter variables are truly place markers, conveying only syntactic information. The type of quantification, such as the arithmetic sum or product, and the range of values for the variable are provided when that variable becomes bound. Quantified noun phrases, on the other hand, acquire such data at the time they are created. "All senators" is a variable with an "all" type of quantification and the class of senators for a range.

The arithmetic or predicate calculus variable has an explicit syntactic marker which indicates the point at which it becomes bound. Phrases such as "sum f(x) for x=1 to 10" clearly bind variables, besides specifying the quantification. In English, however, variables are bound at the clause or sentence boundary, and there is no explicit binding phrase. In our present example

the two quantifiers, "all senators" and "at least 3 nations," are bound at the sentence boundary P_1. The p-marker below shows that two generator phrases have been inserted, corresponding to the quantifiers. These phrases are the representation of a bound, quantified, variable.

Fig. 12. --P-marker for "variable" quantifier technique

P_1: (SS, PI_1) PI_1: (ROU, (P_a), T_{ss})

P_a: (VP, PI_a) PI_a: (GEN, (P_b), T_{all}, (P_6, ptr), R_a)

P_b: (VP, PI_b) PI_b: (GEN, (P_2), $T_{at\ least\ 3}$, (P_8, ptr), R_b)

P_2: (VP, PI_2) PI_2: (ROU, (P_3, P_7), T_{is})

P_3: (IN, PI_3) PI_3: (ROU, (P_4, P_5), T_{image})

P_4: (NP, PI_4) PI_4: (DATA, $\alpha_{location}$)

P_5: (NP, PI_5) PI_5: (VAR, (P_6), all)

P_6: (NP, PI_6) PI_6: (DATA, $\alpha_{senator}$)

P_7: (OJ, PI_7) PI_7: (VAR, (P_8), at least 3)

P_8: (NP, PI_8) PI_8: (DATA, α_{nation})

R_a: ((P_b, PI_b), (P_2, PI_2), (P_3, PI_3), (P_5, NP/0))

R_b: ((P_2, PI_2), (P_7, OJ/0))

The p-marker in figure 12 indicates a kernel in which a copula has an instrumental and an objective case. The instrumental case is the location of some particular senator; the objective is some nation. Built around this kernel is the generation and resolution of the "all senators" and "at least 3 nations" phrases.

While it is always difficult to describe recursive processes, the following is a narration of the execution of this phrase marker. The indentation follows conventional block structure format.

Process P_1:

(1) Process P_a:

(A) generate first (next) senator, say senator i, and refresh, thus making P_5: (NP, PI_5')

PI_5': (DATA, $\alpha_{\text{senator i}}$)

(B) process P_b

(1) generate first (next) nation, say nation j, and refresh, thus P_7: (OJ, PI_7')

PI_7': (DATA, $\alpha_{\text{nation j}}$)

(2) process P_2

(a) process P_3

(i) process P_4: recognize it as DATA and return

(ii) process P_5: DATA

(iii) apply T_{image} to (P_4, P_5)

output: P_6: (NP, PI_6')

PI_6': (DATA, $\beta_{\text{location of senator i}}$)

(b) process P_7: recognize DATA and return.

(c) apply T_{is} to (P_3, P_7)

output: P_2: (VP, PI_2')

PI_2': (DATA, yes, if the location of senator i is nation j; no, otherwise)

(3) apply $T_{\text{at least 3}}$ to P_2 of (i, j)

count affirmatives.

if count < 3, continue generation on j (i.e. repeat from step 1)

if count = 3, output PI_b': (DATA, yes)

if generation complete, output PI_b': (DATA, no)

for any output set P_b: (VP , PI_b')

(C) apply T_{all} to P_b of (i)

if affirmative, continue generation on i (repeat from step A)

if no, output PI_a': (DATA, no)

if generation complete, output PI_a': (DATA, yes)

for any output, set P_a: (VP , PI_a')

(II) apply T_{ss} to (P_a)

output P_1: (SS, PI_1')

PI_1': (OUT, "yes" or "no")

The essence of the "variable" technique is the generation of all quantified classes down to individuals, and the application of the core analysis process to those individuals in the innermost loop. The core processes operate on individuals only and are not aware of the quantification around them. This is conceptually clean, but operationally disastrous.

One of these core processes in the above example is the image routine, which produces the "location of senator$_i$." Since the "variable" technique of quantification invokes the image routine

for every individual in the range, and each invocation requires at least one page load regardless of the data structure, this technique will page proportionately to the number of elements in the range of the quantifier. The analysis of the previous section has shown, however, that such paging is unacceptable, and avoidable.

There is another, deeper, objection to the "variable" technique for handling quantifiers which dooms those systems using the predicate calculus as an intermediate language between English and the data. The "variable" technique, and the language of the predicate calculus, requires that all quantifiers be properly nested. In our example the computation of the "location of senator$_i$" is within the quantification over nations, and normally would be repeated as many times as there are nations. Fortunately the REL refresher mechanism provides a "do-loop" optimization which guarantees that no redundant processing will occur. In this case the refresher stack associated with the nation quantification does not contain P_3 ("location of senator$_i$") so that P_3 is processed only once for each senator.

The multiplicative effect can be seen in another example: "Which boys are friends of at most 3 girls?" The phrase marker associatedwith this query is shown in figure 13. Here "boys" are quantified as the outer variable, "girls" are the inner variable, and the central process is the test, "is boy$_i$ equal to a friend of girl$_j$?"

Fig. 13.--P-marker for second "variable" quantifier example.

P_1: (SS, PI_1) PI_1: (ROU, (P_a), T_{ss})

P_a: (VP, PI_a) PI_a: (GEN, (P_b), T_{which}, (P_4, ptr), R_a)

P_b: (VP, PI_b) PI_b: (GEN, (P_2), $T_{at\ most\ 3}$, (P_8, ptr), R_b)

P_2: (VP, PI_2) PI_2: (ROU, (P_3, P_5), T_{is})

P_3: (AG, PI_3) PI_3: (VAR, (P_4), which)

P_4: (NP, PI_4) PI_4: (DATA, α_{boy})

P_5: (OJ, PI_5) PI_5: (ROU, (P_6, P_7), T_{image})

P_6: (NP, PI_6) PI_6: (DATA, α_{friend})

P_7: (NP, PI_7) (VAR, (P_8), at most 3)

P_8: (NP, PI_8) PI_8: (DATA, α_{girl})

R_a: ((P_6, PI_6), (P_2, PI_2), (P_3, AG/0)

R_b: ((P_2, PI_2), (P_5, PI_5), (P_7, NP/0))

The fact that the innermost quantified variable, $girl_j$, is involved in a computation which is independent of the outermost quantifier means that this computation will be repeated many times unnecessarily. In this case there is no solution: "do-loop" optimization is irrelevant and does not help, and the quantifiers cannot be interchanged. The unaware system which uses the "variable" quantification technique can be devastated by this multiplicatively excessive, useless computation.

The "label class" technique. The REL data analysis system uses a method for handling quantifiers which circumvents the problems discussed above. This method turns a phrase such as "all senators" into a class which is marked with the type of quantification, and in which each element is associated with the identification of a quantifier range element. The label, as the identification is called, represents the instance of the quantified variable which led to the present element. Thus the phrase "locations of all senators" is represented by a class consisting of the pairs <New York, Jones >, < Boston, Smith >, and so on, meaning that a location of Senator Jones was New York, etc. Notationally this class will be written $\alpha^{<0,\ all>}_{<location,\ senator>}$ The subscripts are the class elements; the superscripts identify the type of quantifier (with 0 indicating none). The "label class" technique shifts the burden from the syntactic analysis of variables to the semantic analysis of labels. Re-considering our example "Have the locations of all senators included at least 3 nations?", we now have the simplified phrase marker below.

Fig. 14--P-marker for "label class" quantifier technique

P_1: (SS, PI_1) PI_1: (ROU, (P_2), T_{ss})

P_2: (VP, PI_2) PI_2: (ROU, (P_3, P_7), T_{is})

P_3: (IN, PI_3) PI_3: (ROU, (P_4, P_5), T_{image})

P_4: (NP, PI_4) PI_4: (DATA, $\alpha_{location}$)

P_5: (NP, PI_5) PI_5: (ROU, (P_6), T_{all})

P_6: (NP, PI_6) PI_6: (DATA, $\alpha_{senator}$)

P_7: (OJ, PI_7) PI_7: (ROU, (P_8), $T_{at\ least\ 3}$)

P_8: (NP, PI_8) PI_8: (DATA, α_{nation})

The importance of the "label class" technique for handling English quantifiers lies in the properties of its semantic processing. We first describe the processing of this example and then discuss it. Process P_1:

(I) Process P_2:

(A) Process P_3:

(1) Process P_4: recognize it as DATA and return

(2) Process P_5:

(a) Process P_6: recognize as DATA, and return.

(b) apply T_{all} to (P_6)

output P_5: (NP, PI_6')

PI_6': (DATA, $\alpha^{all}_{senator}$)

(3) apply T_{image} to (P_4, P_5)

output P_3: (NP, PI_3')

PI_3': (DATA, $\alpha^{<0,\ all>}_{<location,\ senator>}$)

(B) Process P_7:

(1) Process P_8: recognize DATA and return

(2) apply $T_{at\ least\ 3}$ to (P_8)

output P_7: (NP, PI_7')

PI_7': (DATA, $\alpha^{at\ least\ 3}_{nation}$)

(C) apply T_{is} to (P_3, P_7)

output P_2: (VP, PI_2') PI_2: (DATA, yes/no)

(II) apply T_{ss} to (P_2)

output P_1: (SS, PI_1') PI_1': (DATA, "yes"/"no")

The essence of the "label class" technique is that processes operate on quantified classes as a group, rather than individually. Thus in step I. A. 3 we apply the image routine to "location" and "all senators" and can utilize the paging optimization discussed in the previous section. This reduction of paging during quantification represents an extremely important breakthrough, for it shifts the economic balance toward the use of abstraction. Since abstraction has been so neglected in recent computer systems, any such shift has a large payoff in informativeness.

The other problem attached to the "variable" technique, that of redundant computation, is also solved by the "label class" method. Every phrase is computed once only and the quantifiers essentially work their way upward through the phrase marker. Quantifiers interact when two labelled classes are merged, as in step I. C for $\alpha \begin{matrix} < 0, \text{ all} > \\ < \text{loc.}, \text{ senator} > \end{matrix}$ and $\alpha \begin{matrix} < \text{at least } 3 > \\ < \text{nation} > \end{matrix}$. In these situations the quantifiers are ordered, consolidated, and sometimes resolved. To explain this process we will use several new examples.

Surface structure ordering of quantifiers. Our first examples concern quantifiers which are similar, possibly identical, and the determination of their order of nesting. We shall consider the following two examples: (a) All people play some sport; and (b) Some sport is played by all people.

The latter sentence is clearly the passive form of the former, and yet differs in an important manner from the normal passive transformation. Consider "John plays baseball" and "Baseball is played by John." These sentences, while different in surface structure, are identical in deep structure and in meaning. Linguists have been careful to note this retention of meaning through the passive transformation. The meanings of our two examples differ, though it is the same passive transformation, in a way reflecting a different ordering of the quantifiers. "All people play some sport" means that each person plays something, and that sport may be different for different people. For this sentence to be true it is enough that each individual play any sport.

On the other hand, "some sport is played by all people" means that there is a single sport, which everyone plays. This requirement that everyone play the same sport is not implicit in the active form of the sentence. The difference in meaning is exactly in the nesting of the quantifiers: the active form places the "all" quantifier outermost followed by the "some" quantifier, the passive has the "some" followed by the "all." Since the deep

structures are identical, the difference in meaning must be a function of the differing surface structures. If we include the feature marking the surface structure subject, our examples have the following phrase markers:

a) P_0: (VP, - PI_0) PI_0: (ROU, (P_1, P_2), T play)
P_1: (AG, sur. subj., PI_1) PI_1: (DATA, $\alpha \begin{matrix} \text{all} \\ \text{people} \end{matrix}$)
P_2: (OJ, - PI_2) PI_2: (DATA, $\alpha \begin{matrix} \text{some} \\ \text{sport} \end{matrix}$)

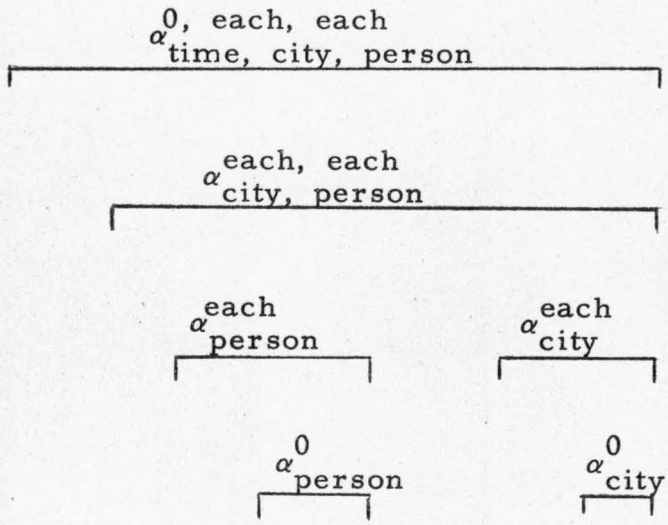
b) P_0: (VP, - ,PI_0) PI_0: (ROU, (P_1, P_2), T play)
P_1: (AG, - ,PI_1) PI_1: (DATA, $\alpha \begin{matrix} \text{all} \\ \text{people} \end{matrix}$)
P_2: (OJ, sur. subj., PI_2) PI_2: (DATA, $\alpha \begin{matrix} \text{some} \\ \text{sport} \end{matrix}$)

Using the simple rule that surface object quantifiers should be nested within surface subject quantifiers, our examples conceptually consolidate the quantifiers into these classes:

a) $\alpha \begin{matrix} < \text{some, all} > \\ < \text{sport, people} > \end{matrix}$

b) $\alpha \begin{matrix} < \text{all, some} > \\ < \text{people, sport} > \end{matrix}$

The quantifiers can then be resolved, innermost first, and in both cases produce the correct interpretation. Another example of this same effect of surface structure is in "when did each person live in each city?" Here one wants as output a list of people and for each, a list of cities and times. Although ignored so far, all data has a time span associated with it in the REL data analysis system. This adds tremendous complexity to the processing routines, yet is absolutely essential to a useful system. In this example, we indicate only a simplified version of the processing.



When did each person live in each city?

Fig. 15.--Parse and label class processing for (each, each) example.

The precedence ordering of quantifiers. The rule that surface object quantifiers are nested within surface subject quantifiers works if the quantifiers are similar. There is a hierarchical ordering, however, which supersedes this rule. We can classify as similar all quantifiers such as some, at least n, at most n, exactly n, all, all but n, etc. These quantifiers are the ones which should be nested within any of the other types. The next group are the ones which count: how many, what proportion of, and what percentage of. These quantifiers should be kept outside the first group, and nested within the last group of quantifiers.

These produce labels to be output as tabular listings: each, which, and what.

One can see the effect of this ordering in the example, "At most 3 people have lived in which cities?" The "which" quantifier, even though it is the surface object, must be treated as the outermost to produce the class $\alpha \begin{matrix} < \text{at most 3, which} > \\ < \text{people, cities} > \end{matrix}$. The answer to this question is a list of cities, since the "at most 3" quantifier is resolved at the clause boundary.

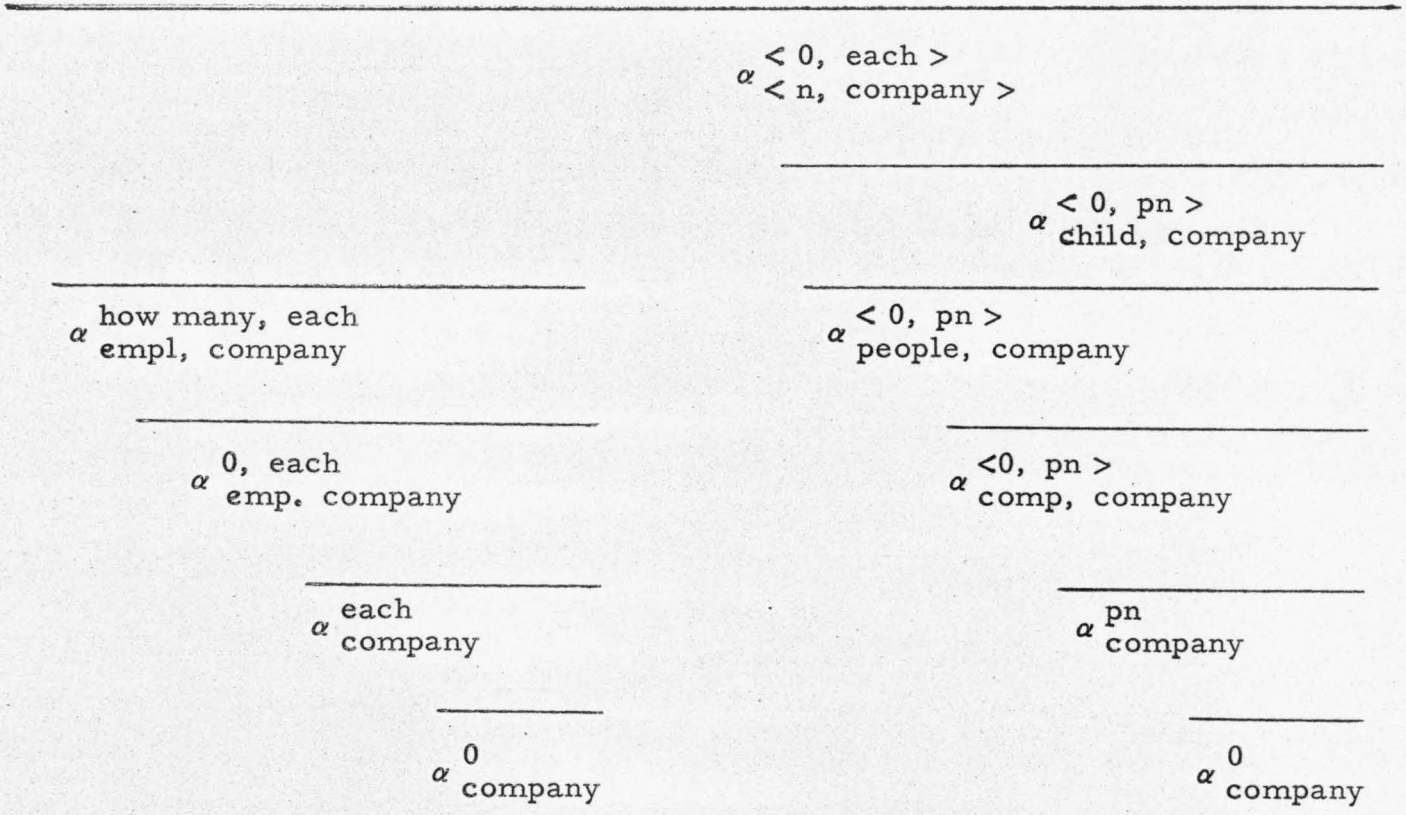
Thus we have a precedence ordering of the quantifiers which partially determines the order of nesting in a multiply-quantified class. The nesting order in turn determines the interpretation of a phrase and finally of the entire sentence. The complete rule for nesting can now be stated: when two quantified phrases are to be merged, the quantifiers are to be nested first by the precedence order and within each precedence group by the left-to-right order of appearance within the sentence, that is, quantifiers on the right are to be nested within quantifiers on the left.

Resolution of quantifiers. Mentioned above was the resolution of a quantifier, that is, the point at which the quantifier disappears and is replaced by a simple, non-quantified set. Quantifiers are resolved by processes which depend on the quantifier type and at points in the phrase marker which depend on the precedence order.

The all, some, at least n quantifiers resolve into booleans by processes corresponding to either universal or existential logical quantification. The how many quantifiers resolve into numbers by a counting operation, and the each or which quantifiers resolve into character strings placed on the output.

The lowest precedence level quantifiers, all, some, etc., are resolved at the clause boundary. This occurs when a verb phrase gets parsed into a non-verb phrase, such as sentence, noun, or time. All other quantifiers are resolved only at the sentence level. This difference is important because of the possibility of subordinate clauses. The all or some quantifiers are eliminated at the subordinate clause boundary: "people who live in some city" represents a non-quantified class of people.

The last sentence of this section illustrates many of the properties of quantifiers and their interaction. Figure 16 is a representation of its parse and label class processing and hints at an exciting development for the future: the label type "pn" used for a generalized anaphoric expression. "How many employees of each company are children of people who have worked for some competitor of that company?"



How many employees of each company are children of people who have worked for competitors of that company?

Fig. 16.--General label class quantification

Use of the REL Data Analysis System

The difficulty of articulating the impact of a responsive, flexible data analysis system must be apparent, and the non-computer scientist reading this will probably have found the inside, technical viewpoint almost incomprehensible. This section will present the system from the other side of the language: the user's view.

As a typical, small-to-medium size data base we will use the demographic data compiled by Professor Bruce Russett of Yale University (1969). It consists of 75 political, social, and economic indicators on each of 133 countries. The total number of datums is therefore approximately 10,000. Some of these indicators are population, GNP, public expenditures, military personnel, newspaper circulation, unemployment, life expectancy, and capital formation. No time series are involved since the data is assumed to have been gathered at one point in time, essentially 1959.

The REL user first declares the lexicon - the names of items relevant to this particular data:

```
United States: = name
Canada: = name
U.S.S.R.: = name
population: = number relation
GNP: = number relation
```

There would be one such declaration for each country and each

indicator. Using the language extension mechanism, we might also provide synonyms:

def: Russia: U.S.S.R.

Now we can input the basic data, either in the form of English declarative sentences or directly from a fixed-format card image:

The population of the United States is 183742.
The GNP of the United States is 443270.
The United States' life expectancy is 73.

We will not be concerned with the units in which each indicator is expressed; clearly this can be handled in a variety of ways. At this point it is possible to ask simple, fact-retrieval questions which involve few details:

What is the working age population of Mali?
What is the agricultural land area of the United States / the agricultural land area of Russia?

This mode of analysis quickly becomes unsatisfactory, especially if the amount of data is large. One needs to generalize and summarize across wide areas through the data, and yet be able to check details when desired, in order to cross-check or verify some generalization in the small. The simple summarizations are first, needing only some grouping of the data:

country: = class
def: nation: country
The United States is a country.
Canada is a country.

What is the total population of all countries?
How many nations have a negative GNP increment?

The language extension mechanism proves useful very early, for it allows concepts to take on a life of their own:

def: per capita "land area": "land area"/population

Two clarifications about this definition: (1) "land area" is a variable for all things with the same part of speech as land area, i.e. number-valued relation, thus the definition is a general one for per capita anything; and (2) this definition is totally bound to the context of our present, particular data base. Clearly this is not a generalized definition of per capita - it is only meaningful if we know that a "population" number relation exists. We re-emphasize that REL English is a formal language - not full, unrestricted everyday English. Yet it is a formal language which can be tailored to a subject matter so that the terms used are meaningful and unambiguous. It is the idiosyncratic nature of the above definition of per capita which makes it extremely useable in our present context, and not at all useable in general:

What is the percapita defense expenditure of each nation?

United States	.23
United Kingdom	.08
Canada	.08
West Germany	.04

A representative sample of the answer to this question has been included to show that the phrase "each nation" is a request for a table of outputs and is a quantifier situation. This is a common means of summary, but the usual method is by the use of descriptive statistics:

What is the average school enrollment?
What is the median of communist vote / total vote of all nations?
What is the correlation between communist vote / total vote and per capita GNP over nations?
Which nations' per capita religious vote is greater than 2* the median per capita religious vote?

One component which determines whether such questions as the last one above will really be asked is the time involved in producing their answers (and therefore also the cost). We can easily estimate the amount of elapsed time it will take the REL system to answer this query. There will be some overhead in initializing the system, parsing the sentence, and so on, but this will be under a second. In terms of the data, the REL data analysis system uses a page size large enough so that the class of countries, the population data, and the religious vote data will each fit on a single page. Thus to get the "per capita religious vote of nations" data will require only 3 page loads, since the other manipulations will be done in main memory. If we triple this for good measure, we still have an elapsed time of 1/2 second. The entire query, even with finding the country names to be printed, will take 1 to 2 seconds.

As we have stressed, however, simple statistics is not all of data analysis. Another important part of the process of imposing our conceptual structure on the data consist of subsetting the data into interesting groupings, each of which is to be studied further.

The most common type of international grouping is by geographical region. Geographers find that local proximity has important influences on the development of a nation:

region: = relation
locate: = verb (region of IN is LO)
Europe: = name
France is located in Europe.

What is the average per capita GNP of European nations?
What is the correlation between communist vote and religious vote over countries located in each region?

The geographic breakdown of homogeneity is not the only possible or desirable one. The compilers of the Yale data base considered the matter (Russett 1964, p. 322):

> When we describe Peru as a Latin American country, we are simply locating it in a particular geographic region. If, however, we attempt to explain certain things about Peru, such as its personalismo in politics or its low per capita income, by saying that it is a Latin American country, several interpretations of this remark are possible. The simplest, which we shall call the geographic interpretation of regionalism, is that being a Latin American nation means having a lower per capita income than, say, North American countries, or means having considerable personalismo in its politics. If [our preceeding analysis] had been presented separately for each of the world's major regions this kind of geographic analysis of the broadest ecological sort, comparing different regions with respect to their typically different social and political characteristics, would have been facilitated.
>
> Another way of interpreting the regional clustering of national data for cross-national comparisons would be to make explanations in terms of generalized cultural, political, or social variables which correlate with regional groupings. Thus, instead of talking about East European states, one can refer to communist countries and mean nearly the same thing. At some stage Mainland China and Castro's Cuba would also merit such a label. Even more generally, as this

> Handbook has done, one might describe such states in terms of a very high percentage of the electorate voting for communist political parties. Again it is clear that European nations (and a smaller number of Asian states, some of which do not have elections) are the particularly involved. Although highly concentrated in Europe and North America, economic development is another important generalizable regional phenomenon.
>
> Describing nations in terms of such universalistic variables might be called 'sociological regionalism.'...
>
> As a research focus and a political fact regionalism may mean more than a clustering of geographically proximate states on Handbook profiles, and more than the description or explanation of regional political and social phenomena in terms of sociological variables. A good deal of the literature of social science suggests that relationships between variables will be different for data from different geographic or cultural contexts.
>
> What is the average GNP increment of nations whose executive stability index is greater than 100?
> What proportion of European nations whose per capita land area is less than .5 have an infant mortality rate greater than 100?

The essence of this rather lengthy passage is not that the REL Data Analysis System can handle regionalism, either geographic or sociological, but that it facilitates the imposition of structure on the data by the researcher. One can express and analyze that view which is relevant--and if that particular structure ceases to be relevant, one can impose a new one. One is neither forced to use pre-existing structure nor limited to one's own obsolete conceptualization.

developed: = class
All nations whose per capita GNP is greater than 1 are developed.
underdeveloped: = class
All nations whose per capita GNP is less than .25 are underdeveloped.
What is the average per capita public expenditure of developed nations?
Is the life expectancy of at least 3 European nations less than the maximum life expectancy of underdeveloped nations?

def: "GNP" ratio of "developed" to "underdeveloped": median "GNP" of "developed"/median "GNP" of "underdeveloped"

What is the foreign trade ratio of developed nations to all nations?
What is the life expectancy ratio of underdeveloped European nations to African nations?

The grouping of entities into classes, the use of relations between entities, and the use of language extensions are all powerful conceptual tools by which we can impose structure on our data. The grouping of the United States, France, West Germany, and so on, into developed nations is a process of abstraction--the emphasizing of certain similarities and the exclusion of differences. At the same time the class of nations has been broken into three classes--developed and underdeveloped nations, and neither--a process of ramification of the structure of the data base in order to obtain a more finely detailed picture. The same effects are seen in the use of the relation "region" which allows phrases such as "European nation". The relational structure has the added advantage that

it handles phrases like "nations located in each region" thus allowing us to quantify over the subsets.

The language extension mechanism, though often underrated, is just as important. Language extensions give substance to concepts and push our own notion of relevance into the language. The definition of "percapita" above singles out population as being important, and the ratio of something to population as meaningful. Definitions are not mere abbreviations - they introduce new possibilities into our universe of discourse and thus change the informativeness of our language. Since the phrases which are defined can be re-defined with a different meaning, or even a primitive one, they are essentially independent of the original definition. Once defined, we utilize a concept without going into its definition, as if it were a primitive entity - which it therefore becomes. Definitions are articulations of theory.

This example, and data base, has thus far barely touched the potential inherent in a relational data system: the explicit use of relations between entities. Even though most of our conceptualizations are concerned with the relationships existing between one thing and another, our data and current theory reflect the inability of historical data systems to manipulate interconnected models. The relational data systems are the beginnings of tools for studying interdependencies of a stronger-than-statistical nature. Since the Yale data does not contain

any explicit relations, we shall add one for explanatory purposes:

ally: = relation
West Germany, the United Kingdom, and Japan are allies of the United States

What is the median GNP of allies of the United States?
What is the total population of the United States' allies/ the total population of Russia's allies?

The above use of the relation is again to subset the data - to cut the universe along desired lines. One can also study the relation itself:

Are all allies of allies of the U.S. allies of the U.S. ?
How many nations are allies of both the United States and Russia?
What proportion the U.S.'s allies are developed?

The net of relational structure can become exceedingly complex and begin to reflect some of the realities of the situation. Clearly we cannot do justice to the power of the relational structure - we can only give the briefest glimpse into the complex process of analysis:

trading partner: = relation

What trading partners of each nation are not allies of that nation?
Which trading partners of China trade with some nation that trades with both Russia and the United States?
What proportion of the underdeveloped trading partners of European nations trade with at most 2 communist nations?

List of References

BISS, K., CHIEN, R., and STAHL, F. 1971. R2 - A natural language question-answering system. Proc. of the Spring Joint Computer Conference, : 303-308.

BRAWN, B., and GUSTAVSON, F. 1968. Program behavior in a paging environment. Proc. of the Fall Joint Computer Conference, 33: 1019-1032.

CHEATHAM, T., FISCHER, A., and JORRAND, P. 1968. On the basis for ELF - an extensible language facility. Proc. of the Fall Joint Computer Conference, 33: 937-948.

CODASYL Systems Committee 1969. A survey of generalized data base management systems. New York: ACM Press.

CODD, E. 1970. A relational model of data for large shared data banks. Communications of the ACM, 13: 377-387.

CONWAY, R., MAXWELL, W., and MORGAN, H. 1971. File processing with ASAP, an introduction. Cornell University.

COOPER, W. 1964. Fact retrieval and deductive question-answering information retrieval systems. Journal of the ACM, 11: 117-137.

CRAIG, J., BEREZNER, S., CARNEY, H., and LONGYEAR, C. 1966. DEACON: Direct English access and control. Proc. of the Fall Joint Computer Conference, 29: 365-380.

DOSTERT, B., and THOMPSON, F. 1971. How features resolve syntactic ambiguity. Proc. of the Symposium on Information Storage and Retrieval, 19-32.

EARLEY, J. 1971. Toward an understanding of data structures. Communications of the ACM, 14: 617-627.

FELDMAN, J., and ROVNER, P. 1969. An ALGOL-based associative language. Communications of the ACM, 12: 439-449.

FELLER, W. 1950. An introduction to probability theory and its applications, volume I. New York: John Wiley and Sons.

FEYNMAN, R. 1965. The character of physical law. Cambridge: the MIT Press.

GORDON, G. 1969. System simulation. Englewood Cliffs: Prentice-Hall.

GREEN, C., and RAPHAEL, B. 1968. The use of theorem-proving techniques in question-answering systems. Proc. of the 23rd National Conference of the ACM, 169-181.

GUETZKOW, H. 1968. Some correspondences between simulations and "realities" in international relations. in New Approaches to International Relations, New York: St. Martin's Press.

HERMANN, C. 1967. Validation problems in games and simulations with special reference to models of international politics. Behavioral Science, 12: 216-231.

Inter-University Consortium for Political Research 1970. Archival inventory and service guide. (mimeograph).

JONES, C. 1970. At last: real computer power for decision makers. Harvard Business Review, 48: 75-89.

KELLOG, C., BURGER, J., DILLER, T., and FOGT, K. 1971. The Converse natural language data management system: current status and plans. Proc. of the Symposium on Information Storage and Retrieval, 33-46.

KUHN, T. 1970. The structure of scientific revolutions. Chicago: University of Chicago Press.

LEVIEN, R., and MARON, M. 1967. A computer system for inference execution and data retrieval. Communications of the ACM, 10: 715-721.

LEVIEN, R. 1969. Relational Data File: Experience with a system for propositional data storage and inference execution. The RAND Corporation memorandum RM-5947-PR.

McKELLAR, A., and COFFMAN, E. 1969. Organizing matrices and matrix operations for paged memory systems. Communications of the ACM, 12: 153-165.

NIE, N., BENT, D., and HULL, C. 1970. Statistical Package for the Social Sciences. New York: McGraw-Hill.

PIERCE, J. 1961. Symbols, Signals, and Noise: the nature and process of communication. New York: Harper and Row.

QUILLIAN, M. 1969. The teachable language comprehender: a simulation program and theory of language. Communications of the ACM, 12: 459-476.

RANDALL, D. 1970. Formal methods in the foundations of science. Unpublished Ph. D. dissertation, information science, California Institute of Technology.

RUSSETT, B., ALKER, H., DEUTSCH, K., and LASSWELL, H. 1964. World handbook of Political and Social Indicators. New Haven: Yale University Press.

SALTON, G. 1968. Automatic information organization and retrieval. New York: McGraw-Hill.

SAMMET, J. 1966. The use of English as a programming language. Communications of the ACM, 9: 228-230.

SCHWARCZ, R., BURGER, J., and SIMMONS, R. 1970. A deductive question-answerer for natural language inference. Communications of the ACM, 13: 167-183.

THOMPSON, F. 1966. English for the computer. Proc. of the Fall Joint Computer Conference, 29: 349-356.

THOMPSON, F. 1969. The dynamics of information. talk presented at the Sesquicentennial Celebration of the University of Virginia, (mimeograph).

THOMPSON, F., LOCKEMANN, P., DOSTERT, B., and DEVERILL, R. 1969. REL: A rapidly extensible language system. Proc. of the 24th National Conference of the ACM, 399-417.

TUKEY, J. 1962. The future of data analysis. Ann. Math. Statist., 33: 1-67.

TUKEY, J. and WILK, M. 1966. Data analysis and statistics: an expository overview. Proc. of the Fall Joint Computer Conference, 29: 695-709.

WHORF, B. 1956. Language, thought, and reality. Cambridge: the MIT Press.