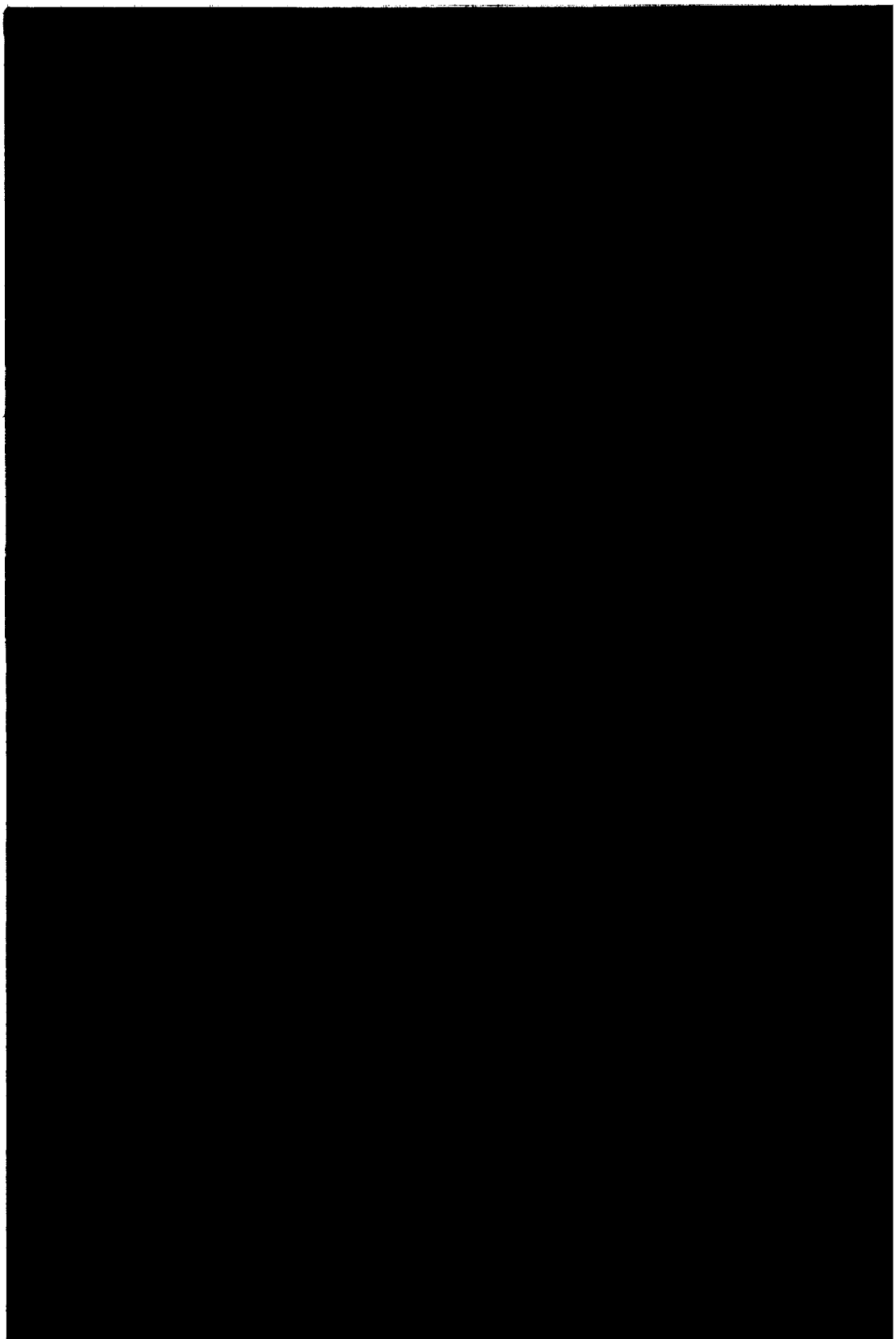


This blank page was inserted to preserve pagination.

DIAGNOSIS

by

GEORGE ANTHONY GORRY

B.Engineering, Yale University, 1962

M.S., University of California (Berkeley), 1963

Submitted in Partial Fulfillment
of the Requirements for the Degree
of Doctor of Philosophy in
Computer Science

June 1967

Signature redacted

Signature of Author . . . . . .
Alfred P. Sloan School of Management, May 12, 1967

Signature redacted

Certified by . . . . . .
Thesis Supervisor

Signature redacted

Accepted by . . . . . .
Chairman, Departmental Committee on Graduate Students

"Work reported herein was supported (in part)
by Project MAC, an M.I.T. research program
sponsored by the Advanced Research Projects
Agency, Department of Definse, under Office of
Naval Research Contract Number Nonr-4102(01).
Reproduction in whole or in part is permitted for
any purpose of the United States Government.

A SYSTEM FOR COMPUTER-AIDED

DIAGNOSIS

by

GEORGE ANTHONY GORRY

Submitted to the Alfred P. Sloan School of Management on
May 12, 1967 in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science.

ABSTRACT

This thesis describes a model diagnostic problem and a computer program designed to deal with this problem. The model diagnostic problem is an abstract problem. A major contention of this thesis, however, is that this problem subsumes the principal features of a number of ostensibly different real diagnostic problems including certain problems of medical diagnosis and the diagnosis of machine failures. A second major contention of this thesis is that strategies for the solution of the model diagnostic problem can be formulated in terms sufficiently explicit to permit their incorporation in a computer program.

The model diagnostic problem assumes that the system being diagnosed (e.g. a person, or machine) is in one of a finite number of known states. Tests can be performed at some cost to discover attributes of the system, for example signs or symptoms in medical diagnosis. The current state of the system is to be deduced from the observed attributes and past experience with similar systems. In the model, this experience is represented principally in terms of probabilities (e.g. the conditional probability of a certain attribute given the system state).

The statement of the model diagnostic problem requires that the diagnostician also account for the cost of various misdiagnoses. In particular for each pair of states i and j, the cost of misdiagnosing state j as state i, l_{ij}, is given. Thus the diagnostician must balance the cost of performing additional tests against the expected reduction in the cost of misdiagnosis. This requirement suggests the value of sequential diagnosis.

A computer program was developed to solve the model diagnostic problem. It consists of 1) an inference function which is based on a Bayesian analysis of attributes and includes a flexible way of dealing with non-independent attributes, 2) a pattern-sorting function which allows the program to detect irrelevant attributes and patterns of attributes corresponding to two different system states, and 3) a test selection function which employs various heuristics to select good tests for the user of the program to perform on the system under consideration. The diagnostic program is specialized for a particular problem by providing it with the appropriate experience. The program is embedded in an environment (set of programs) which facilitates the study of various diagnostic strategies.

The diagnostic program was implemented on the time-sharing system at Project MAC. It was applied to two medical problems, the diagnosis of congenital heart disease, and the diagnosis of primary bone tumors. The results obtained here suggest 1) that a computer program can be of considerable value as a diagnostic tool, and 2) that it is quite advantageous for such a program to perform sequential diagnosis as it interacts with the user.

Thesis Supervisor: Joseph Weizenbaum
Title: Associate Professor of Electrical Engineering and Political Science

Acknowledgments

I would like to express my gratitude to my thesis advisor, Professor Joseph Weizenbaum, for his advice and encouragement. Also I would like to thank the members of my committee, Professors Donald Carroll and Murray Eden, for their illuminating criticism of my work. Special thanks are due to Dr. G. Octo Barnett of the Massachusetts General Hosptial. Dr. Barnett obtained for me the data used in the research. He also provided insights into the nature of some fundamental problems of medical diagnosis. In total his assistance was invaluable.

I would like to acknowledge the assistance of Charles Colemen of the Boston Programming Center of IBM who provided me with valuable facilities for a major part of my research.

Mrs. Jean Stanton typed the drafts and final copy of the thesis, and I owe her particular thanks for her work.

Finally, my thanks to Lucinda who, for a year, had to share her husband with a machine.

G.A.G.
Brookline, Massachusetts
May, 1967

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

This empty page was substituted for a blank page in the original document.

Chapter 1

DIAGNOSTIC PROBLEMS AND PROCESSES

There are many problem areas in which attention is focused on some system. In these areas, the principal problem is to ascertain the current state of the given system. In general terms such a problem is a diagnostic problem. The problem-solver or diagnostician is equipped for his task with information distilled from past experience with such systems, and he attempts to couple this general knowledge with specific observations or tests of the given system in such a way that he can deduce the identity of the current state. The extent of the general knowledge, its organization, and the particular manner in which it is brought to bear on the diagnostic problem, the diagnostic process, may vary considerably among different problem areas, but the general nature of the problem persists.

Thus the medical diagnostician deals with the problem of discovering the "state" of the patient. Through training and experience, the physician has learned the sign and symptom patterns associated with possible diseases from which the patient can suffer. One problem is the effective utilization of this experience which is framed in terms of the abstraction of the disease and the reality of the individual patient. An additional complication arises from the

fact that different diseases may result in similar signs and symptoms. The physician exploits his general knowledge or experience in the selection of a sequence of tests to apply to the patient. The results of these tests provide him with information from which he constructs a more complete picture of the health of the patient. These tests may include simple questions as in the history-taking or complicated medical procedures such as in an exploratory operation. Since tests may exact a high cost (in terms of risk to the patient, patient discomfort, the time of skilled persons, money, etc.), it is the additional task of the diagnostician to properly balance this cost against the potential usefulness of the test results. For these and other reasons, medical diagnosis is often a complex and difficult intellectual problem.

A second example of a diagnostic problem is that of debugging computer programs. A program containing one or more errors can be thought of as a system for which it is desired to determine the state. The state in this case is characterized by the particular combination of errors. The programmer brings his past experience with a variety of programs to bear on this diagnostic problem. By controlling the inputs to the program, applying traces, or altering instruction sequences, or employing a post mortem, he can perform a range of tests on the program. The results of these tests may suggest new tests as well as providing the programmer with new insight into the problem currently confronting him. Like medical diagnosis, program debugging

is often a difficult task, requiring considerable judgment both in the selection of tests and the interpretation of results.

The research reported here is concerned with a particular diagnostic problem and a diagnostic process for solving that problem. It has several aims. The first is to formulate the model of the diagnostic problem in such a way that the definition subsumes the principal features of problems in a number of ostensibly different problem areas. For example, the definition might apply both to medical diagnosis and to program debugging, although it might not be the particular definition employed by diagnosticians in the respective areas. That such a model can be formulated is the major contention of this thesis. The second aim is to develop and investigate strategies for the solution of this model diagnostic problem. Because they are to be stated in terms of an abstract problem, such strategies will be independent of any real diagnostic problem. These diagnostic procedures then are to be embodied in a computer program. This step serves two purposes. First, the program provides an explicit statement of the diagnostic strategies, and thus facilitates the testing of these strategies on particular problems. Second, if the strategies in the program prove effective in practical applications, the program could be of considerable value in computer-aided diagnosis. In the event that this approach were successful, the resulting program may be useful in a number of distinct diagnostic problems, since the methods it employed would be problem-independent. The second

major contention of this thesis is that given a model for the diagnostic problem, effective strategies for the solution of the problem can be formulated in terms appropriate for their implementation in a program.

Such a program for diagnosis could be embedded in an environment (other programs) which would permit two different uses of the program. First, the program could be applied to actual diagnostic problems so that its effectiveness could be determined. Second, the environment could permit the study of a variety of artificial problems, each designed to test a particular aspect of program performance. The first type of application might be termed "open diagnosis"; and the second, "closed diagnosis." Closed diagnosis may facilitate the development of improved diagnostic strategies.

In order for a diagnostic problem to exist, one must have at least some knowledge of the nature of the system being considered. Further the various states of the system must manifest themselves through certain observable attributes.[1] It should also be possible to apply tests to the system at some cost to obtain more attributes. Finally, the general knowledge of the system must include some comprehension of the relationships among signs, states, and tests. The prerequisites are satisfied by the two examples of diagnostic problems presented above. In fact, in simplest terms, this is the basis

[1]The term attribute is used in this thesis to denote any observable manifestation of system state which is employed in the deductive phase of diagnosis. For example, it includes both signs and symptoms in medicine.

for the diagnostic problem studied in this work.

A Brief Outline of a Diagnostic Process

The basic outline of a diagnostic process is as follows. Because the observation of certain initial attributes suggest a diagnostic problem in some system, the diagnostician wishes to ascertain the current state of the system. He selects a test (based on some criterion) and applies it to the system. The application of the test yields to update his current view of the problem. He then applies another test and obtains more attributes. This process continues until the diagnostician makes a decision about the current state. Now this is a most sketchy outline of the diagnostic process. There can be a great deal of sophisticated information processing during each iteration of the process. The point is that test selection and inference are the two principal features of diagnosis as performed in a number of distinct areas. The outline above seems equally applicable to medical diagnosis, qualitative chemical analysis, and the problem of diagnosing a malfunctioning automobile. At this level, then, the diagnostic processes in these and other areas exhibit considerable similarity. Inference and test selection appear to be the keys to diagnostic strategies of some generality. If it could be demonstrated that these features of the process necessarily differ fundamentally from area to area, then there would be little hope for the formulation of general diagnostic strategies. In fact, as will be shown in this work, there is reason to believe quite the contrary.

It appears that, for a number of areas, problem-independent diagnostic strategies can be developed. Note that the strategies employed by experts in different fields may be quite dissimilar, there is no requirement that the strategies developed here resemble theirs. The criterion by which strategies will be judged is how effective they are in particular applications, not how closely they approximate those currently used by human experts.

The diagnostic process then merits careful study for several reasons. First, as indicated above, variations of this problem arise in many different contexts and so the problem is of general interest. Second, the nature of the diagnostic problem is such that it often requires a great deal of intellectual effort to solve it, and any means of improving the problem-solving process will be of considerable value. Finally, the general form of the problem suggests the value of a man-machine partnership in the problem-solving process. Before such a partnership can be established, however, the diagnostic process must be carefully explored in order to determine respective parts to be played by man and machine.

Some Further Comments on the Difficulties of Diagnosis

Diagnostic problems on the whole are difficult ones, particularly for non-experts. Moreover, a great many diagnostic problems constitute considerable challenges to the skill of even the most expert diagnostician. Several factors contribute to the complexity of the diagnostic problem. First, an expert diagnostician must be aware of

a large number of relationships among system states and attributes. As evidence of this, consider the considerable training required to develop the skills of a medical diagnostician. Observation of many different attributes may be required to identify a particular state, and a given attribute may suggest many possible states. These facts coupled with the often large number of states and attributes require the diagnostician to master considerable amounts of information.

Often the relationships mentioned above are known only in probabilistic terms. In such a case, the task of the diagnostician is complicated by the need for some form of probability analysis, a task which generally proves quite difficult for human beings. The accurate assessment of probabilities for a large number of possible states given observed attributes requires extensive training and experience.

Another factor complicating the task of the diagnostician is the difficulty of establishing and maintaining an appropriate structure for all the information relevant to the diagnostic area. Much of the usefulness of that information in the diagnostic process accrues from its organization. A major portion of the expert's skill is derived from his ability to associate particular attributes or attribute patterns with possible system states and subsequent testing strategies. Again extensive experience and training are required to organize the relevant information into a useful associative structure. Unfortunately such a structure is not easily maintained. Associations

which are seldom used may be effectively lost to the diagnostician. As a result, his field of competence tends to become narrow. This tendency is accelerated when the diagnostician must devote considerable effort to the mastering of a continual stream of newly-relevant information.

A computer program to provide general diagnostic assistance to its user would help circumvent some of these difficulties. One of the significant advantages to be gained from the use of a computer is the sheer bulk of information which it can maintain. A diagnostic program would be able to deal with extremely large information structures. Since the program would be independent of the content of the information structure which it employed, that content could be continually updated without affecting the operation of the program (although better information should result in better program performance).

The amount of logical and probabilistic inference with which the program could cope would exceed that comprehensible to a human being. This capability would permit the more extensive exploration of possible testing strategies. Because the program could consider more possible diagnoses than a human being, it would provide a strong safeguard that a particular state is not overlooked in the diagnosis. Finally, a diagnostic program which was "table-driven" would be of all the more value because of its potential applicability to a variety of problems.

Note that diagnostic strategies suited for a computer are not necessarily suited for a human diagnostician. While human diagnosticians possess many special skills and hence serve as good sources of information about diagnosis, the purpose of this research does not restrict the set of possible strategies to those employed by humans. The goal is to develop strategies which enable the peculiar capabilities of the computer to be exploited. Additional insight into the nature of the human diagnostic process and the discovery of ways to improve it would be a valuable, but derivative result of this research.

A Preface to the Material Which Follows

This thesis describes a computer program for diagnosis and presents the results of some experiments performed with this program. The design of the program was strongly influenced by the model diagnostic problem chosen for this research. Although later chapters contain detailed discussions of this problem, a brief summary of its principal characteristics is presented here to provide some perspective on the problem.

The statement of the diagnostic problem considered here assumes that the system is in one of a finite number of states. The object of the diagnosis is to identify the current state of the system. Experience with similar systems is assumed to be available. This experience is in the form of probabilities for the various states and

probabilities of attributes given state. Test costs are constant and known. Furthermore the application of a test does not change the state of the system. Tests are also assumed to be accurate. Finally, it is assumed that the decision loss for each possible misdiagnosis is given in the same units as test costs. This work, then, is concerned with the development of strategies to solve diagnostic problems which can be stated in keeping with these assumptions.

Chapter 2 examines some of the research reported in the literature which has direct relevance to this work.

Chapter 3 presents two views of a diagnostic problem. In the first view, diagnosis is considered as a problem in pattern recognition. The implications and limitations of this view are examined. Then the problem of diagnosis is formulated as a sequential decision problem. This formulation underscores the computational problems associated with the determination of optimal testing strategies. Finally, a discussion of heuristic considerations in test selection is presented.

A system for the study of computer-aided diagnosis is described in detail in Chapter 4. This system includes both a diagnostic program and a variety of programs which provide an environment within which different diagnostic strategies can be studied.

The next three chapters are devoted to experiments performed with the diagnostic system. Chapter 5 discusses the use of the sys-

tem in the diagnosis of primary bone tumors; and Chapter 6, an application of the system to the diagnosis of congenital heart disease. A number of other experiments with the system are discussed in Chapter 7. Chapter 8 presents a discussion of the results of the research and delineates some areas for further investigation.

Chapter 2

LITERATURE SURVEY

A. Diagnostic Programs

In recent years, there has been an increasing amount of work done on various aspects of diagnosis. Some of this work has been aimed at the development of computer programs to perform particular diagnostic tasks. Other work has been more oriented toward the study of human diagnosticians and the strategies they employ. A brief survey of this work is presented in this chapter. Examples of computer programs for diagnosis are discussed. Of particular interest are the diagnostic strategies and models employed by such programs. Finally, some broad views of diagnosis and its attendant difficulties are considered.

By far the greatest concentration of research in computer-aided diagnosis has been focused in the area of medical diagnosis. A number of programs have been written which are capable of performing diagnosis in particular medical areas. These programs, as a rule employ a Bayesian analysis of attributes based on a disease-attribute probability matrix for the given set of diseases considered. That is the programs compute the probability of disease D given the set of attributes A as follows

$$P(D/A) = \frac{P(D)\ P(A/D)}{\sum_{D} P(D)\ P(A/D)}$$

where P(D) is the a priori probability of D.

P(A/D) is the conditional probability of A given D.

The use of a disease-attribute model and Bayesian inference was advocated by a number of researchers as early as 1959 (R1, R2, R3, R4,). While other means of inferring diseases from their attributes were suggested at this time (R5, R6), the Bayesian approach has proved the most widely used. In certain areas the use of analog computers has been explored, but this work will not be reviewed here.

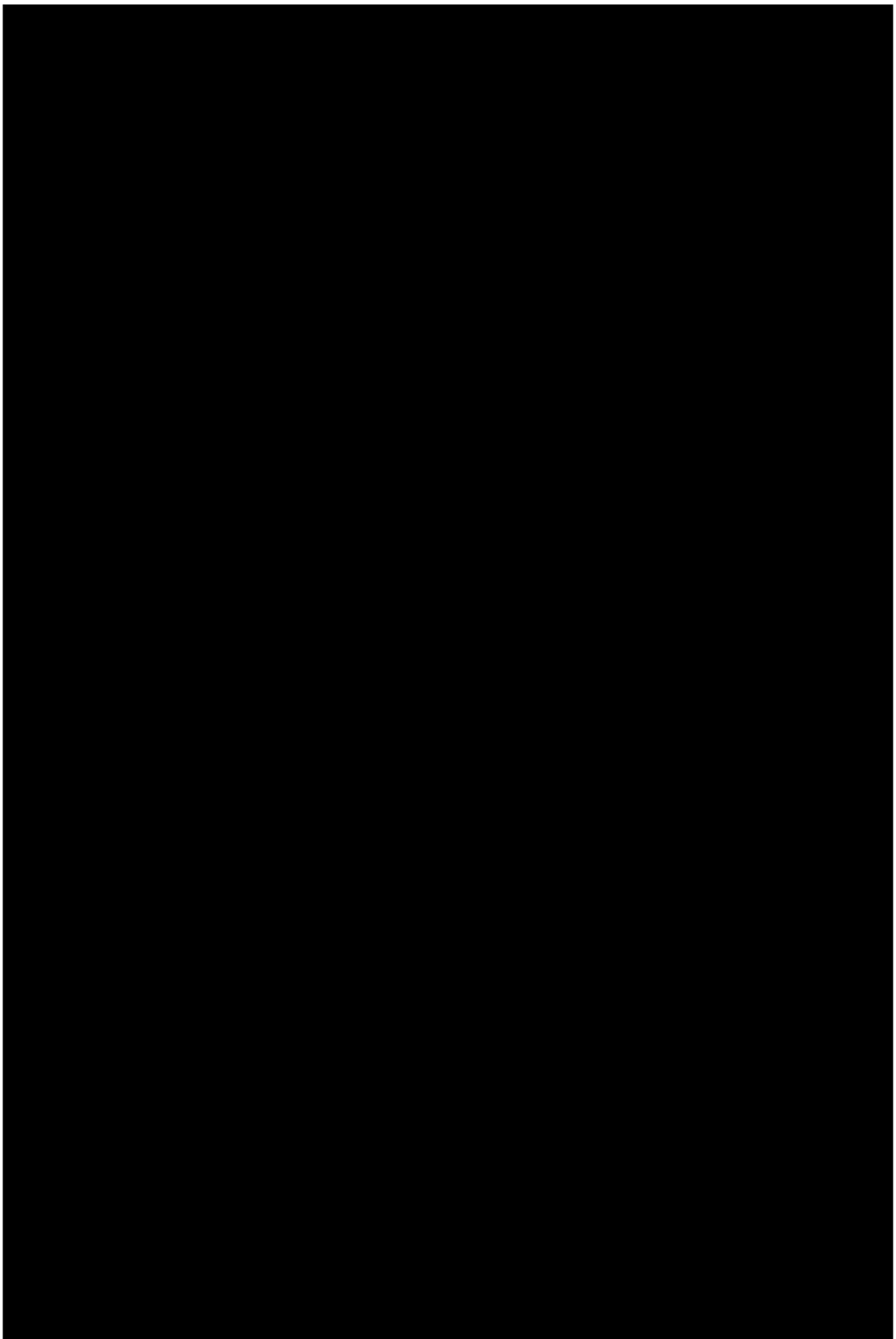
In recent years, computer programs incorporating the Bayesian model have been developed for problems of heart disease (R7, R8), Thyroid disease (R9), epigastric pain (R10), Cushing's syndrome (R11) and others. Some of these programs have enjoyed striking success in attaining levels of performance comparable to that of the expert human diagnosticians. For example, a Bayesian analysis of 268 cases of patients with one of three thyroid problems yielded the accepted diagnosis in 96% of the cases. (R9). In a similar analysis of acquired valvular heart disease patients, a computer program correctly identified 96% of the problems. (R7). In both cases this level of performance compares favorably with that attained by experienced diagnosticians.

In order to provide a more detailed view of the use of Bayesian analysis in computer-aided diagnosis, two studies will be reviewed here. The first is the diagnosis of congenital heart disease; and the second, the diagnosis of thyroid function.

In a series of papers (R12, R13, R14), Warner, Toronto, and

Veasy have reported on the development and use of a computer program for the diagnosis of congenital heart disease. This program employs fifty-seven possible attributes to classify patients into thirty-five different disease classes. The basic strategy employed by the program is the use of Bayes' rule to obtain the posterior conditional probabilities for the different diseases given a particular set of attributes. The necessary a priori disease probabilities and conditional probabilities of attributes given disease were derived from statistical studies of a large humber of known congenital heart disease patients. In certain instances, the statistical information so obtained was deemed inadequate and the probabilities involved were estimated from 1) the available literature and 2) consideration of the pathologic physiology of the disease. The program takes into account the significance of attributes which are absent as well as those which are present. Thus, the absence of cyanosis is significant in the diagnosis. The program is also designed to account for certain mutually exclusive sets of attributes. For instance, if one of a set of mutually exclusive attributes is present, it would be incorrect to consider the absence of the other attributes in the set as additional information in the diagnosis.

The program is used in the following way. For each patient examined, the examining physician determines the presence of absence of the required attributes. When the examination has been completed, the information obtained is punched on cards and fed to the computer



> in the field. Furthermore, the accuracy of the computer diagnosis is still improving with refinements in the data matrix. (R-12)

Overall and Williams (R-9) developed a computer program for the diagnosis of thyroid function. The object was to classify patients into one of four classes: 1) no thyroid disease, 2) hypothyroidism, 3) enthyroidism or 4) hyperthyroidism. By analyzing 879 cases, the authors obtained a disease-probability matrix which included 21 indices of thyroid function. Although over 800 cases were involved in the analysis, not all of the 21 measures were available for each case. Relative frequencies of each attribute were based on the number of cases in which the necessary data were available. Independence of attributes was assumed, although the authors note that this assumption is suspect.

In an extensive series of tests, the program performed extremely well. According to the authors

> . . . computer diagnoses agreed with the clinical diagnoses in over 96% of the cases in which anything like complete data were available. (R-9)

Both of these examples of computer-aided diagnoses lend credence to the belief that Bayesian attribute-disease models of diagnosis may prove extremely useful in a whole range of medical applications.

As noted earlier, not all applications of mathematical methods to medical diagnosis have been founded on Bayesian inference. An interesting example of a different view of the problem involves considering a point in an n - dimensional space (where n is the number of attributes). From past experience with diseases, one can

consider each disease as representable by a class of points in the space. The diagnosis of the current disease is derived from a consideration of the "closeness" of the corresponding point to the classes for each of the known diseases respectively.[1] In a recent paper (R-7), Lerner discusses the use of such an approach in the recognition of handwritten letters and the detection of oil-bearing strata in petroleum geology. In the latter problem (another type of diagnostic problem), he reports that a program based on this method far surpassed the performance of the most experienced experts. He then advocates the application of this method to problems of medical diagnosis and asserts that the possibilities of this approach "considerably exceed those of doctors-diagnosticians."

While this method differs markedly from that employed in the two medical applications above, it shares with them a very important limitation. In Chapter 1 it was suggested that the diagnostician performs two major tasks in his problem-solving. The first task is the interpretation of attributes manifested by the system being diagnosed. An equally important task is the selection of an appropriate testing strategy. All of the programs above map a set of attributes into a diagnosis in one stage. There is no test selection function performed in any of these programs. As a result, all the data which are to be employed by the program must be collected before the program is invoked. There is no opportunity for selective testing based

---

[1]This approach will be examined in more detail in Section A of Chapter 3.

on an analysis of an incomplete set of attributes. Thus, it may happen that the cost of determining a number of attributes (for example, by taking an X-ray) is incurred unnecessarily. While this may not be a major problem in the particular areas discussed above, it is easy to think of situations in which this approach would be highly undesirable. Consider, for example, the computer-aided diagnosis of diseases from a group which exhibit clusters of relatively disjoint attribute patterns. The approach outlined above required the determination of a full set of attributes to be made available to the program. Since only a small subset of the set of all attributes is necessary for a diagnosis, many attributes are unnecessary in any particular application. If the cost of obtaining these unnecessary attributes is high, then the diagnostic procedure will be less than satisfactory. This is because the quality of diagnosis should reflect its cost as well as its accuracy. As Lusted has observed (R-17),

> A great many medical diagnostic tests have been developed to supplement the patient information obtained from history and physical examination. These tests vary greatly in the amount of discomfort to the patient, complexity, and cost. It is obvious that diagnostic tests should be kept to a minimum.

It seems that a more satisfactory solution is to permit the diagnostic program to operate sequentially, choosing tests for the user to run based on a continually updated view of the problem. The program could engage in a dialogue with the user as it performs both the inference and test selection functions of diagnosis. The

testing strategy evolved by the program should reflect the information derived from the attributes observed to date, past experience with similar systems, the cost of tests, and the relative seriousness of various disease states. Part of the research reported in this thesis is aimed at developing a program which satisfies these requirements.

Less has been done with computer-aided diagnosis in other areas. One problem which has received attention, however, is the diagnosis of faults in a computer. Although the problems here are not well understood at present, recent research (R-18) shows considerable promise. Significant results pertaining to the selection of an optimal set of diagnostic tests have been obtained (R-19), but they are restricted to the case of a single fault.

B. Perspectives on Diagnosis

One of the chief motivations for this research is belief that a computer is potentially a very useful tool to be employed in diagnostic problems. The need for such a tool becomes apparent when the difficulty of particular diagnostic problems is considered.

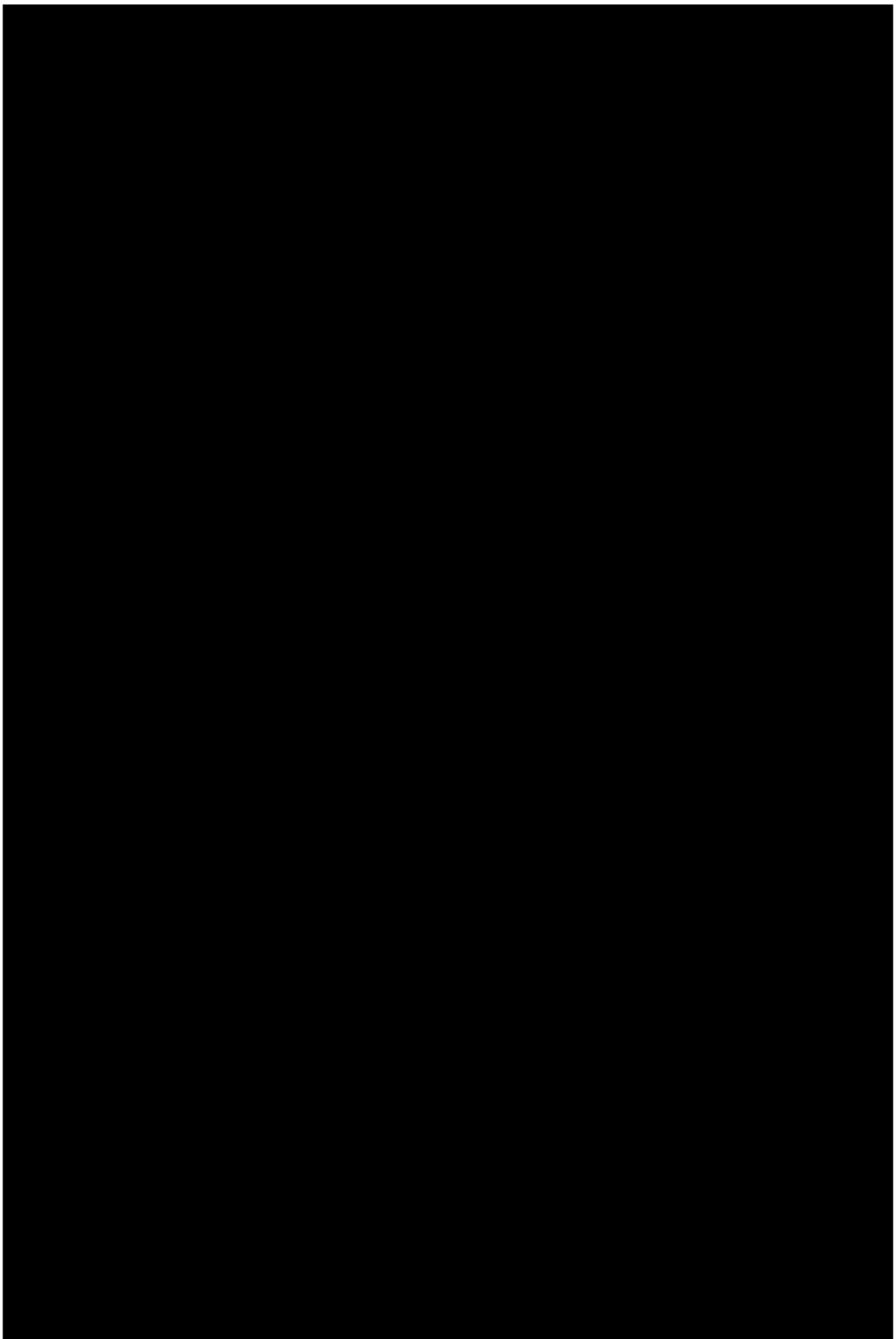
A considerable portion of the effort expended in implementing computer programs is devoted to program debugging. As programming applications become increasingly sophisticated, the complexity of the associated problems of debugging increases at an equally rapid rate. The tremendous effort required to debug a large operating system is a testament to the magnitude of the diagnostic problem in-

involved. This is so even though many of the programmers involved in such an effort are experts.

The non-expert who ventures into the world of programming also faces many diagnostic problems. Often the magnitude of these problems relative to his limited programming skill and experience is such as to prevent him from effectively using the computer in his particular research. In both these cases, there is a need for an improved diagnostic facility. Research into the potential usefulness of diagnostic computer programs seems especially appropriate in this context.

Much the same situation exists in medicine, although here there exists more explicit evidence of difficulty of problems in medical diagnosis and the need for new aids in the problem-solving process. Physicians receive extensive training in their profession, and they devote considerable efforts to the development of their diagnostic acumen. For all their training, however, the difficulties of the diagnostic problems confronting them have resulted in a surprisingly low level of performance. In a recent research report of the United States Public Health Service entitled "Completeness and Reliability of Diagnosis in Therapeutic Practice," the author concludes from an extensive study

> On the basis of available evidence, I estimate if we regard all diagnosable diseases at a given time that are considered of significance for current health as 1, the number of therapeutically determined diseases constitute numerically 0.4. Of this 0.4 nearly half are conditions diagnosed incorrectly, This suggests that correctly



Chapter 3

TWO VIEWS OF DIAGNOSIS

This chapter concerns the theoretical framework for the study of computer-aided diagnosis. Here the nature of the diagnostic problem is examined and the model for the problem is developed. Two views of diagnosis are considered. The first view is that of diagnosis as a pattern recognition problem. This consideration brings into focus those features of the diagnosis which distinguish it from the "classical" pattern recognition problem. The second view involves analyzing diagnosis as a problem in sequential decision-making. The problems arising from this formulation are explained and various means of circumventing these problems are discussed. The view of diagnosis as sequential decision-making is the one taken for this research and so this discussion leads directly to the specification of a computer program for performing general diagnosis.

In the following chapter, a discussion of a program to perform general diagnosis is presented within the framework of the program actually implemented as part of this research. Each of the major logical functions of the program is discussed in turn with the emphasis on the way in which these functions match the requirements of a diagnostic process. In a very real sense, the program can be taken as a statement of an overall diagnostic strategy for computer-

aided diagnosis.

A. DIAGNOSIS AS A PROBLEM IN PATTERN RECOGNITION

Consideration of the diagnostic problem as a pattern recognition focuses attention on some of the more significant aspects of the problem. Also it is quite natural to conceive of diagnosis as a pattern recognition problem. The observable attributes associated with the system of interest in a diagnostic problem do constitute a pattern which is the direct evidence upon which a classification decision is based. Thus a medical diagnostician confronted with an ailing paitent employs his observations of the patient's symptoms and signs in conjunction with his experience and training to deduce the nature of the patient's problem. While there are many features which are shared by the diagnostic problem and a wide variety of particular pattern recognition problems, there are additional constraints on the former which add to its complexity. The purpose here is to explore both the similarities and differences between the diagnostic problem and the "classical" pattern recognition problem.

The classical pattern recognition problem is fundamentally one of recognizing class membership and establishing decision criteria for measuring membership in each class. Given a set of pattern classes the problem is to assign a new pattern to one of the classes. For example in the recognition of handwriting, knowledge of the general properties of individual letters is utilized in the determina-

tion of the identity of that segment of handwriting which is currently of interest. The individual pattern classes may be known in a variety of ways ranging from a set of representative patterns to a functional characterization of the probabilistic process by which patterns of the class are generated. In general, a pattern is comprised of a set of features; each feature being represented by some numerical value. In the handwriting recognition problem, an unknown letter could be represented by numerical values for such features as the height, number of loops and the number of intersections the letter makes with certain reference lines. Such a representation leads quite naturally to the representation of a pattern as an n - dimensional vector where n is the number of features which are taken to be relevant to the classification problem.

Hence, each pattern class can be conceived of as a set of points in an n - dimensional space. Similarly, any pattern which is to be classified can be represented as a point in the space (provided, of course, the same set of features obtains). The problem of classifying a new pattern sample involves determining the "closeness" of the sample to each of the respective classes. For instance, we may decide a certain letter is an "e" because it more closely resembles representatives of the class of known "e's" than representatives of other classes of letters. In the n - dimensional space, this corresponds to measuring the distance (in some abstract sense) between the point denoting the new pattern and those representative of

the various classes. The problem of establishing criteria upon which the "resemblance" of a particular letter to the class of letters known to be "e's" is but one instance of the general problem of deciding exactly how the "closeness" of a sample to various classes is to be established. For a given application, the determination of an appropriate metric is a fundamental problem of pattern recognition.

Consider the schematic of a pattern recognition problem presented in Figure 1. Here two pattern classes are of interest, classes A and B. In this case, there are two features in the patterns and an orthogonal coordinate system corresponding to these features is shown. Notice that in this simple example all members of class A are "closer" to all other members of class A than to any member of class B and vice versa. Unfortunately, this condition does not hold in general. The more common case is to have "close" or intersecting pattern classes. Members of a class can be closer to members of another class than to certain other members of the same class. For example, some handwritten "e's" look very much like "i's" and vice versa. A schematic of intersecting pattern classes is presented in Figure 2. The problem of recognizing the pattern x in these figures involves establishing a metric which can be employed to decide whether x is "closer" to the class A or the class B (or in some cases deciding that x is a member of neither A nor B). The actual decision regarding the identity of x can be based on the cost of misclassification as well as the chosen metric.

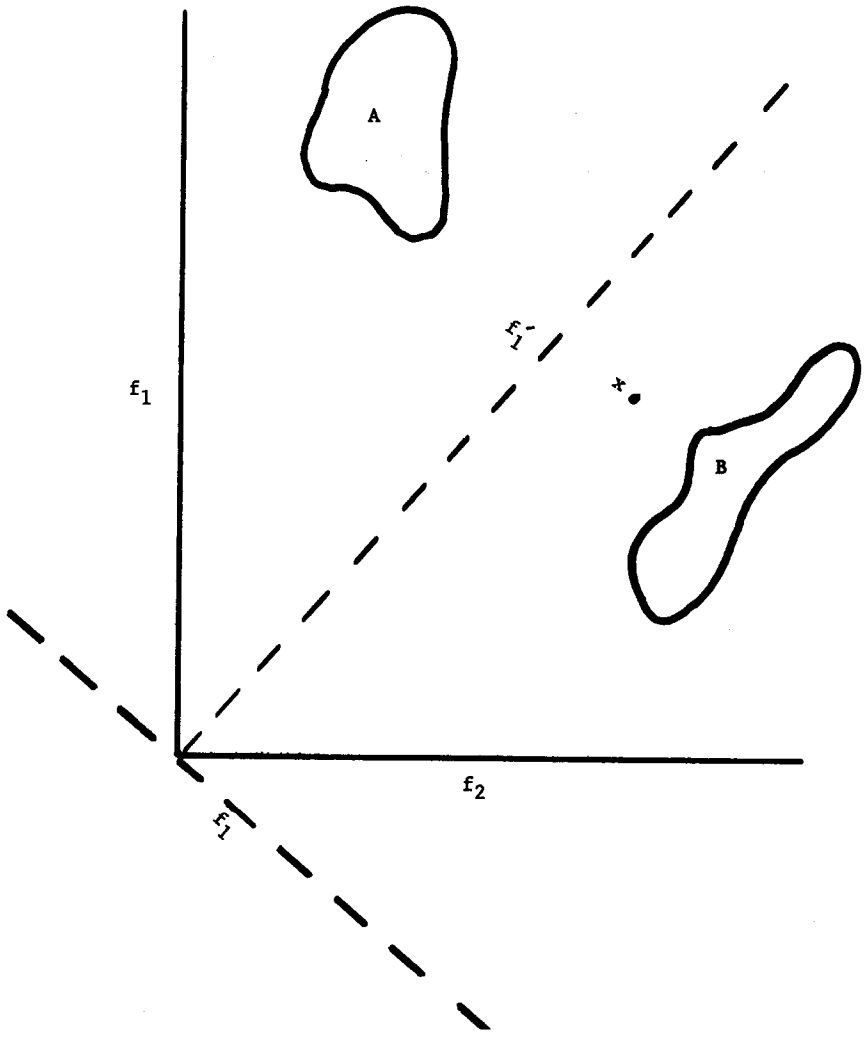


Figure 1

Two Pattern Classes

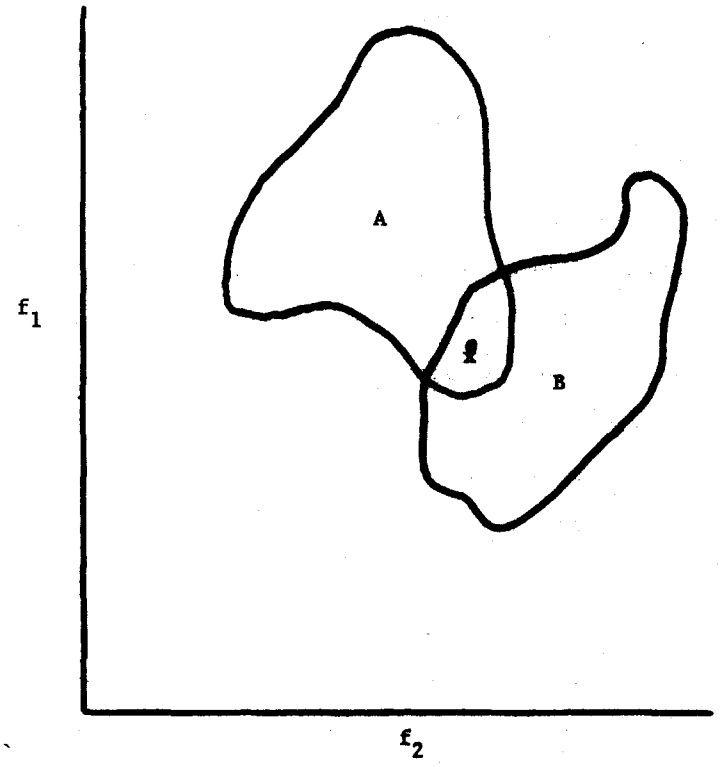


Figure 2

Intersecting Pattern Classes

When the pattern classes are inherently close or intersecting in the space, recognition is more difficult. In some cases matters can be improved by devising class separating transformations. Such a transformation has the property that the classes resulting from the application of the transformation to the original classes are more separated from one another in the transform space. Figure 3 represents the effect of a class-separating transformation on classes A and B. The particular transformation will depend on both the characteristics of the classes to be transformed and the constraints placed upon the transformation. Suffice it to say here that transformations of this type can be derived by solving constrained optimization problems. Given such a transformation, the pattern to be recognized is first transformed and then its "distance" from each of the transformed classes is measured. It is this distance in the transform space which is incorporated in the classification decision rule.

The problem of diagnosis has much in common with the pattern recognition problem discussed above. The pattern classes in the pattern recognition problem correspond to the system states in the diagnostic problem, and there is a similar analogy between particular patterns and sets of attributes. The object of diagnosis is to classify a set of attributes as being a manifestation of a particular system state. Again, the notions of an n - dimensional space and vector representations of attribute sets is suggested. There is an important difference between diagnosis and the pattern recognition method out-

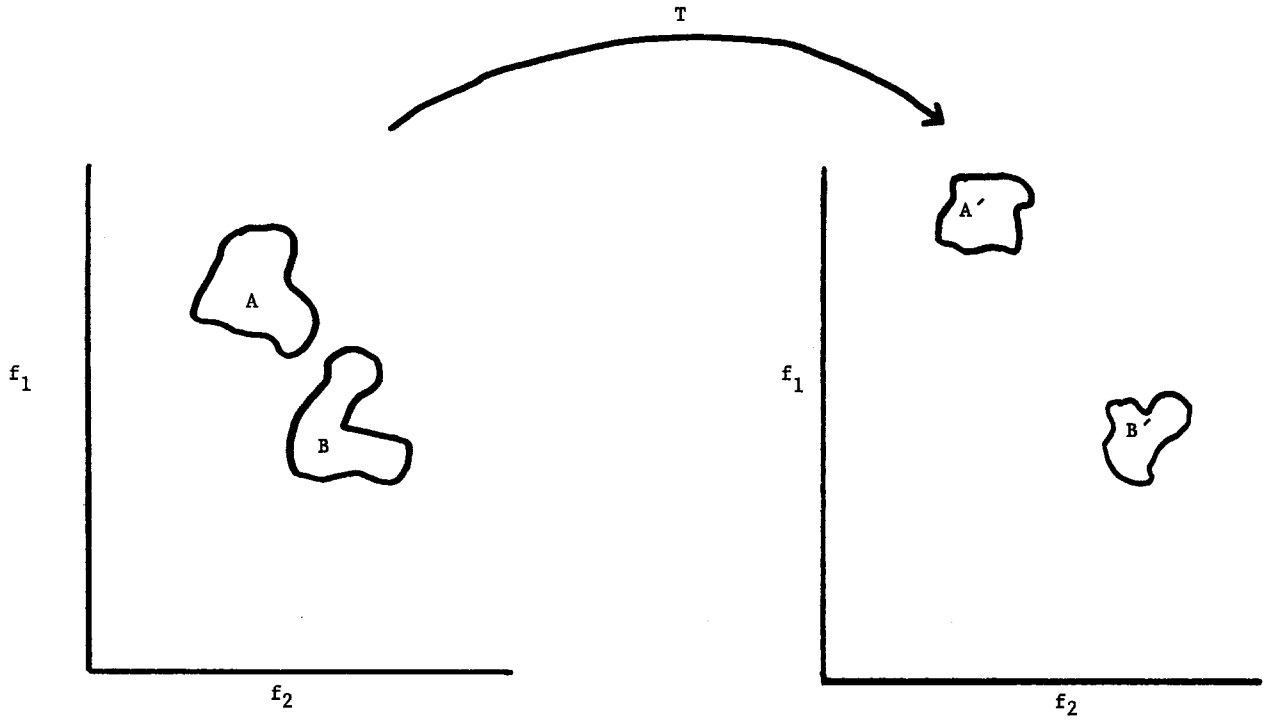


Figure 3

Transformation of Pattern Classes

lined above. In the latter, it was assumed that a pattern to be recognized is given as a point in the sample space. This implies a complete specification of the corresponding vector. In the usual diagnostic problem, the pattern of attributes is incompletely specified. There exist means for obtaining the values of unspecfied components of this vector (tests which can be run, etc.), but in general there is a cost associated with the use of these means. These costs make it advantageous to analyze the diagnostic problem sequentially and to make decisions based on an incompletely specified attribute vector.[1] Doctors, for example, make diagnostic decisions without performing all possible tests on the patient.

Thus, in the diagnostic problem, one is concerned throughout with subspaces of the sample space. The dimensionality of the subspace which contains the pattern vector is reduced by obtaining previously unspecified values for certain pattern features. In general, each value so obtained reduces the dimensionality of the subspace in which the point corresponding to the fully specified attribute set must lie. Because of the costs associated with the tests for particular attributes, a good diagnostic scheme must include some means for assessing the expected value of a test in determining the class to

---

[1]Note that this distinction between pattern recognition techniques and diagnostic techniques is not a necessary one. Certain pattern recognition schemes have employed sequential methods while most medical diagnosis programs have avoided sequential analysis entirely. The distinction, however, does have appreciable generality.

which the attribute vector belongs. While the sequential nature of the diagnostic process complicates its realization, it also offers a potential advantage of the pattern recognition scheme described above. Although an attribute vector may be incompletely specified, the subspace corresponding to it may include only one class. In such a case it may be possible to make the classification decision at that point without investigating the remaining attributes. This reduction in the amount of the processing required for a classification decision is especially significant when many of the system states are represented by disjoint subspaces in the n - dimensional sample space. This reduction can be obtained only if the diagnostic scheme incorporates some stopping rule for the attribute sampling (or testing) process.

So while the pattern recognition problem and the diagnostic problem have a number of features in common, there are significant differences between the strategies indicated for their solution. The former problem concerns the classification of a fully-specified vector into one of a number of known classes. The latter problem is equally one of classification, but the initial specification of the vector is generally incomplete. Part of the problem is to ascertain which tests to run (at some cost) to obtain a more complete specification of the vector. Decisions based on an incompletely specified vector are the rule rather than the exception. Note, however, that there may well be inherently close or intersecting

classes in the diagnostic problem as in the pattern recognition problem.

One aspect of the pattern recognition problem which was not discussed above was that of choosing the coordinate system for the sample space. This has a direct and significant analogy in diagnosis. In the discussion of pattern recognition, it was assumed that the pattern features were given. The efficiency and the accuracy of the recognition scheme often can be improved by the selection of a new coordinate system (set of features). The problem of establishing the coordinate system is often termed the pattern detection problem.

Thus, for example, in Figure 1 the dotted coordinates are in a sense more efficient, for they permit the characterization of classes A and B solely in terms of one coordinate. Again general mathematical techniques are known for establishing "good" coordinate systems for a number of problem types.

Clearly, a similar situation obtains in diagnostic problems. Generally speaking, the attributes considered in diagnostic problems are chosen without any particular regard for the efficient separation of pattern classes. It is apparent, however, that there is potential value in conducting such an analysis for a given problem area. In certain areas, especially in a medical diagnosis, there has been an increasing awareness of the importance of the proper choice of pattern features; a number of articles on the "taxonomy of disease"

have appeared in the literature.[1] While this problem is an extremely interesting one, it is beyond the scope of this thesis. Here the pattern features of attributes for any particular area are taken as given.

This discussion provided only a brief overview of pattern recognition and its relation to diagnosis. The particular type of pattern recognition which constitutes diagnosis will be explored in considerable detail in other sections of this work.

B. DIAGNOSIS AS A SEQUENTIAL DECISION PROBLEM

In this section, the problem of diagnosis is formulated in terms of statistical decision theory. This formulation is in very general terms, but it suggests a number of the factors which complicate particular diagnoses. In many areas of diagnosis, attention is focused on a system. In medicine the system is a human being; in program debugging, a computer program. The object of the diagnostic problem is to determine the state of the system (e.g. the disease in the person or the error in the computer program). This state is one of a finite but perhaps quite large number of possible states. Information about the state of the system can be obtained by performing a variety of tests on the system . Information obtained from testing

---

[1] In recent years, there has been much medical work directed at developing specific tests for diseases. Thus a particular attribute (test result) may indicate exactly one disease.

coupled with experience with other diagnostic problems is employed by the diagnostician in his attempt to deduce the state of the system. In this work, the goal of diagnosis is taken to be the determination of the state of the system of interest. It is assumed that knowledge of the system state will greatly facilitate further (non-diagnostic) action. For example, the identification of the state of a patient as "tuberculosis" may lead directly to a course of treatment. The system under consideration here is a finite state machine. The diagnostician knows about all the states of the machine in the sense that he has available probability distributions which characterize the response of the machine to certain tests given the machine state. In particular, this information relates attributes, the results of the tests, to particular system states.[1] At the outset of the problem, the machine is in a particular, but unknown state and the task of the diagnostician is to employ the available tests to obtain information about the identity of that state. Tests are assumed to be free from error and it is further assumed that they do not alter the state of the system.

Associated with each test is a cost of applying it to the system (called the testing loss) and thus it is advantageous to make a decision about the state of the system based on a limited number of tests. On the other hand there is a decision loss associated with an

[1]An attribute is binary-valued. That is, each attribute is either present or absent. A test is used to determine the presence or absence of some number (perhaps greater than one) of attributes.

incorrect decision. The loss resulting from each particular decision about the unknown state as a function of the actual state is given by a loss function for the problem. For example, the loss resulting from the decision that a tumor is benign when it is in fact malignant is very costly and a diagnostic procedure for tumors should take cognizance of this fact. In general, the possibility of loss for an incorrect decision indicates the value of extensive testing prior to any decision. The problem is to balance the testing loss and the decision loss in a sequential decision function for the problem. This function would specify a diagnostic procedure such that the total expected loss of the final decision is minimized. The following is a formal statement of this problem.

1. The states of the Machine $\underline{M}$ are M_j, $j=1,n$. and the current state is denoted by M_u. It is assumed that M_u does not change during the course of the diagnosis.
2. $\Pi = (\Pi_1, - - - \Pi_n)$ is a vector of a priori probabilities for M_u. That is $\Pi_i = P(M_u = M_i/\xi)$ and ξ denotes experience.
3. $T = \{t_1, - - - t_r\}$ is the set of available tests.
4. $(t_i)q$ is a vector of length q with each $t_i \epsilon T$. It represents a series of tests with test t_i being run at the $i^{\underline{th}}$ stage.

5. $S = \{S_1, -- S_p\}$ is the finite set of possible attributes for M and the set T.
6. $(S_i)_q$ is a vector of length q with each $S_i \epsilon S$. It denotes a sequential set of attributes.
7. d_t is a terminal decision and $d_t \epsilon D_t$ where D_t is the finite set of all possible terminal decisions.
8. $C((t_i)_q, (S_i)_q)$ is the testing loss for a sequence of tests $(t_i)_q$ resulting in the attribute sequence $(S_i)_q$ followed by terminal decision at stage q+1.
9. $P((S_i)_q/M_j)$ is the conditional mass function for $(S_i)_q$ given M_j.
10. $P((t_i)_q, d_t/(S_i)_q)$ = conditional mass function for the testing sequence $(t_i)_q$ followed by terminal decision d_t given the attribute sequence $(S_i)_q$.
11. $\bar{L}(\pi, d_t)$ is the decision loss function.
12. $\theta(d/(t_i)_q, (S_i)_q)$ is the sequential decision function to be determined.

Let $\bar{L}_1(\pi, \theta)$ = the average decision loss

$\bar{L}_2(\pi, \theta)$ = the average testing loss.

then the problem is to determine θ such that

$$\bar{L}_1(\pi, \theta) + \bar{L}_2(\pi, \theta)$$

is a minimum.

$$\bar{L}_1(\ \ , \theta) = \sum_{q=0}^{\infty} \sum_{T_q} \sum_{j=1}^{n} \pi_j \sum_{S_q} \sum_{D_t} L(\pi, d_t)\theta(d_t/(S_i)_q, (t_i)_q) \cdot P((S_i)_q/M_j)$$

$$L_2(\ _1,\theta) = \sum_{q=0} \sum_{t_q} \sum_{j=1} \Pi_j \sum_{S_q} P((t_i)_q, d_t/(S_i)_q) \ . \ C((S_i)_q,(t_i)_q) P((S_i)_q/M_j)$$

where T_q is the set of all $(t_i)_q$

and S_q is the set of all $(S_i)_q$

The great difficulty with this problem is not conceptual but computational. For finite sets of attributes and decisions, the optimal solution can be obtained in principle by laying out a decision tree. Such a tree includes by two types of nodes--decision nodes and "nature's nodes." Nodes of the former type are characterized by 1) a current view of the diagnostic problem as embodied in the probability distribution over the states of the system. (This distribution accounts for both the attributes observed to date and the a priori likelihood of system states in a manner to be made explicit later in this thesis.), and 2) a branch emanating from the node for each alternative available to the decision-maker at the node. In the context of diagnosis, then, there is at each decision node one branch for each possible test which can be run and one branch corresponding to a terminal decision. Once an alternative branch away from a decision node has been chosen by the decision a particular one of nature's nodes is encountered.

Such a node represents the possible outcomes of the decision corresponding to the branch which leads to the node. Each of these "outcome branches" leads to a new decision node. A portion of such a decision tree is shown in Figure 4. The node A is a decision node

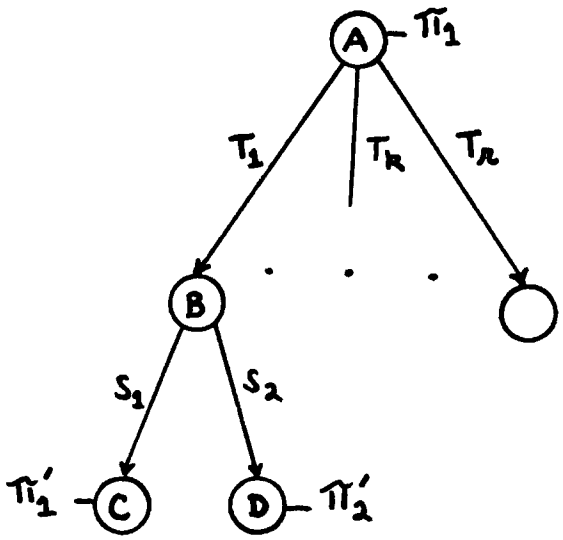


Figure 4

Section of a Decision Tree

which is characterized by the prior probability distribution and history embodied in the path to the node. There is a branch from this node for every relevant test (given the history and π) as well as a branch corresponding to a terminal decision. If a particular test is chosen, say test T_1 in the diagram, a new node (here node B) is obtained. This node is one of the "nature's nodes" mentioned above. There is a branch from this node for each possible test outcome given T_1 and given the state of the diagnosis at B, the conditional probability for each attribute branch can be computed.

If it is assumed that the total number of potentially useful test sequences is finite then the entire tree for the diagnosis can be specified. By folding back this tree in terms of expected loss, one can obtain an optimal decision for every decision node on the tree. This problem is amenable to techniques such as dynamic programming. There is little conceptual difficulty in solving the problem.

The difficulty is the exponential growth of the number of decision nodes with the number of signs and tests. Since diagnostic problems involving large numbers of possible attributes are common, it is expected that the problems of searching large decision trees contribute a large part of the complexity of specific diagnostic problems. One of the major concerns of this research is with the development of effective heuristics for this tree searching problem. While such heuristics produce sub-optimal solutions, it is possible

that the reduction in the size of the search space may more than offset this disadvantage.

As an indication of the potential size of such a problem, consider the diagnosis of a ten-state, twenty-attribute system. Such a case might arise when one was attempting to employ twenty attributes to classify a person into one of ten disease groups. Assuming that there is a test for the presence or absence of each attribute and that each test is run but once, the number of decision nodes in the decision tree for the problem can be expressed as

$$ {}_{n}N_{k} = \frac{2^{k} n!}{(n-k)!} $$

Where ${}_{n}N_{k}$ = the number of decision nodes

k = the depth of the tree

n = the number of tests.

For this example, n is 10, and the number of decision nodes in a tree of depth k is given by

$$ {}_{10}N_{k} = \frac{2^{k} 10!}{(10-k)!} $$

Table 1 gives values of ${}_{10}N_{k}$ for selected values of k. Notice the extremely rapid increase of ${}_{10}N_{k}$ with k. Also, at any given decision node at depth k it is necessary to compare (n-k+1) decisions (one for each of the n-k remaining tests and one for the possible terminal decision). Although in many cases such an attribute set is highly

redundant, it is often possible that a depth of 5 may be required for an optimal decision. In such a case there are still almost a million decision nodes. Even in the simple case of a specific test for each state, there are n! different decision nodes, where n is the number of states. Again the growth of the decision tree with n is enormous.

Table 1

Growth of Search with Depth

k = 0	10^{N_5} =	1
1		20
3		5,760
5		967,680

While there are certain factors in particular diagnostic areas which allow the decision tree to be considerably reduced in size, the determination of an optimal testing strategy reamins computationally infeasible for the most part. The value of good heuristics is apparent from considerations such as the above.

C. HEURISTIC CONSIDERATIONS IN TEST SELECTION

As previously noted, the problem of obtaining an optimal testing strategy for a particular diagnostic area generally will be computationally infeasible. Many diagnostic areas are characterized by overlapping attribute patterns for different states and highly redundant attribute patterns, however, and there is strong motivation for developing "good" diagnostic strategies. Unnecessary or redundant tests may exact a high cost which could be avoided by a more efficient testing strategy. In certain areas of medicine, tests are quite costly and may cause the patient considerable discomfort. If such tests contribute little additional information to the diagnosis, it is especially important that these tests not be employed. A second difficulty is that a poor sequence of tests may generate results which, being unnecessary for a diagnosis, simply tend to obscure the truly relevant attributes. One approach to this problem was mentioned earlier. This approach consists essentially of sharpening the taxonomy of the problem states. While success here can substantially reduce the redundancy in attribute patterns, it will not necessarily make the determination of an optimal testing strategy computationally feasible. While the possibilities of this approach are extremely interesting, they will not be considered here. For the purposes of this work, it is assumed that in any diagnostic area, the attributes for states are given. No attempt is made to improve on the efficiency of the given attributes with regard to the characterization of the states.

A second approach to the problem of test selection is to develop heuristics for the selection process. Such heuristics would employ only limited segments of the decision tree in evaluating the potential efficacy of relevant tests. The general nature of the diagnostic problem is such as to offer two distinct means of controlling the growth of the number of decision nodes considered. The size of the decision tree (the number of decision nodes) depends on the number of tests considered at any decision node, and the depth of the analysis of that tree. By restricting either of these quantities, the diagnostician can limit the growth of the tree. In this discussion, heuristics which limit the number of branches from a decision node will be called breadth-limiting; and those which limit look-ahead, depth-limiting. In what follows, the set of relevant tests for a particular decision node will be taken to mean all those tests which can result in a sign which is manifested by at least one state with a non-zero probability in the prior for the node. The set of relevant tests is a subset of the set of all tests.

Breadth-limiting heuristics are easily formulated. Perhaps the simplest is to limit the number of branches from a decision node to some fixed number. If this number is less than the number of possible test branches for a given node, then a decision rule for selecting (or rejecting) branches must be established. In terms of the diagnostic problem, this means selecting a subset of the

relevant tests for consideration given a prior distribution for the unknown state.

Heuristics which limit the number of branches from a decision node to a certain fixed number have several shortcomings. Principal among these is the problem of the selection decision rule. If certain tests are to be selected over other tests, then some measure of test effectiveness should be employed. That is, one test is chosen over another because by some standards the former is more promising. The difficulty with this is that almost any reasonable measure of expected test effectiveness requires information obtained from a look-ahead in the decision tree. To assess the potential value of a particular test, one needs to consider the likelihood of various test results and the value of these results in improving the current view of the diagnostic problem. If this look-ahead is performed, the purpose of the heuristic is defeated. A breadth-limiting heuristic is intended to select a subset of relevant tests without employing a look-ahead procedure. Then this subset is subjected to further analysis.

Since a breadth-limiting heuristic probably should not employ a look-ahead to obtain information, the only information upon which it should make its decisions is that contained in the current prior distribution and the test cost data.[1] Thus one possible breadth-

[1]This may be overly restrictive, since one can imagine breadth-limiting heuristics which employ a priori probabilities. Such heuristics are not in general very sophisticated, and are not considered here

limiting heuristic is "At any decision node consider at most 5 tests in order of increasing cost." This heuristic obviously ignores all the information embodied in the current prior distribution, and so while it limits the breadth of the decision tree, it does not appear to be a particularly good heuristic.

An alternative breadth-limiting heuristic employs the current prior distribution to generate the subset of relevant tests which are to be considered. For each state there are a number of relevant tests. These tests may produce an attribute which is significant in the diagnosis of the state. Consider, for example, a problem in medical diagnosis in which one of the diseases which currently is being considered as the explanation of the patient's problem is tuberculosis. Since a chest X-ray is a useful test in the diagnosis of this disease, it would be considered a relevant test. On the other hand, the absence of any attributes associated with an injured ankle would exclude an X-ray of the ankle from the set of relevant tests at this stage in the diagnosis. The union of the sets of tests relevant to currently possible states is the set of all relevant tests. By limiting the number of states considered, one can limit the number of branches at the decision node. A heuristic of this type is "Create the total set of relevant tests from the sets of relevant tests for the three most probable states (based on the current prior)." In the above example, if tuberculosis were currently the most probable disease, the diagnostician might choose to

consider only those tests which are relevant to tuberculosis and ignore all others. Note that such a heuristic is only potentially breadth-limiting. There is no guarantee that any test branches are excluded in this way since the same set of tests may be relevant to all states currently being considered. Also the actual number of branches from a given decision node is not specified and generally will vary from node to node.

Such an heuristic has intuitive appeal, however, because it prunes branches corresponding to tests for attributes specific to improbable states. If an attribute for an improbable state is also manifested by a state which is currently quite likely, however, then the appropriate test will be included in the set of those considered. The weakness of this heuristic lies in its sensitivity to the current probability distribution on the states of the system. This distribution can undergo radical change upon the observation of one new attribute. Thus, states which were previously unlikely can become very probable as a result of one new observation. This phenomenon cannot be accounted for by breadth-limiting heuristics based on the current prior distribution. In fact, no breadth-limiting heuristic which does not employ look-ahead can completely account for this possibility. A breadth-limiting heuristic of this type is applied at each decision level, however, and in some sense it can "recover" from a drastic change in the probability distribution. This capability is derived from the consideration of the probability distribution at the current decision node. Thus, when a state which was formerly improbable

at one decision node becomes probable, it will automatically be incorporated in the test selection scheme at the next level. Unfortunately, this state may not become very likely until a large number of tests have been run. If it is the actual state, its probability can remain low simply because the "wrong" tests are being run. Thus a doctor may fail to obtain a chest X-ray of a patient because it seems unlikely that the patient has tuberculosis, when this disease would become very probable if only the X-ray were taken. This, of course, is a general problem encountered with all test selection heuristics.

The evaluation of the heuristic involves a comparison of the benefits of its tree-pruning power with the losses incurred from the sub-optimal testing strategies it produces. In general, a heuristic based on the current probability of various system states appears to be the most promising form of a breadth-limiting heuristic, but its actual value can be determined only in the context of a particular diagnostic problem area. For example, in one area a breadth-limiting heuristic which restricts the search to tests relevant to the n most probable states may prove useful. In another area, tests relevant to all states with current probability greater than some threshold may be considered. Finally, in certain areas breadth-limiting heuristics may be of no value regardless of the particular specification. One of the areas explored in this research is that of evaluating several breadth-limiting heuristics in particular diagnostic problem areas. In such an evaluation, the capability of closed diagnosis may be particularly valuable.

As noted in the beginning of this section, there are two general types of heuristics which reduce the number of decision nodes considered in test selection: breadth-limiting and depth-limiting. As the name of the latter implies, such heuristics limit the extent of the look-ahead in the decision process for test selection. As with breath-limiting heuristics, there are several variations of the depth-limiting heuristic to be considered.

Perhaps the most obvious form of the depth-limiting heuristic is one which sets a fixed depth of search for all branches of the tree. Thus given a particular decision node, the search would proceed down all branches from that node to a depth k, where k is a fixed number. The information derived from this search would then be employed in a decision rule to determine the test to be run next. The parameter k is a relative depth, that is at a decision node at level p, the search is conducted to a depth of $p+k$ before making the decision for level p. An alternative depth-limiting heuristic might employ a variable depth look-ahead. Such a heuristic might attempt to explore more "promising" branches to a greater depth than less promising ones. The difficulty here is to decide which branches are promising. It is, in fact, the general problem of heuristic test selection all over again.

There are several problems to be resolved in the development of any depth-limiting heuristic. First consider the effect on the decision process of limiting the depth of search. If the depth is

limited to k, then the "terminal" nodes will be characterized by probability distributions for the unknown state. (See Figure 4.) Since, in general, there will be a number of states with non-zero probability at any given terminal node, there must be some way of assessing the value of being at the node. One of the major problems in the development of depth-limiting heuristics then is the definition of measures of the desirability of nodes which do not represent a certain diagnosis.

One way of establishing the value of a node is suggested by the presence of a loss function. The value of the node can be obtained by assuming a decision about the unknown state is to be made there. Then the prior distribution for the node and the loss function can be employed to find the expected decision loss for the node.[1] From this loss the value of the node is derived. While this measure seems to be a natural one, it is not without its weakness. The problem with the measure is that it is based on an assumption which is generally untrue. In most cases, one will not make terminal decisions at the nodes which are "terminal" for one state in the look-ahead. For example, if the search depth is limited to 2, the value measure assumes that a terminal decision will be made two tests from this point. Since the actual terminal

---

[1] An additional assumption should be noted here. This is the assumption that given the prior distribution, the minimum expected loss decision is made.

decision may not be made until many tests have been run, this measure distorts the value of tests considered for the current level. The problem is that the values of the loss function at the decision nodes of a given level may bear little relation to the values of the best testing strategies which include these nodes. The potential effectiveness of this "loss function" measure is difficult to assess. The expectation is that it depends upon the particular problem area in which the measure is employed.

A second problem with this heuristic is its potential sensitivity to the actual loss function employed. If the heuristic is very sensitive to the loss function then uncertainties as to the true nature of this function may result in testing strategies which are decidedly sub-optimal. The problems of accurately assessing the loss function for a particular application will be discussed later in this thesis.

The above discussion of breadth-limiting and depth-limiting heuristics purposely considered the two independently in order to make clear the considerations involved. The motivation for such heuristics in test selection is the desirability of reducing the number of decision nodes considered. Since the number of decision nodes is dependent on <u>both</u> the breadth and depth of the search, the heuristics employed in an actual problem will interact. Generally speaking, the depth of the search can be increased only at the expense of the breadth, because there is a constraint on the total number of nodes to be considered. The particular balance of

these two heuristics may significantly affect the effectiveness of the test selection process. An additional complication is introduced by the possibility of changing this balance during the course of the diagnosis when many states are possible. It may be desirable to limit the depth and allow full breadth. This is particularly true if the prior distribution is quite diffuse. As the diagnosis progresses and certain states are eliminated from further consideration the breadth of the tree may be reduced and the depth of search may be increased correspondingly. The relation between the depth and the breadth of the search is an important matter for investigation in the development of heuristic test selection schemes.

More of the practical considerations involved in developing heuristics will be discussed in a later section describing the heuristics employed by the diagnostic program and their relative effectiveness.

Chapter 4

A DIAGNOSTIC SYSTEM

The considerations outlined in the previous chapter led to the design and implementation of a diagnostic system. This system is composed of three major parts. The first is a set of programs which perform the actual diagnostic function. The second is a set of programs which facilitate the study of a variety of diagnostic problems and strategies. The third part of the system is the information structure which contains all the relevant information which these programs employ in performing diagnosis for a given problem area. While the content and, to some extent, the nature of the information structure vary with the particular application, it is convenient to consider this structure as a third general part of the diagnostic system. These three aspects of the diagnostic system will be discussed in detail in this chapter.

The diagnostic system is currently operating on the Project MAC time-sharing system at the Massachusetts Institute of Technology. The diagnostic system is designed to exploit the inter-active capabilities of the time-sharing system. The programs of the diagnostic system are written in MAD and FAP. They make very extensive use of the SLIP-MAD system developed by Professor Joseph Weizenbaum of M.I.T. The SLIP-MAD system (hereafter referred to as SLIP) is a set of list processing functions embedded in the host language MAD. Because

discussions of SLIP are available elsewhere (R-20), only a brief outline of the system is given here.

The basic data structure employed in the SLIP system is a SLIP list. A SLIP list is a list composed of cells where a cell is a pair of adjacent words of storage. The first word of the pair is divided into an identifier field, a link-left field and a link-right field. Each cell in a SLIP list contains a forward (right) link and a backward (left) link. SLIP lists are symmetric in the sense that lists have no particular orientation, the top and bottom of a list are equally accessible. The identifier is used to indicate the type of element stored in the second word of the cell. This element is referred to as the datum. An example of a simple SLIP list is given in Figure 5. Notice that any cell may contain an actual datum rather than a symbolic designation for the datum.

Every SLIP list contains a special cell known as the header of the list. This cell contains the address of the first cell on the list in its right-link field and the address of the last cell on the list in its left-link field. Any storage location which contains the address of a list header in both its address and decrement fields is said to contain the name of that list. A SLIP list structure can be defined as a SLIP list whose data terms may themselves be names of SLIP lists.

There may be associated with any SLIP list a description list or DLIST. If a SLIP list possesses a DLIST, the address of the header

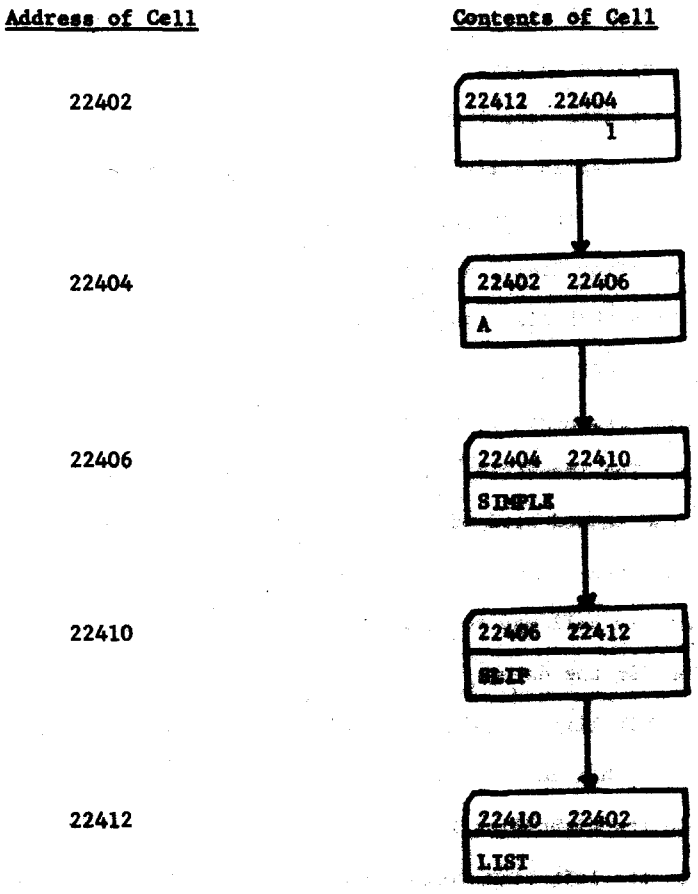


Figure 5

A Simple Slip List

of the DLIST is contained in the left-link of the datum of the header cell. The DLIST, which is itself a SLIP list, is used to store data pairs. A variety of SLIP functions are available for creating and accessing these pairs.

The SLIP library is a set of functions for manipulating SLIP lists. Typical functions permit the reading or searching of lists, additions to or deletions from lists, and the creation or erasure of lists. SLIP maintains an available space list, and the system includes an automatic garbage collection facility.

Because the SLIP library consists of compiled subroutines which can be invoked from MAD or FAP programs, SLIP programs run at object speed. The fact that SLIP is embedded in an algebraic language, MAD, means the full arithmetic and logical capability of the latter is available to the programmer in a list-processing application. These two features make SLIP a convenient language to use in the implementation and debugging of a large list-processing application such as the diagnostic system developed in this research. For this particular application, the need for both the flexibility of list-processing and the algebraic power of MAD is well served by the SLIP-MAD system.

A. THE INFORMATION STRUCTURE FOR THE DIAGNOSTIC SYSTEM

The manner in which the information relevant to a particular diagnostic problem area is organized has a considerable effect on the capabilities of the diagnostic program. The information contained

in this structure for a particular application constitutes the "experience" which the diagnostic program brings to bear on problems. This experience includes relationships between observable attributes and states of the system to be diagnosed. For example, in an area of medical diagnosis, the information structure would contain the relationships between signs and symptoms and the appropriate diseases. Also included in the structure is information about the tests which are relevant to the given diagnostic area and their associated costs. Because of the probabilistic nature of many of the attribute-state relationships as well as other important relationships, the information structure must maintain a large number of individual probabilities. The general content of the information structure will be explained below.

The large number of state, attributes, and tests encountered in many diagnostic areas places a premium on efficient searching of the information base during a diagnosis. The efficiency of search can be maintained at an acceptable level only through the proper organization of the relevant information.

A number of questions were considered in the design of the information structure currently employed by the diagnostic system. One of the principal questions was that of what information should be maintained in the structure. To a large extent, the particular diagnostic problem under investigation here determined the answer to this question. Since the model of diagnosis makes reference only to states, attributes, tests and various probabilities, these factors

constitute the basic information blocks in the structure. Another question is how, given the basic information blocks, these blocks should be related in order to facilitate access by the diagnostic program to the relationships which are significant in the deductive process of diagnosis. For example, the following questions typify the types of demands made on the structure.

- What are the symptoms of pneumonia?
- Which diseases exhibit a rash on the arms as an attribute?
- What is the probability that a patient will have a temperature greater than 103° given that he has pneumonia?

The information structure described here was developed through the consideration of a number of alternative forms, although there obviously are other forms which might serve as well. To a certain extent, the information structure reflects the use of the SLIP system by the diagnostic program. For example, the information structure is a SLIP list structure. While in certain instances this results in inefficient utilization of main storage, this disadvantage was more than offset by the convenience of being able to employ the full SLIP library in the development of the diagnostic system.

A basic information block in the structure is either a state, an attribute, or a test. Each of these basic blocks is represented by a SLIP list in the information structure. In what follows these

blocks will be referred to as state lists, attribute lists, or test lists. A typical state list is depicted in Figure 6; in this instance, the state list corresponding to pneumonia in a medical diagnosis problem. The list name of each attribute list relevant to pneumonia appears on the state list for this disease. There are two data pairs on the DLIST of each state list. The stored attributes are the a priori probability of the state and the print name of the state. The latter is the name by which the user of the program makes reference to the state. In order to facilitate the retrieval of the state list corresponding to a particular print name (as, for example, when the user makes a request for information about the disease pneumonia), all the state lists are grouped on a number of hash lists. Each hash list is a sublist of a list called the master state list. The retrieval of the state list corresponding to a particular print name is effected as follows: First a SLIP function is used to map the given print name onto the integers 0 to N-1, where N is the number of hash lists on the master state list. If the integer K-1 results from this mapping, the K^{th} hash list is searched for a state list with the desired print name. Since the same hashing function is employed in the creation of the master state list, the appropriate list will be found if one exists. Roughly speaking, this technique reduces the average search time for such requests by a factor of 1/N as compared to a search in the absence of hash lists.

An attribute list includes the list names of all the test lists corresponding to tests which can result in the given attribute.

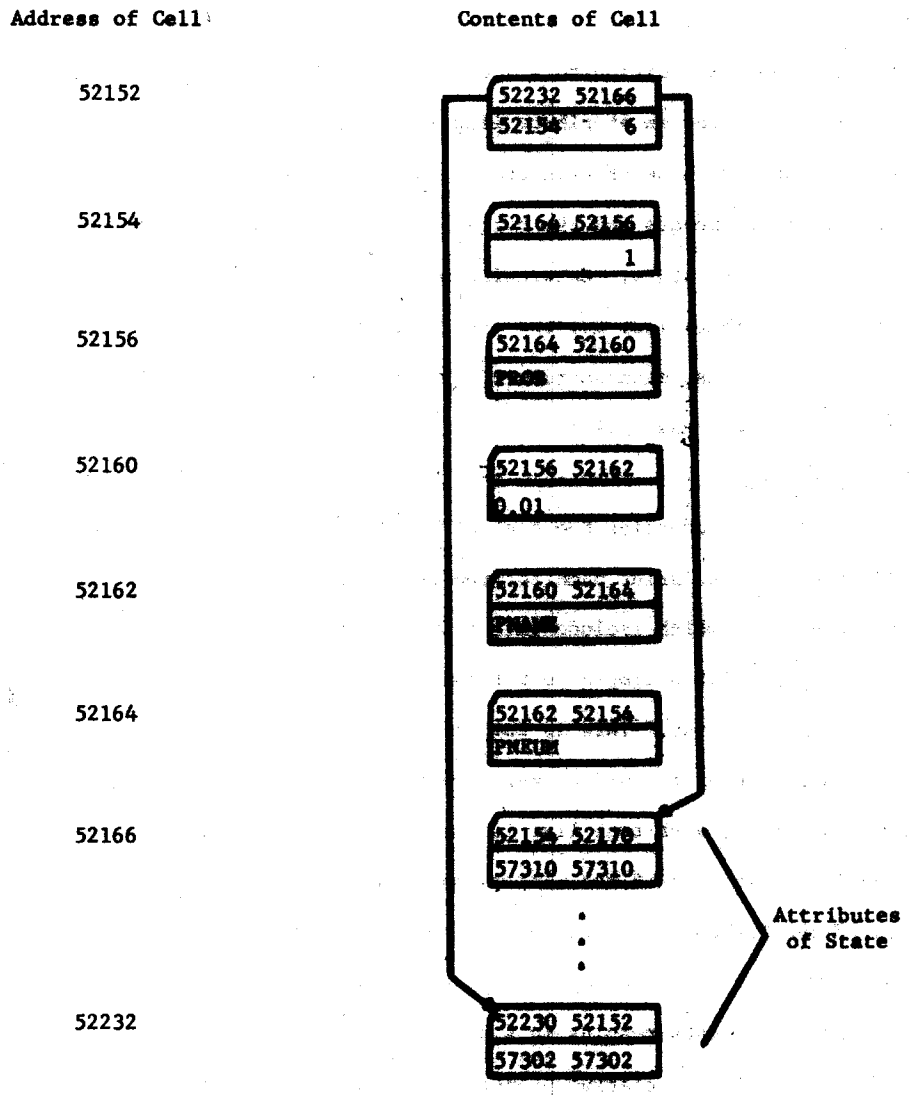


Figure 6

A Sample State List

The DLIST for an attribute list contains a data pair for the attribute print name in addition to a special data pair for a member list. The member list for an attribute list is a standard SLIP list which contains the list name of each state list on which the name of the attribute list appears and the corresponding probability of the attribute given the state. Continuing the example above, Figure 7 depicts the attribute list for the attribute "fever." As in the case of the state lists, each attribute list is a sublist of a hash list, and each of these hash lists, in turn, is a sublist of the master attribute list.

A test list contains the cost of the test and a DLIST. The DLIST contains the print name for the test and a member list for the attribute lists which include this test. In Figure 8 a simple test list is shown with a single cost (independent of state) and a deterministic member list. This is the form of test list used in this research although it would be relatively easy to make it more complex. As above, each test list is a sublist of a hash list, which is in turn a sublist of the master test list. A schematic of a portion of the information structure is shown in Figure 9.

The presence of two-way links between attributes and states and attributes and tests results in a highly associative information structure. This associative property facilitates the accessing of information pertinent to a diagnosis. Thus a search for attributes given state and a search for states given attribute are equally efficient. Similarly the accessing of possible attributes resulting from a

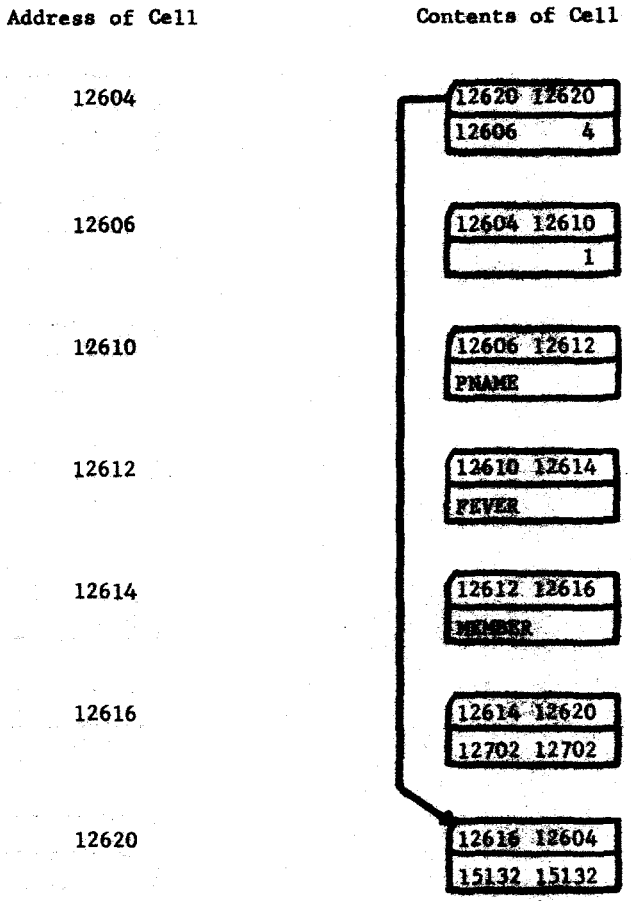


Figure 7

A Sample Attribute List

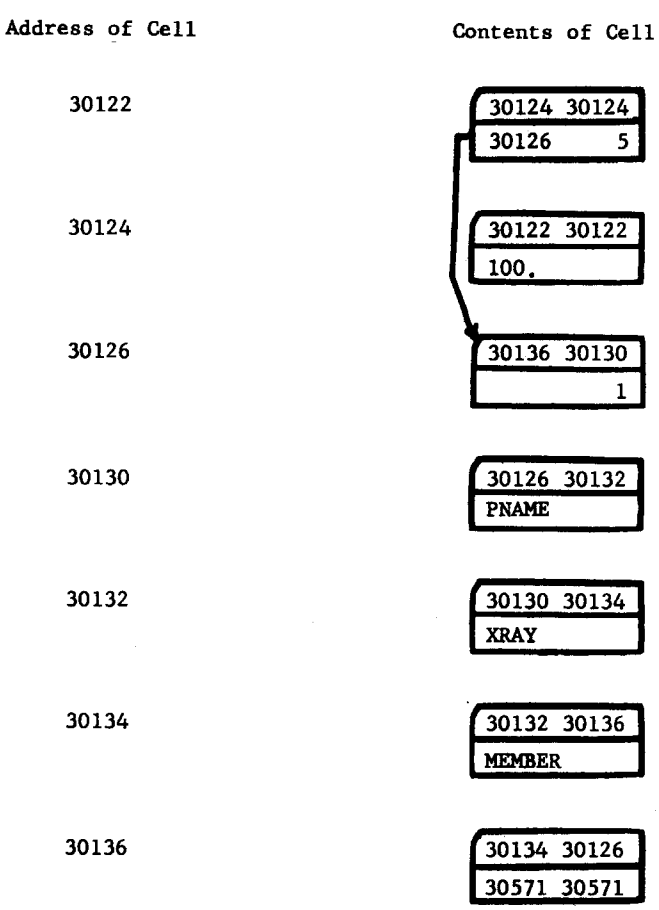


Figure 8

A Sample Test List

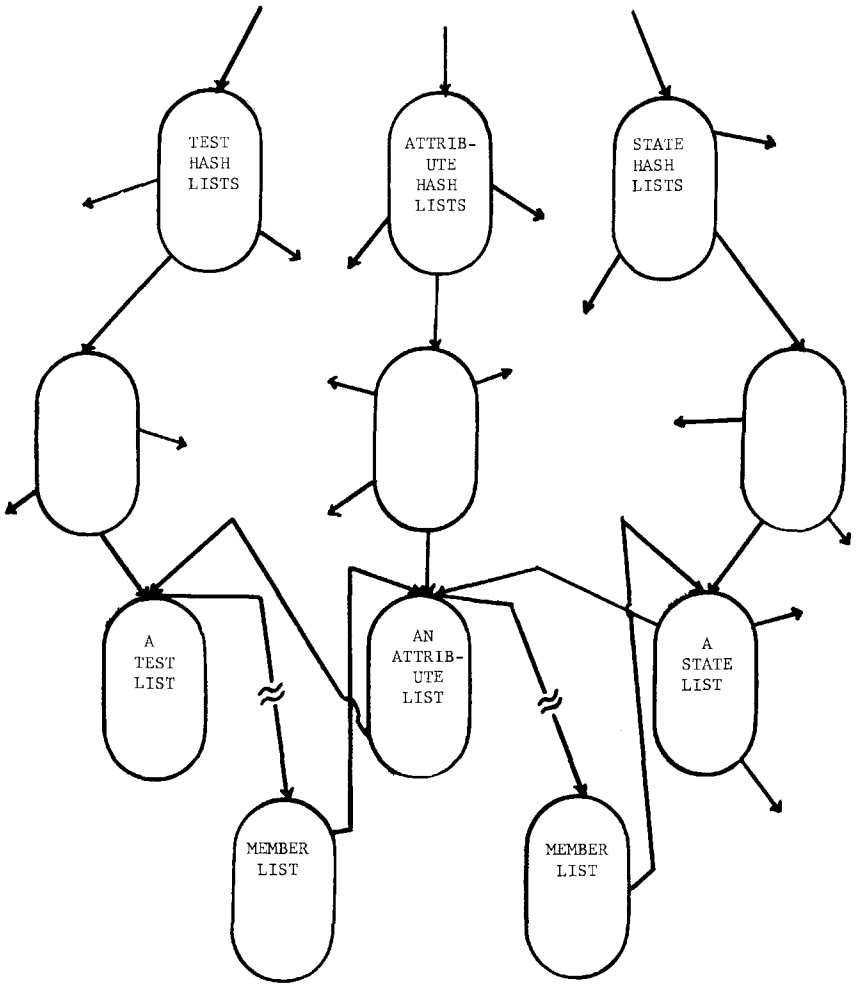


Figure 9

A Portion of an Information Structure

particular test is made straightforward by the presence of the member list.

One example of the importance of this associative aspect of the information structure is its use by the diagnostic program in the initial "pruning" of the space of possible diagnoses in response to the observation of initial attributes. Generally, these initial attributes are presented as the user's statement of the problem. For the program to operate in a reasonably efficient manner, it must use this initial statement of the problem to develop a drastically reduced set of states for further consideration. This is directly analogous to the "pruning" employed by a doctor when upon the observation of a few initial signs or symptoms, he reduces the list of diseases he considers as possible causes of the problem to a very small number relative to the set of all diseases. The diagnostic program would employ the member list for a given attribute list to rapidly determine the set of all diseases which were known to exhibit the corresponding attribute. While this reduction of the search space is crucial to the success of the program, it must not be irreversible if the program is not to be led astray by spurious information or noise. Since it is unreasonable to expect that those who prepare the information structure can anticipate all variations in attribute patterns for a given state, it is expected that the program at times will be confronted with problems involving attributes which are not relevant to the principal problem. The strategies employed by the pro-

gram and the nature of the information structure have a strong effect on the program capability in such a problem environment.

The information structure currently employed by the diagnostic program associates with each state only those attributes which are relevant in the diagnosis of that state. Thus there would be no association between the state "tuberculosis" and the attribute "sore thumb" in the information structure for medicine.[1] The advantage of this is that the size of the information structure is limited. Thus while there may be many attributes, only a subset is associated with any state. As will be discussed later, this creates problems in performing diagnosis in a noisy environment. Certain routines associated with the diagnostic program are responsible for making decisions about the significance of the attributes observed in a diagnosis. The function of these routines is also the subject of a later section.

The discussion of the information structure to this point has implied that the attributes for a given state are taken to be independent. Since in many cases the assumption of attribute independence is not justified, it is necessary that inter-attribute dependencies be representable in the structure. This capability is available in The current program through the use of clustering routine, the

---

[1]Since the program does not determine what information is included in the structure, the user can associate any attributes and states. The point is that certain associations are not expected.

relation-definition routine, and the relation interpreter.

In order to provide a general capability for dealing with inter-attribute dependencies, the diagnostic program must be able to cope with a variety of relationships among attributes. The important relationships most likely vary from one diagnostic problem area to another. It does not seem advisable to attempt to catalog these relationships within the program itself, since it is extremely difficult to predict just which relationships will be required. Also, if the relationships are incorporated within the program itself, it is difficult to introduce now ones as they become of interest in a particular problem area.

What is required then is a flexible facility for the program to accept new relationships and having so accepted a relationship, to incorporate it correctly in the inference process of diagnosis. In an attempt to provide this facility, the diagnostic program provides the user with the means to define a variety of relationships among attributes. A relationship is defined by specifying as a Boolean function the conditions under which the relationship is true. This function is employed by the diagnostic program whenever it is necessary to determine whether the relationship is satisfied for a particular state.

Consider, for example, the case in which it is necessary to account for the time of the appearance of certain attributes of a particular disease. Imagine that for the disease in question the

attribute "rash" appears two days after the appearance of "fever." Let the function BEFORE accept five arguments and be defined as

```
BEFORE (A1,A2,A3,A4,A5) =
    (EQ (MINUS (CHAR A1 A2) (CHAR A3 A4)) A5)
```

Here EQ, MINUS, and CHAR are system primitives (defined by the diagnostic program). The function CHAR is used to retrieve characteristics of attributes. For example, the value of

```
(CHAR TIME FEVER)
```

is the time at which the attribute fever was observed.

By specializing the function BEFORE as

```
BEFORE (TIME, RASH, TIME, FEVER, 2)
```

The relationship for the disease in question can be checked.

Such relationships are defined by the DEFINE subroutine which the user can invoke as required. Relationships can also be built into the information structure when it is first established if they are known to be necessary. To define a relationship among the attributes of a particular state, one uses the CLUSTER routine. This routine re-organizes the state list for the state involved, producing an attribute-cluster. Thus, for the example above, the reorganized state list might look as that in Figure 6. As with individual attributes, a conditional probability given state is associated with each attribute cluster.

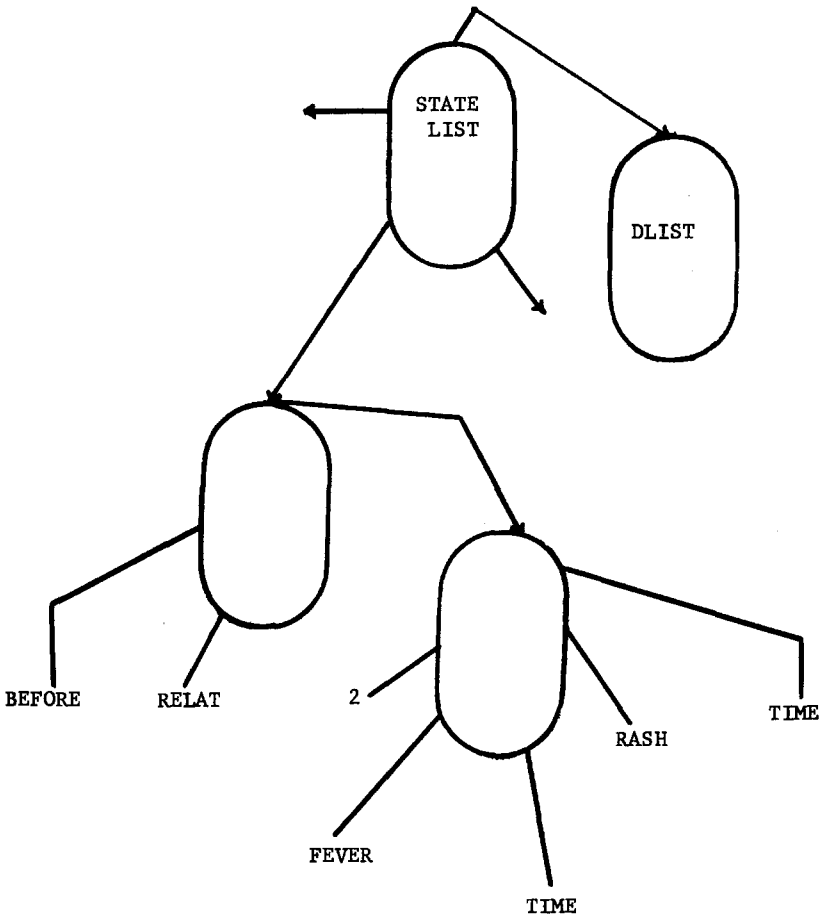


Figure 10

State List with Cluster

Any number of relationships can be defined for the structure provided that they can be expressed in the prescribed manner. Complex relationships can be specified by using functions of functions. Note that attributes remain independent for any state unless a relationship involving them is defined for that particular state. Thus, in one disease "fever" and "rash" may be related in some way, while in another they may be independent.

The diagnostic program employs an interpreter to determine the truth of relationships during diagnosis. The interpreter permits the correct incorporation of relationships in the diagnostic inference. The manner in which the interpreter is employed will be examined in detail later.

B. THE DIAGNOSTIC PROGRAM

The diagnostic program and its associated routines are the heart of the system. These programs embody the various diagnostic strategies employed by the system. When one uses the system in the solution of a diagnostic problem, he interacts with the diagnostic program. This program provides the interface between the user and the facilities of the system. There are three basic functions performed by the diagnostic program. (Although, in fact, each of these functions is delegated to a set of subroutines, it is convenient to consider them as logical functions of the diagnostic program.) In brief these three functions are:

1) The interpretation of the attributes of a particular

problem based on the information contained in the information structure. This function is called the inference function.

2) The selection of tests for the user to apply to the system being diagnosed in order to obtain further clues as to the system state. This is the test selection function.

3) The analysis of the attributes of a problem to determine whether there are irrelevant attributes present or to detect attribute patterns from more than one system state occurring simultaneously. This is the pattern-sorting function.

The design of the diagnostic program permits the alteration or replacement of any of these three functions independently of any of the others. This flexibility is important, because these functions are fundamental to this scheme for diagnosis, and it is necessary to study different versions of the functions. The possibility of changing individual functions without changing the remainder of the program greatly facilitates this study.

Before the diagnostic program can be used in a particular problem area, an information structure for that area must be established. This requires that a disk file containing all the relevant information be created. The disk file can be created using the standard input and editing facilities of the time-sharing. The

formatting of the file, although specified, is quite simple, and if the necessary information is available, the only difficulty in creating the file is dealing with the large amount of information which may be required. The information in the file consists of state-attribute relationships and test cost data. An example of a portion of such an input file is shown in Appendix 1. A system program processes the input file and from it constructs the information structure for the problem area.

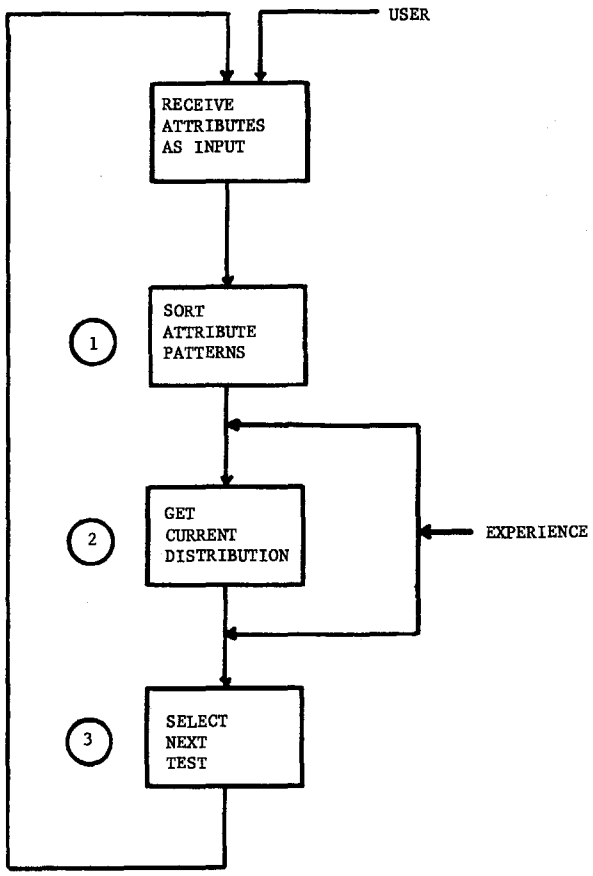
A second file containing the loss structure for the problem area is required by the diagnostic program. At present this loss structure is always a matrix. Any element of this matrix, l_{ij}, is the estimate of the loss for diagnosing state j as state i. The exact manner in which this information is employed will be made clear below.

As a preface to the discussion of the logical functions of the diagnostic program, consider this example of a particular application of the program. Suppose the program currently is set up to diagnose a certain group of diseases. This means that the appropriate information structure and loss structure have been established. A user wishing to invoke the assistance of the program does so by providing an initial problem statement. This statement is essentially a list of the attributes which have been observed. Assume for the example that this list is

- temperature of 102°
- severe coughing
- sore right ankle

As indicated in Figure 11, the program first invokes the pattern sorting function for the current attributes. In this case, the pattern sorting function hypothesizes that the attribute "sore right ankle" is not relevant to the principal medical problem of the patient, and so removes it from the list of attributes for later investigation. After the attributes have been processed by the pattern sorting function, the set of all diseases which exhibit the relevant attributes is obtained and a probability distribution for diseases given these attributes and the "experience" in the information structure is created. The creation of this probability distribution is the task of the inference function. This distribution results from a consideration of both the current attributes and the knowledge of the various diseases. It is the current view of the diagnostic problem assumed by the program.

Now the program invokes the test selection function. The object of this function is to select a good test for the user to apply to the patient in order to gain more information. In selecting this test, the test selection function considers the current probabilities of the various diseases, the cost of each test, and the usefulness of the results expected from the test. The user is informed of the test which has been selected. The test may be as simple as asking the patient questions about his recent exposure to other sick persons, or it may be more involved, for example, a chest X-ray. In any event, when the user has obtained the results of the



1 PATTERN-SORTING FUNCTION

2 INFERENCE FUNCTION

3 TEST SELECTION FUNCTION

Figure 11

Flow of Diagnostic Program

test, he reports them to the program. These test results are new attributes and the program again enters the loop shown in Figure 11. This dialogue with the user continues until a diagnosis has been obtained. A more detailed trace of a session with the diagnostic program is presented in Appendix 2. This brief example provides an overview of the operation of the diagnostic program. In what follows, each of the primary functions of the program will be discussed in detail.

1. THE PATTERN-SORTING FUNCTION

As explained in an earlier section, only those attributes significant to the diagnosis of a particular state are associated with that state in the information structure. Thus the attribute "sore ankle" would not be associated with the disease tuberculosis in the information structure; this means that the name of the attribute list for the attribute "sore ankle" does not appear on the state list for the disease "tuberculosis". Similarly the member list of the attribute list for "sore ankle" contains no entry for the state list of tuberculosis. If the name of a state list does not appear on the member list of a given attribute list, then the conditional probability of the attribute given the state is taken to be zero by the program. As will be discussed in the following section, the particular method of deduction employed by the program (Bayes' rule) results in a zero posterior probability for the state given the attribute. For instance,

if in the course of a diagnosis in which tuberculosis was considered a possible cause of the attributes the attribute "sore ankle" were observed, the updated probability of tuberculosis would be zero. Since the program removes from current consideration any state with zero probability, this approach makes maximum use of each attribute to reduce the set of possible diagnoses.

The problem encountered here is that while "sore ankle" is not an attribute of tuberculosis, one certainly can have tuberculosis and a sore ankle. This is but one example of the more general problem of irrelevant or noise attributes. Unless special precautions are taken, such attributes can eliminate the actual state from consideration when processed by the inference function. A number of solutions to this problem are possible.

One approach is to associate every attribute with every state, employing ℓ probabilities whenever an attribute is not considered relevant to the diagnosis of a particular state.[1] As long as ϵ is greater than zero, no state will be eliminated from consideration in the manner described above. The difficulty is that this method prevents the drastic reduction in the set of possible diagnoses which is necessary for efficient operation of the program. A second approach is to employ the ϵ probabilities as above, but to eliminate

---

[1]This probability might be taken to be the unconditional probability of the attribute. Since this probability may be quite small, the problem discussed here could still be encountered.

from further consideration those states whose posterior probability falls below a fixed threshold. This method is unsatisfactory because the posterior probabilities for the various states can undergo radical change as additional attributes are observed and employed by the inference function. Thus, there is no guarantee that a state with a very low probability in the early stages of the diagnosis will remain improbable with the observation of new attributes. This problem can be even more severe if the noise attributes are the first observed. In either event, the actual state may be removed from further consideration by this method. Another approach is to decide whether an attribute is relevant to the diagnosis or merely noise before it is processed by the inference routines. This is a very difficult task to accomplish given the particular model employed in diagnosis by the program. The model of the system being diagnosed consists principally of state-attribute relationships without any information about causal connections. Thus, the only way to evaluate the relevance of an attribute to the diagnosis is to consider some measure of its probability given the diagnosis to date. Since almost every measure of this kind depends on the current prior distribution, which, in turn, depends on the observed attributes assumed to be relevant, a cyclical argument results.

A second problem arises when attributes characteristic of two or more distinct states are observed, as in the case of an individual with more than one disease. This is more than a problem of simple

noise since the program must detect two or more patterns. Again the methods mentioned above are inadequate to cope with this problem.

The solution to this problem which has been incorporated in the diagnostic program involves processing a number of attribute patterns in parallel during a diagnosis. A pattern is a subset of the set of observed attributes which has the following two properties: 1) At least one state in the information structure exhibits all the attributes in the pattern with a non-zero probability and 2) The pattern is not a subset of any other pattern. If the set of observed attributes contained a number of the attributes of tuberculosis and the attribute sore ankle, one pattern would be the set of tuberculosis attributes. A second pattern would be obtained by choosing a disease for which sore ankle is an attribute and taking the intersection of the set of attributes for that disease and the set of observed attributes. Perhaps the set of attributes obtained in this way, using a second disease on the member list of "sore ankle," might be different from both those previously obtained. If so, this set is still another pattern.

Throughout the course of a diagnosis, a pattern stack is maintained by the pattern-sorting function. A schematic of the pattern stack is presented in Figure 12. Each pattern is represented by a sublist of the pattern stack, and associated with each pattern is the probability distribution for the states of the system given the attributes of the pattern.

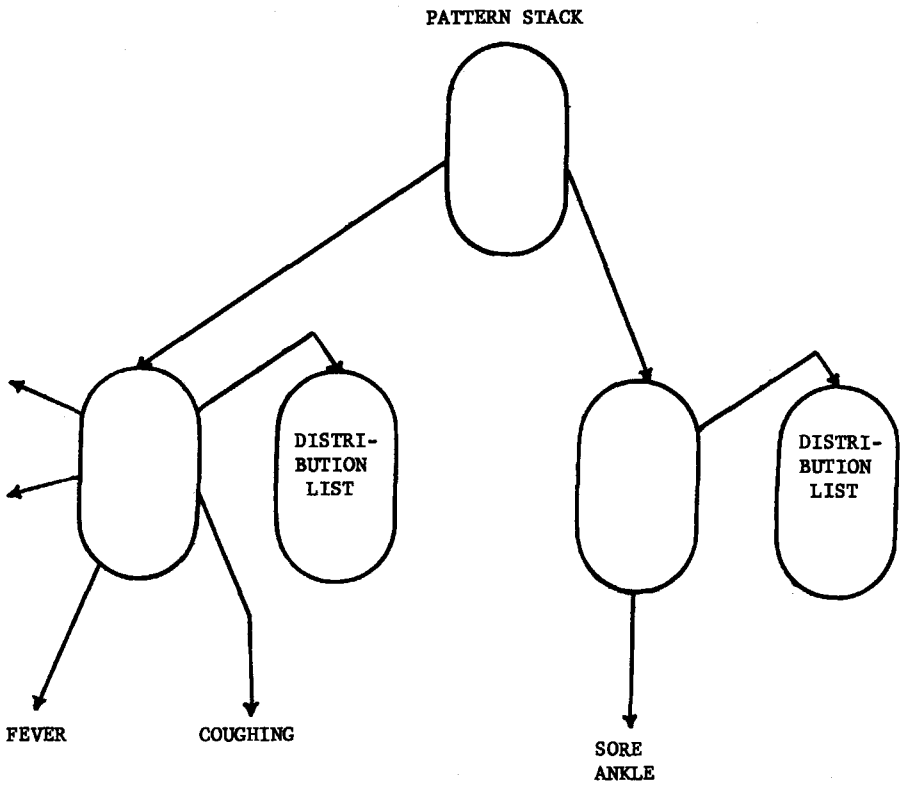


Figure 12

Pattern Stack

Whenever a new attribute is obtained in a diagnosis, it is processed against every pattern in the pattern stack. The new attribute is used to update a pattern if it is relevant to at least one state in the probability distribution for the pattern. After this updating, the attribute is added to the pattern. If no state in the probability distribution of a pattern is known to exhibit the new attribute, no changes are made to either the pattern or the distribution. The actual manner in which distributions are updated to account for new attributes is discussed in detail in the next section on the inference function.

When the new attribute has been processed against all patterns, a routine called PATFRM is invoked to form new patterns if possible. PATFRM retrieves the member list of the attribute list corresponding to the new attribute. For each state on the member list, the set of probability distributions in the pattern stack is searched. If the state is found in this set, the pattern for the state is already in the pattern stack. If the state is not found, the intersection of the set of attributes denoted by the appropriate state list and the set of observed attributes is a new pattern. This pattern and the corresponding distribution for the states is added to the pattern stack. While it is conceivable that this procedure could generate many patterns for a given information structure and attribute sequence, this is not a serious problem. First in most areas the number of distinct patterns which can be formed by this procedure for a

given attribute set is quite limited, because states exist in groups which have overlapping attribute patterns. Secondly, the number of patterns considered can be limited by considering only those patterns with a probability greater than some threshold.

This procedure also includes a provision for removing patterns from the stack. If the inference function determines that the probability of a particular pattern is zero, the pattern and its associated distribution is eliminated from the pattern stack. The contents of the pattern stack, then, can be quite dynamic during the course of a diagnosis as new attributes trigger the addition and deletion of patterns.

As an illustration of this aspect of the pattern sorting function, consider the following example. At a given stage in a diagnosis of a medical problem, three attributes have been observed. These attributes are A, B and C. Also assume that of the diseases represented in the information structure, none exhibits all three of these attributes. A number of diseases exhibit A and B as attributes, however, and so this is a pattern. The point here is that while a disease which exhibits A and B <u>can</u> occur with C also present, C is not considered relevant to the diagnosis of any of these diseases. For the diseases for which C is a relevant attribute A is also relevant. For this situation the pattern stack can be represented as in Figure 13A. Here the symbol π denotes the distribution list for a pattern.

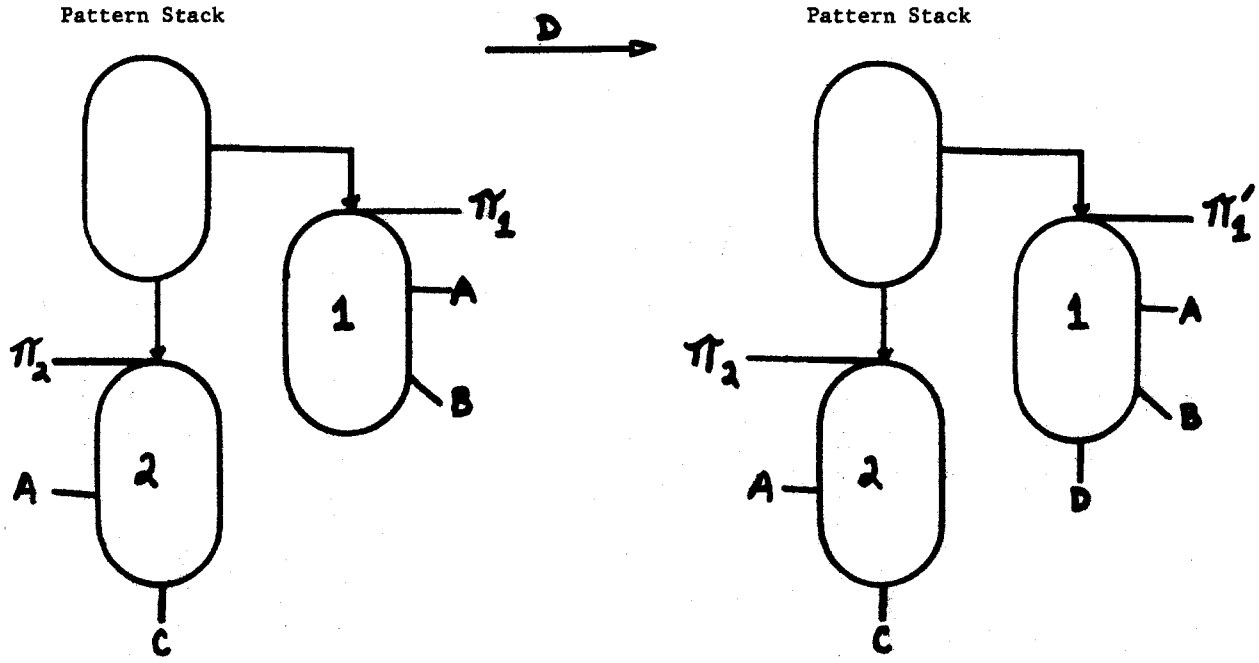


Figure 13A

Figure 13B

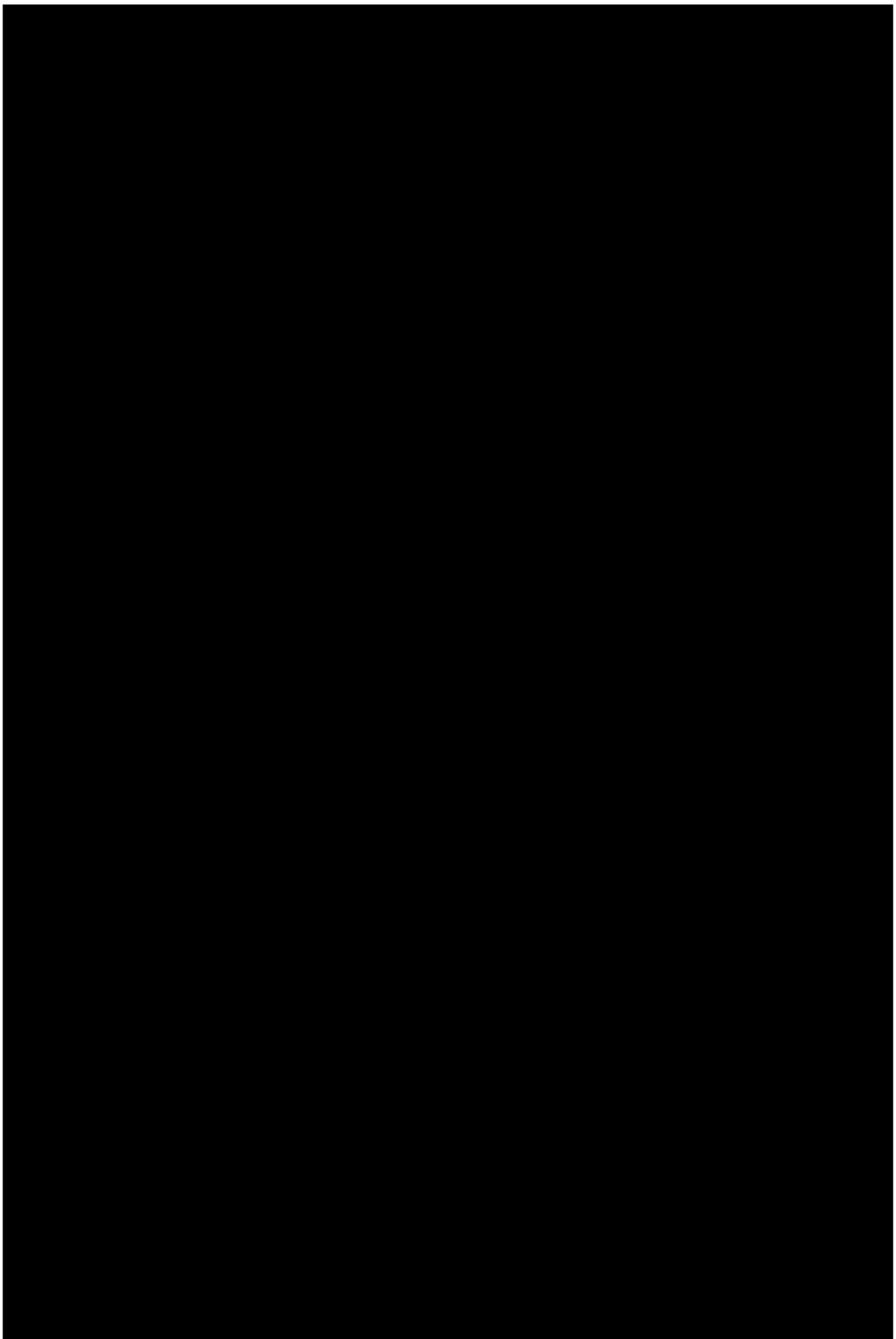
Effect of New Attribute on Pattern Stack

Now when the new attribute D is observed, it is processed through the pattern stack. Assuming that the new attribute is relevant to some of the states in distribution π_1, this distribution is updated by the inference function to produce π'_1 and the attribute D is added to the pattern. Attribute D is not relevant to the second pattern in the stack, and so this pattern and its associated distribution remain unchanged. Finally, the routine PATFRM is invoked to search for new patterns. Assume that no new patterns are formed. Thus, at the end of this phase in the processing of the new attribute, the pattern stack appears as in Figure 13B.

Now in the event that there is more than one pattern in the stack, the diagnostic program must make a decision as to which pattern to diagnose. Thus, the program must generate a hypothesis about the significance of the various patterns in the stack. For example, if one pattern corresponds to a majority of the attributes of tuberculosis, and the other to a single attribute "sore ankle," it is extremely important for the program to give priority to the former pattern. The problem is to establish pattern selection rules which will make the "correct" decision in such a case.

One consideration which is relevant to the selection of a pattern is the seriousness of the states suggested by the pattern. For this reason, an attribute quite specific to a very serious disease will strongly influence the course of a medical diagnosis.

In order to account for the relative seriousness of different



and π_j = a priori probability of state j.

Values of θ decrease with increasing seriousness of states. This can be seen in the following simple example.

		LOSS		
	1	2	θ
1. Benign tumor	0.7	0	1,000,000	300,000
2. Malignant tumor	0.3	100	0	70

While other more sophisticated measures of seriousness can be developed, this simple one was deemed suitable for the purposes of this research.

Once the seriousness of the various states has been established, the problem of pattern selection can be solved in a quite reasonable way through the use of the Bayesian model. For each pattern, a conditional distribution on states can be obtained by the inference function. For each pattern, the distribution is conditioned on the attributes of that pattern alone--all other patterns are ignored. Thus for the k<u>th</u> pattern

$$\pi_j^k = \frac{\pi_j P(\{S_{ik}--S_{mk}\}/M_j, \mathcal{C})}{P(\{S_{ik}--S_{mk}\}/\mathcal{C})}$$

Where π_j^k is the conditional probability of the j<u>th</u> state (M_j) given the pattern $\{S_{ik} \ldots S_{mk}\}$.

The <u>seriousness measure</u> for the k<u>th</u> pattern is given by

$$\gamma_k = \sum_{j=1}^{n} \pi_j^k \theta_j$$

and the pattern selected is the one with minimum γ.

This measure has several desirable properties. Consider the case of an attribute which is very specific to a serious disease. If that attribute is observed, the conditional probability for the serious disease given the pattern containing the attribute will be close to one. Since the corresponding value of θ is small, the value of γ for the pattern will be small. Hence this pattern will quite likely be selected. On the other hand, if the attribute is not specific to the serious disease, the conditional probability for the disease given the pattern will be less; and the resulting value of γ, greater.

The measure also favors a pattern which contains many attributes provided that the pattern strongly indicated one or more serious states. The posterior distribution does not have to be spiked, however, for a pattern to be chosen. For example a pattern which results in equal probabilities for six states may also be chosen if the seriousness of the individual states so warrants. This measure accounts for both the specificity of a pattern and the seriousness of states associated with the pattern. In this respect, it seems to be a good way to select patterns for investigation.

A routine called SELECT chooses the current pattern for the diagnostic program, and this pattern may change from time to time as additional information is gathered by the program. The current pattern is the one employed by the test selection function for

evaluating tests. Before each use of the test selection function, SELECT chooses the current pattern based on all information currently available.

A number of other processing routines affect the pattern stack during the course of a diagnosis. Recall that whenever the pattern sorting function produces more than one pattern in the stack, the selection of a pattern for further diagnosis constitutes a hypothesis about the significance of a group of attributes. If a consistent diagnosis for the current pattern is obtained, then the hypothesis is tentatively confirmed. If there are no other attributes to account for then a consistent diagnosis for all attributes has been obtained. Otherwise the remaining patterns must be considered. It is possible that a second pattern is being diagnosed, new attributes may prove the hypothesis about the first pattern to be incorrect. In this case, the attributes in this pattern can no longer be considered accounted for. These possibilities are dealt with in the following way by the pattern sorting function. The program maintains a list called the "unaccounted-for" list, and on it are all those attributes which have yet to be attributed to a particular system state. When the current pattern is "diagnosed" or assigned to one state, the attibutes in the pattern are removed from the unaccounted-for list, and the pattern itself is marked. A marked pattern is ignored in test evaluation, although it is updated with new attributes whenever appropriate. When the current pattern has been marked, all unmarked patterns are deleted from the stack. Then PATFRM is called

for each attribute in the unaccounted-for list. Patterns are formed using the unaccounted-for list as the total attribute set. If the unaccounted-for list is empty, a consistent diagnosis for all attributes has been obtained. Otherwise, the diagnosis continues on the new patterns.

This means that attributes which are included in marked patterns are not utilized in the formation of new patterns at this time. If, for example, the total attribute set were (A, B, C, D) and (A, B, D) had been tentatively diagnosed, the only unmarked pattern would be (C). This is true even though there may be states which exhibit both C and A. If, however, the test selection function chooses a test which can detect A, A will be added to the unmarked pattern. This is because the program always consults the history of the diagnosis before requesting the user to run a test. If on the other hand, the program would normally account for C without employing knowledge of A, it will do so.

If a new attribute causes the probability of a marked pattern to become zero, a special recovery procedure is invoked. First, each attribute of the marked pattern is transferred to the unaccounted-for list. If one of these attributes is added to the list, it is also processed against all the other patterns in the stack. When the stack has been updated with such an attribute, PATFRM is invoked to check for new patterns based on this attribute. Finally, the marked pattern is deleted from the pattern stack, and diagnosis continued.

Thus, the contents of the pattern stack may be quite volatile during a diagnosis, although cases of extreme volatility are not expected to occur very often. In any event, the use of the pattern stack permits the program to deal with noise and multiple patterns in a reasonably efficient manner. By allowing the user to interact with the program during diagnosis, it is possible to employ his judgment with regard to the merits of pursuing particular patterns.

2. THE INFERENCE FUNCTION

In general, the observation of a new attribute provides the diagnostic program with additional information about the current state of the system being diagnosed. Based on this observation, the program may significantly alter its estimate of the likelihoods of the various states. This section discusses in detail the manner in which the program incorporates observations of attributes into its current view of the diagnostic problem. The routines which process new attributes for their effect on the current view of the problem collectively are called the inference function.

The basic analysis of attributes and inference done by the diagnostic program is based on Bayes rule. Bayes rule can be stated as follows

$$P(M_j/S_t,\mathcal{E}) = \frac{P(M_j/\mathcal{E})P(S_t/M_j,\mathcal{E})}{P(S_t/\mathcal{E})}$$

where $P(M_j/\mathcal{E})$ is the probability that the current state is M_j

conditional on the total experience to date.

$P(S_t/M_j, \mathcal{E})$ is the probability that the system will exhibit attribute S_t given that it is in state M_j and the diagnostic experience $\mathcal{E}$.

$P(S_t/\mathcal{E})$ is the probability of the system exhibiting S_t unconditional on state.

$P(M_j/S_t, \mathcal{E})$ is the conditional probability that the state of the system is M_j given $\mathcal{E}$ and the newly observed attribute S_t.

The quantity $P(M_j/\mathcal{E})$ is called the prior probability and $P(M_j/S_t, \mathcal{E})$ is called the posterior probability of the state M_j. The observation of the attribute S_t increases the experience or information available on which to make a decision about the unknown state. The posterior probability is an adjustment of the prior probability to account for the new information. After this adjustment has been made, the posterior probability is the new prior probability when further attributes are observed. Consider the following example of this basic inferential process:

Suppose there are only two states relevant to the current diagnostic problem, M_1 and M_2, and three attributes S_1, S_2 and S_3. The a priori probabilities for the two states as well as the conditional probabilities for the attributes given the states are presented in Table 2.

TABLE 2

EXAMPLE FOR BAYESIAN ANALYSIS

	A priori probability	Conditional Probability of Attribute/State S_1	S_2	S_3
M_1	0.8	.1	.4	.1
M_2	0.2	.7	.6	.9

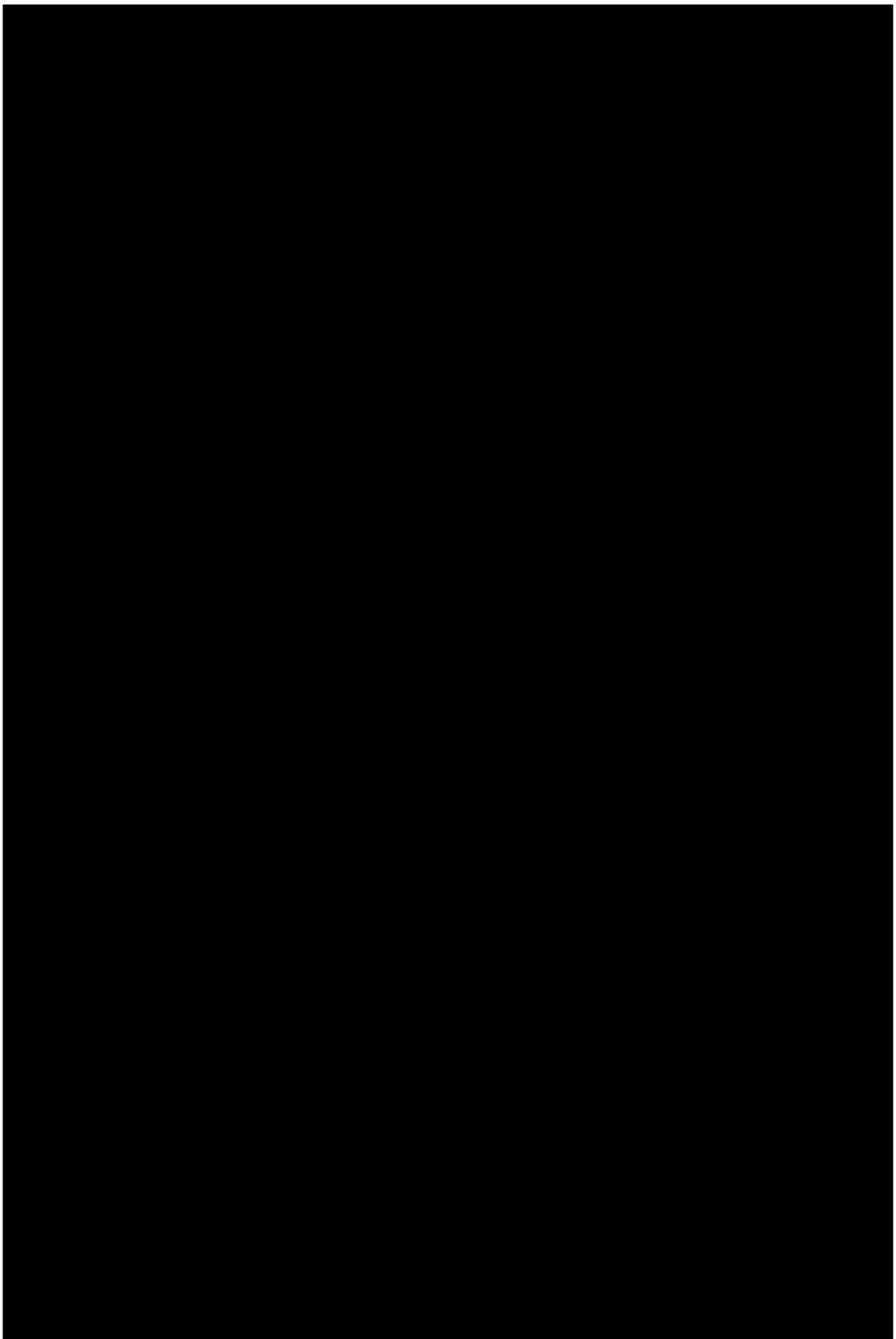
The initial experience of the program, before any attributes have been observed, is embodied in the a priori probabilities. Thus, the current distribution on states is (0.8, 0.2). Now assume that tests employed in the diagnosis reveal the presence of attribute S_1. According to Bayes rule, the posterior distribution is (.82, .18). That is

$$P(M_1/S_1,\mathcal{E}) = \frac{(0.8)(0.8)}{(0.8)(0.80)+(0.2)(0.7)} = 0.82$$

$$P(M_2/S_1,\mathcal{E}) = \frac{(0.2)(0.7)}{(0.8)(0.8)+(0.2)(0.7)} = 0.18$$

Thus, the new attribute has little effect on the view of the problem taken by the program. If two more tests yield the attribute S_2 and then the attribute S_3, the corresponding distributions are:

$$P(M_1/S_1,S_2,\mathcal{E}) = 0.75 \qquad P(M_2/S_1,S_2,\mathcal{E}) = 0.25$$



tern and its distribution list are removed from the stack. While Bayes rule is easily applied in principle, the inference function must include special routines to insure that inter-attribute relationships and the "history" of the diagnosis are correctly accounted for in the probabilistic analysis.

The routine UPD which performs the updating of the pattern stack based on the observation of a new attribute is to a large extent a simple encoding of Bayes rule. The routine, however, does not obtain the requisite conditional probabilities directly. Instead, it calls PIJ to obtain the conditional probability of attribute "j" given state "i" and the history of the diagnosis to date. The reason for this indirection in the accessing of probabilities is really a pragmatic one. The insulation UPD from the probability-retrieving process allows changes in this process to be made without affecting the basic inference process.

As noted, the function of PIJ is to retrieve conditional probabilities from the information structure. In the simplest case, this involves retrieving a number directly from the information structure. When the attribute of interest is involved in an attribute cluster for the given state, the process of determining the conditional probability is more involved.

The general form of an attribute cluster is either

a. $(\theta_1\ R_1)$

or b. $(\theta_1\ R_1 \oplus \theta_2\ R_2 \oplus \ldots \oplus \theta_n\ R_n)$

where R_j is an inter-attribute relationship;

θ_j is the conditional probability of R_j given the state

Φ is either "exclusive or" or "or."

Here R_j can be any inter-attribute relationships (including functions of functions, etc.) as long as it does not include Φ. The reason for this restriction is to eliminate ambiguity from the probability assignments. In fact, the restriction does not limit the class of logical relationships which can be defined, only the form which individual members may assume. Thus, for example, R_j might be the cluster for the relationship

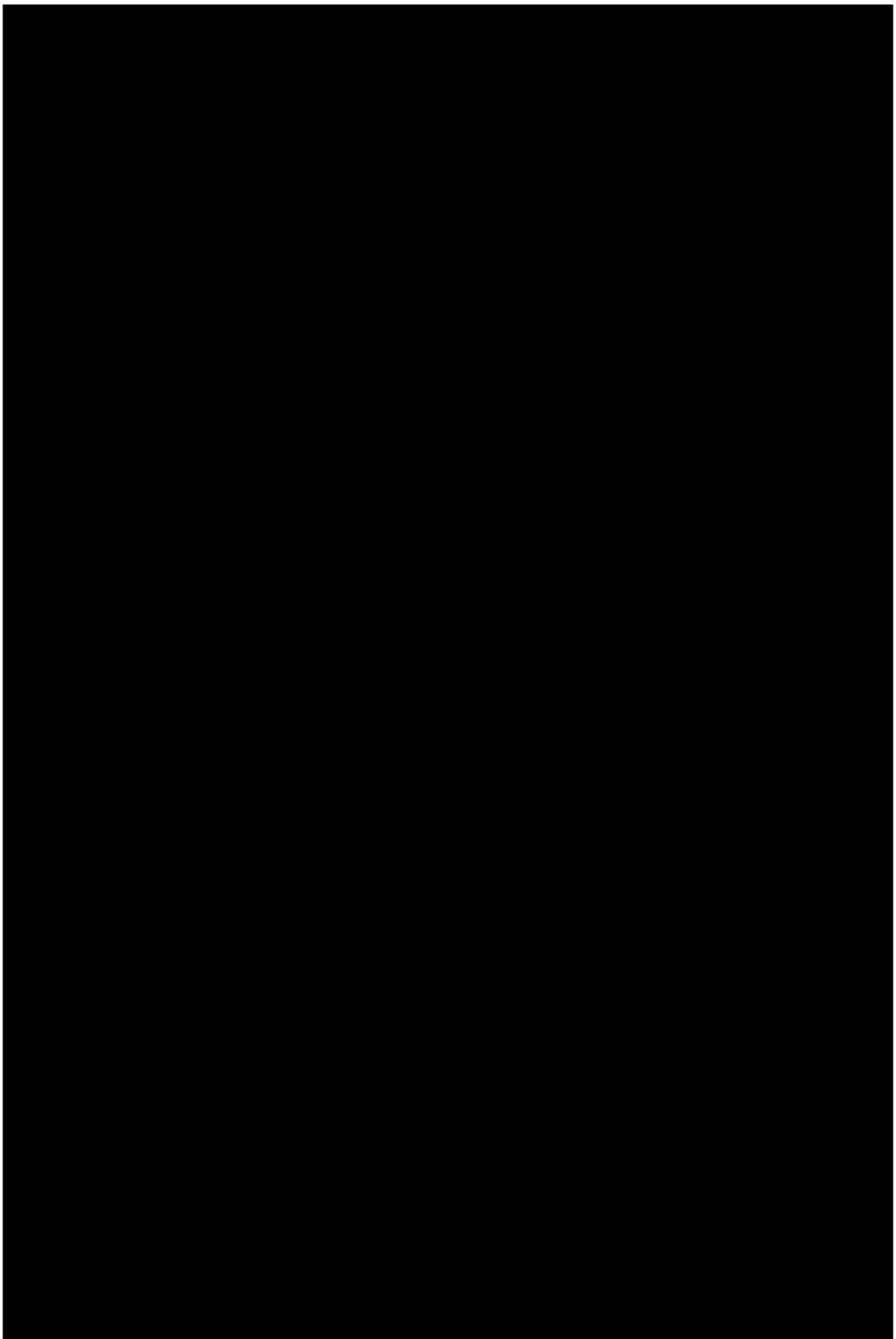
"Either A_1 precedes A_2 in time or A_1 does not appear at all."

In order to evaluate the conditional probability of an attribute involved in an attribute cluster, PIJ must be able to evaluate the truth of the relationships R_j. It does this by calling the routine INTERP to determine the true value of each R_j. INTERP is an interpreter, which retrieves the definitions of any functions involved in R_j and applies these definitions to the appropriate arguments from the attribute cluster. The interpreter employs a push-down stack and recursive calls in the evaluation. All functions are reduced in this way to their component primitive functions. Routines to evaluate the primitive functions are built into the system.

The operation of the interpreter differs in certain aspects from that of a normal interpreter of Boolean functions, because this interpreter must deal with variables whose current value is unknown.

For example, suppose the relationship under consideration for a particular state M_j is "A_1 precedes A_2 in time" with probability 0.5. Assume A_1 has just been observed and the conditional probability of A_1 given the state is desired. If A_2 has not yet been observed, the relationship is incomplete (or from a logical standpoint, undefined). From a Bayesian point of view, however, the conditional probability is well-defined; it can be obtained by assuming that A_2 will in fact follow A_1 in time. This assumption results in a value of 0.5 for the conditional probability of A_1 given M_j. If A_2 is observed later, then its conditional probability can be obtained in a similar manner, but the prior observation of A must be taken into account. This means that the desired probability of A_2 is conditional on the state M_j and the previously observed A_1. Hence the proper conditional probability is 1.0.

In general terms, the interpreter assumes the truth of any relationship which is incomplete unless that relationship is demonstrably false given the current information of the diagnosis. The interpreter must also indicate whether any attributes involved in a cluster have actually been observed. Given these modifications of the interpreter function, the routine PIJ can deduce the proper conditional probability for the given attribute-state pair. PIJ embodies a number of logical tests on the truth of the R_j and the number of observed attributes involved in each. For the types of relationships allowed in the information structure, these quantities are sufficient to deter-

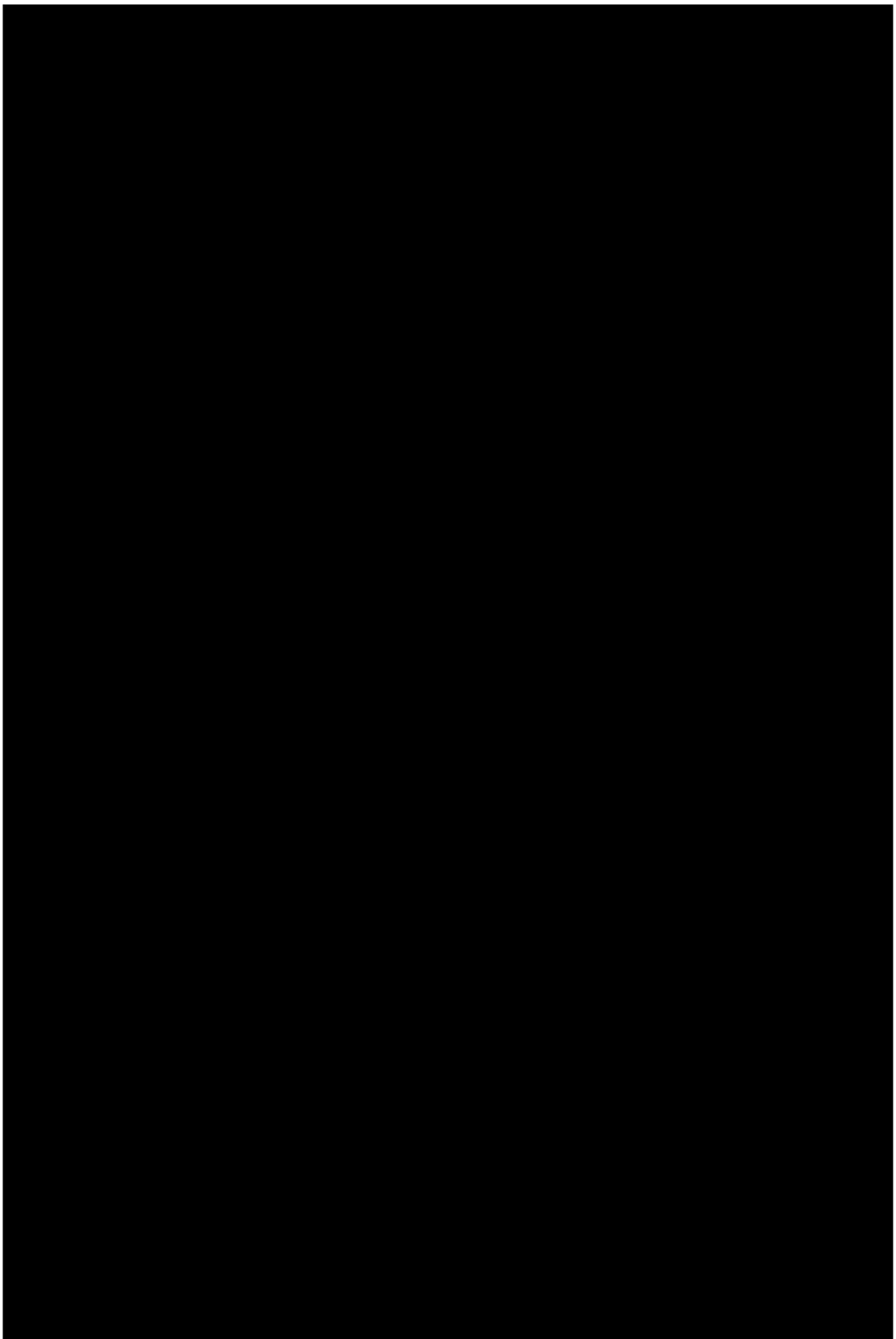


the relevant attributes are first presented by the user. Through the use of the interpreter, the diagnostic program is able to deal with variety of relationships within a particular problem area.

3. THE TEST SELECTION FUNCTION

The value of heuristics for test selection in diagnostic problems has been underscored in previous sections. In this section, a particular test selection program is discussed. This program (which is, in fact, a number of subroutines) is the one employed in the diagnostic program. The nature of the program strategy and organization is explained and some of its limitations are noted.

From the model of a diagnostic problem discussed in Chapter 3, it will be recalled that one of the major tasks in diagnosis is the selection of a good set of tests to apply to the system. The determination of such a testing strategy involves a consideration of both the costs of tests and the information which they are expected to yield. Thus, any heuristic for the test selection process should reflect these considerations. Another consideration involves the amount of computation involved in applying the heuristic in a particular diagnosis. In order to facilitate the study of a class of such test selection heuristics, the test selection function was designed to be in large part independent of the particular heuristics employed. While the class of heuristics permitted is not particularly large, it does include heuristics which lead to markedly different test selection



searched by the test selection function during a particular stage in a diagnosis are searched to the same depth.[1] The limitations arising from this inflexibility will be discussed later.

The breadth of the search is controlled indirectly by the user through the use of a threshold probability. At a given decision node, only those tests which are relevant to a state with a probability greater than the threshold are considered by the test selection function. For example, if the probability distribution at a given decision node is (0.2, 0.3, 0.5) for states M_1, M_2, M_3 and the threshold is 0.25, only those tests relevant to states M_2 and to M_3 will be considered. Those tests which are relevant to M_1 alone will be ignored. A test is considered relevant to a particular state only if an attribute which is associated with the appropriate state list in the information structure is a possible result of the test given the probability distribution for the current decision node. Since the control of the breadth of search is indirect, in general, the user cannot easily predict the extent of the pruning of the decision tree which will result. Some feeling for reduction in the search space can be gained from experience with the program in a particular problem area. Note that in the above example, if all the tests which are relevant to

---

[1] An exception occurs when a particular node corresponds to a certain diagnosis. The search of the branch containing this node will terminate there.

state M_1 are also relevant to either M_2 or M_3, then the threshold probability will not result in any pruning of the decision tree. The maximum search breadth is obtained with a threshold of zero.

Like the search depth parameter, the threshold parameter can be set prior to each stage in the diagnosis. Also these two parameters can be varies independently of one another (subject only to a practical constraint of available storage). This flexibility permits the overall selection strategy to change during the course of the diagnosis.

There are four routines in the test selection package, each performing a distinct function in the tree search. The principal routine is SEQDEC which serves as the main control for the process of test selection. The diagnostic program communicates with the test selection package through SEQDEC. It provides this routine the name of the node in the decision tree which corresponds to the current state of the diagnosis. SEQDEC then analyzes the tree to the appropriate depth and breadth to obtain the testing decision.

Because the decision tree can require considerable storage even for limited search depth and breadth, the tree is developed dynamically. That is, new levels are added only as they are needed, and levels are erased when they have been analyzed. SEQDEC is called with the name of a decision node as an argument. This decision node is represented by an empty SLIP list which has on its DLIST a list containing a probability distribution over system states. This distribution incorporates all the attributes which were observed on the

path from the beginning of the tree to the current node.

SEQDEC first determines the expected loss for an optimal decision at this node. The manner in which this value is determined will be explained below. If the level of the current node equals the required depth of search this expected loss is returned as the expected loss for the node. If not, the current loss for this node is assigned this value and if the level of the node is the topmost level of the analysis, the terminal decision and its value are stored in a special list. In any event an additional level must be "grown" on the tree. First the routine RELTST is called by SEQDEC. RELTST determines the set of tests which are relevant to the states whose probability at the current node exceeds the threshold. Excluded from this set are all those tests which have been actually run. These latter tests are known to RELTST because whenever a test is selected by the diagnostic program and run by the user, its name is placed on a list called TSTRUN in common storage. RELTST stores the names of the relevant tests on the current decision node list.

After RELTST has collected the set of relevant tests, SEQDEC processes each of these tests in turn. SEQDEC begins reading the list of tests. For each test, a routine called GROW1 is invoked. This routine determines all possible results of the given test and their respective probabilities. For each result, the routine constructs a new decision node. First the current test is placed on the top of TSTRUN to simulate the running of the test and then for each

of the possible results of the test, SEQDEC calls itself recursively to obtain the expected loss of the resulting decision node. When this value has been obtained, it is weighted by the probability of the given result and the product accumulated. The sum of the expected loss for each result is combined with the cost of the test. The current test is removed from TSTRUN and the portion of the decision tree which has just been analyzed is erased. If the analysis is at the topmost level the value of the test is saved. This means that the expected losses for all alternatives at the current level are available. In the event that the best alternative cannot be employed (e.g. a test cannot be run for some reason), the next best alternative can be chosen. In any case, the expected loss for this test is compared with that of the best decision to date for the node. If it is less, the current test becomes the best decision. The analysis then proceeds to the next test alternative. When all alternatives have been evaluated for the current decision node, SEQDEC returns the expected loss of the best decision as determined by the analysis.

The determination of the optimal terminal decision as accomplished by a routine called DLOSS. This routine employs the probability distribution, the decision node and the loss function to determine the value of the minimum expected loss terminal decision for the node. If π_j is the probability of the state M_j in the current distribution and l_{ij} is a typical element from the loss function matrix, DLOSS selects state M_K where

$$\bar{E} = \sum_{j=1}^{n} l_{kj} \pi_j = \min_i \sum_{j=1}^{n} l_{ij} \pi_j$$

and $\bar{E}$ is the expected loss of the optimal terminal decision for the node. The state selected by DLOSS and the value $\bar{E}$ are returned to SEQDEC.

By controlling the breadth and the depth of the search employed by the test selection function, the user can generate a number of different test selection heuristics. For example, he might use a threshold close to zero and a depth of one early in a diagnosis when many states are still possible. Because the probability distribution based on only a few attributes may be quite diffuse, a low threshold is needed to insure that significant tests are not overlooked. On the other hand, the potentially large number of decision nodes requires a limited depth of search. As the diagnosis progresses and a few states become relatively probable, the threshold can be raised with less danger of missing significant tests. With the higher threshold it may be possible to improve the evaluation of tests by increasing the depth of the search.

The selection scheme above can be supplemented by the use of two additional features of the program. First, the user can restrict the set of relevant tests to those associated with the best terminal decision at a given node. In the case when the loss function is a constant for all ordered pairs of states, this corresponds to considering the tests which are relevant to the most probable state.

Since the routine DLOSS can determine the best terminal decision at a given decision, the appropriate state can be made available to RELTST. By considering only the tests relevant to this state, the user in a sense in limiting the search to those tests which will tend to prove or disprove the hypothesis that the given state is indeed the best decision. In practice, the user obtains this option by setting the threshold probability to a number greater than one.

In order to permit the user an even greater facility to test hypotheses, the program permits him to request a search for tests to prove or disprove the hypothesis that "the state of the system is M_k." If the user chooses to test such a hypothesis, the test selection function will alter its method of evaluating decision nodes. All decision losses ($l \neq j$) are set temporarily to a certain very high value. The routine DLOSS then considers only two states in its evaluation of the loss for a given node. One state is M_k and the other is "not M_k." With these adjustments, the test selection function will rank tests according to their expected value in proving or disproving the presence of state M_k.

A comparison of a number of particular selection heuristics employed in this research will be presented later in the thesis.

C. THE GENERATOR PROGRAM

The diagnostic program discussed in the previous sections is a major tool in this research. By exploiting the interactive capabilities of the program, the user can employ it directly in the solution

of actual diagnostic problems. Of equal importance, however, is the availability of the program as a test vehicle for a variety of overall diagnostic strategies. By specifying the heuristics to be employed in the pattern sorting and test selection functions, one is defining a diagnostic strategy. Since diagnostic problems tend to be difficult and the program operation is quite complicated, it is not an easy task to make generalizations about a given diagnostic strategy. There are many important questions which can be asked about a diagnostic strategy such as

- How is the performance of the program affected by noise signs?
- What is the effect of uncertainty in the probabilities on the performance of the program?
- How do various changes in the relevant probability distributions affect program performance?

Questions such as these are difficult to answer based on experience with only a few problem areas. If one is constrained to work with descriptions of actual systems, it may be very difficult to establish the conditions required for the test of a particular aspect of the program. If, on the other hand, one can employ a wide variety of system descriptions, the program can be exercised more thoroughly. One approach is to create an information structure with the desired properties and to test the diagnostic program with simulated problems from this artificial problem area. Information gained from such studies of diagnosis "in the abstract" may suggest improvements in the program.

It may also provide a deeper insight into the problems involved in solving real diagnostic problems. If such a simulation facility were available, simulated cases generated from the structure for an actual problem area could be utilized to conveniently investigate aspects of diagnosis in that area.

The diagnostic system includes such a simulation facility in the form of the generator program. This program is the third major part of the diagnostic system. Like the diagnostic program, the generator makes extensive use of the information structure. The system for which problems are to be simulated is described in the standard manner by the user. This description is converted to an information structure which is available to both the diagnostic program and the generator. The basic operation of the generator is as follows. First, a state is chosen at random from the set of possible states for the system in accordance with the a priori probability distribution. Then a certain number of initial attributes (the number being controlled by the user) are generated at random given the description of the state in the information structure. The set of initial attributes constitutes the problem presented to the diagnostic program. The latter is called to process these attributes. It selects a test in the usual manner. Given the state and the test, the generator selects a test result and conveys this response to the diagnostic program. This interaction between the generator and the diagnostic program continues until the latter arrives at a diagno-

sis. This diagnosis then can be compared with the "known" state used by the generator.

As an example of the operation of the generator, consider its use in the following simplified problem. The generator is used to simulate disease case histories for the disease-attribute probability matrix presented in Table III. The relevant tests are listed to the right of the matrix. Assume that cases are to be drawn at random from the structure and that one initial attribute is to be presented to the diagnostic program.

The generator first selects the disease. It does this by creating a list of all possible diseases and cummulative probabilities. For this example, the list would be

(D1 0.3 D2 1.0)

Each cummulative probability is the sum of the a priori probabilities of the diseases preceding it in the list. Then a random number between zero and one is generated. The list of diseases and cummulative probabilities, called the generation list, is searched for a disease with the property that the probability preceding it is less than and the probability following is greater than the given random number. This disease satisfying this condition is chosen for this case. Thus, if the random number generated in the example were 0.41, the disease selected would be D2. Assuming the disease D2 has been chosen, the generator now selects the initial attributes which define

TABLE 3

Disease Description for Generator Example

Disease	a priori Probability	P(Attribute/Disease) A1	A2	A3	A4	A5	A6
D1	0.3	0.3	0.7	0.5	1.0	0.5	0.5
D2	0.7	0.8	0.2	0.3	0.2	0.6	0.4

Test	Attributes
T1	A1, A2
T2	A3
T3	A4
T4	A5, A6

with the appropriate probabilities and returns it to the diagnostic program. This iterative process continues until the diagnostic program has completed the diagnosis.

In this example, only one attribute was generated for each test. There are tests, however, from which several attributes can be obtained. Such tests are marked in the information structure, and the generator will generate a set of test results for these tests.

The diagnostic system will record an extensive history of each diagnosis or selected aspects of that history on a history file if requested to do so by the user. A schematic of the relationships among the three major parts of the diagnostic system is presented in Figure 14. In the remainder of this section, certain features of the generator-diagnostic program interaction will be discussed in detail.

The subroutine GETSYM is the principal link between the generator and the diagnostic program. It is this routine which is called by the diagnostic program whenever the latter requires a test to be run. If the diagnostic program is being controlled by the user from the console, then GETSYM retrieves the test results from him. If the generator is in control, a routine called GENSYM is invoked to generate an appropriate response to the chosen test. The diagnostic program itself is independent of the source of responses to tests. GENSYM is also used by the generator to select the initial attributes of a problem. All system output (such as requests for test results, distributions, etc.) is processed by a special output package. This

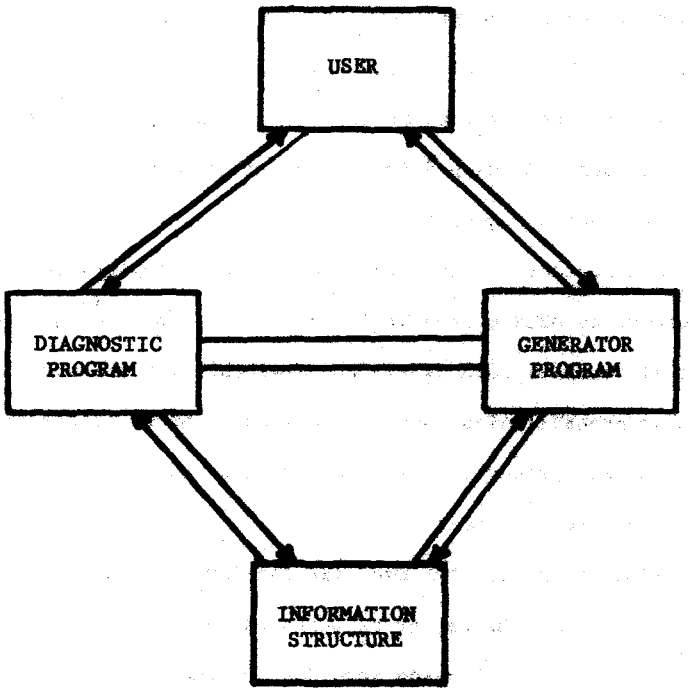
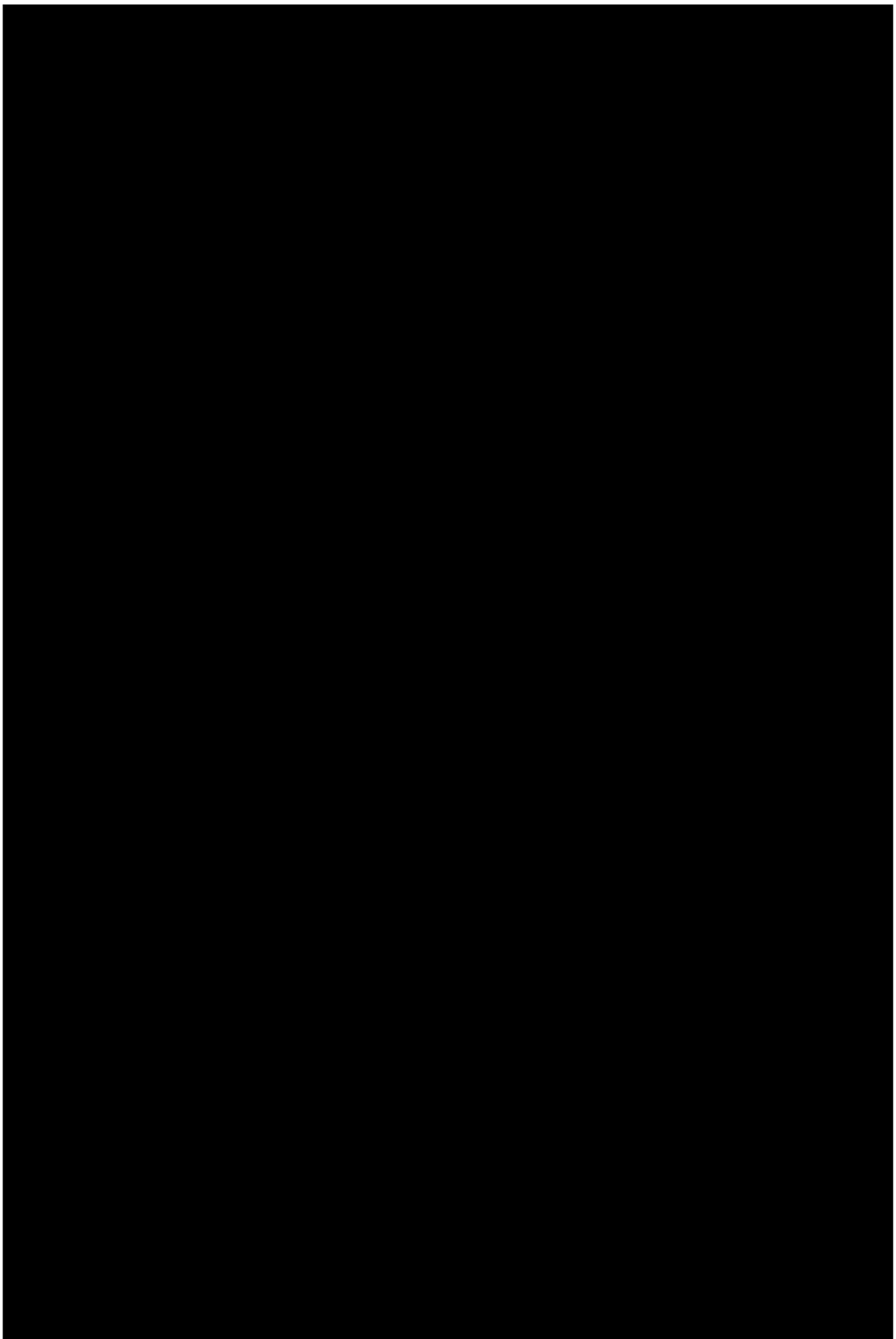
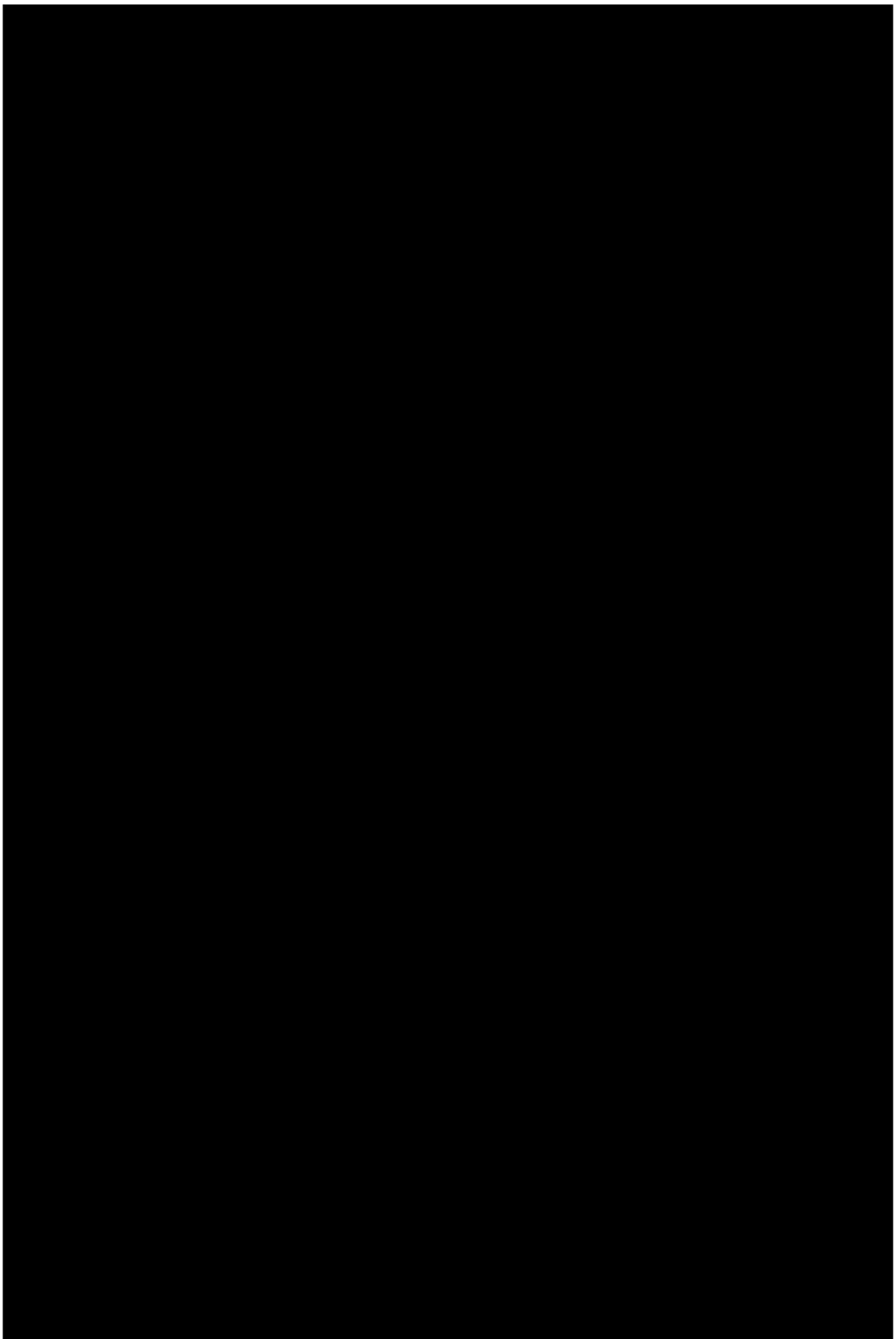


Figure 14

Schematic of Diagnostic System



another advantage is that many cases can be simulated in a reasonable amount of time.



The second problem encountered in the diagnosis of bone tumors is the large number of potentially useful attributes which can be extracted from a radiograph. Generally speaking, there are four direct kinds of information which are obtained from a radiograph of a bone tumor (R-20)

1) Destruction of bone
2) Proliferation of bone
3) Mineralization of tumor matrix
4) Location, size, and shape of tumor.

Each of these general classes of information is broken down into a number of more specific attributes. The result is the large number of attributes mentioned above. Hence, the diagnostician is confronted with a considerable amount of data which he may employ in classifying a particular tumor.

The particular study discussed here involved the diagnosis of actual cases of bone tumors, each of which was classified into one of nine histological types. These types are listed in Table 4. The evidence employed in the diagnoses consisted of fifty-three attributes obtained principally from radiographs. (The age of the patient was the only non-radiologic attribute considered.) The attributes are listed in Table 5 along with their abbreviations used in discussions of particular diagnoses.

The case histories and the disease-attribute probability matrix used in this study were obtained from Dr. G. S. Lodwick of the

University of Missouri. Dr. Lodwick and his associates developed the matrix as a result of many years experience with cases of bone tumors. Thus, the matrix represents the distillation of extensive diagnostic experience with the problem. It reflects both the statistical experience and understanding of the disease processes involved of the workers who created it. The papers cited above summarize their work and are recommended to any reader who is interested in a more authoritative view of the problem than the competence of this author permits him to present.

B. Experiments in Bone Tumor Diagnosis

The diagnostic system was used to study various aspects of bone tumor diagnoses. The disease-attribute probability matrix provided by Dr. Lodwick was used as the basis for an information structure for the system. A state was defined for each of the nine types of bone tumor. A set of thirty-two tests were defined. Some of these tests such as that of determining the age of the patient can result in one of a number of attributes. In the case of the age test, the possible attributes are: 1) age 0 to 9 years, 2) age 10 to 19 years, 3) age 20 to 29 years, 4) age 30 to 39 years, and 5) age 40 years and over. Other tests are specific for one attribute, such as the test of checking for geographic destruction of bone. The set of tests and the respective attributes which may result is presented in Table 6. Throughout the remainder of this

TABLE 4

HISTOLOGICAL TYPES FOR BONE TUMOR DIAGNOSIS

	Type	Abbreviation	Relative Incidence
1.	Chrondoblastoma	CB	0.05
2.	Chrondosarcoma	CS	0.17
3.	Ewing's Sarcoma	ES	0.15
4.	Fibrosarcoma	FS	0.10
5.	Giant Cell Tumor	GC	0.15
6.	Osteosarcoma	OS	0.25
7.	Parosteal Sarcoma	PS	0.05
8.	Reticulum Cell Sarcoma	RC	0.05
9.	Chrondomyzoid Fibroma	CF	0.03
			1.00

Note: This formulation assumes that each patient has one and only one of the given diseases.

TABLE 5

ATTRIBUTES FOR BONE TUMOR DIAGNOSIS

Attribute	Meaning
S02	Age 00-09 years
S03	Age 10-19 years
S04	Age 20-29 years
S05	Age 30-39 years
S06	Age 40 years and over
S07	Tumor Size 01-30 Millimeters
S08	Tumor Size 31-60 Millimeters
S09	Tumor Size 61-90 Millimeters
S10	Tumor Size 91 MM and over
S11	Shape-Round (L LT 1.5 X W)
S12	Shape-Elongated (L GE 1.5 X W)
S13	Location-Central
S14	Location-Eccentric
S15	Location-Cortex/Parosteal
S16	Long Bone
S17	Flat Bone
S18	Small Bone
S19	Sacrum and Pelvis
S20	Any Bone-Epiphysis
S21	Any Bone-Growth Plate
S22	Tubular Bone-Articular Cortex
S23	Tubular Bone-Metaphysis
S24	Tubular Bone-Shaft
S27	Matrix-Radiolucent
S28	Matrix-Floccules
S29	Matrix-Solid
S30	Matrix-Lumpt
S31	Matrix-Clouds
S32	Destruction-Geographic
S33	Destruction-Motheaten
S34	Destruction-Permeated
S35	Margin-Regular
S36	Margin-Lubulated
S37	Margin-Ragged
S38	Margin-Indistinct
S39	Transition Sharp or Smudged
S40	Invasive Zone
S41	Special Sign-Fracture
S42	Special Sign-Displacement
S43	Proliferation-Sclerotic Rim
S44	Prolif.-Multiple Small Foci
S45	Proliferation-Endostosis
S46	Periosteal-Hyperostosis
S47	Periosteal-Buttress
S48	Periosteal-Trabeculae (Septae)
S49	Cortex Expanded
S50	No Codman's Triangle
S51	One Codman's Triangle
S52	Two or More Codman's Triangles
S53	No periostosis
S54	Laminated Periostosis
S55	Amorphous Periostosis
S56	No Spiculation
S57	Sunburst Spiculation
S58	Hair-on-end Spiculation
S59	Velvet Spiculation
S60	Periosteal Response-Delicate
S61	Periosteal Response-Coarse

TABLE 6

TESTS FOR BONE TUMOR DIAGNOSIS

	Test	Possible Results
1.	TEST2	S02, S03, S04, S05, S06
2.	TEST7	S07, S08, S09, S10
3.	TEST11	S11, S12
4.	TEST13	S13, S14, S15
5.	TEST16	S16, S17, S18, S19
6.	TEST20	S20, N
7.	TEST21	S21, N
8.	TEST22	S22, N
9.	TEST23	S23, N
10.	TEST24	S24, N
11.	TEST27	S27, N
12.	TEST28	S28, N
13.	TEST29	S29, N
14.	TEST20	S30, N
15.	TEST31	S31, N
16.	TEST32	S32, N
17.	TEST33	S33, N
18.	TEST34	S34, N
19.	TEST35	S35, S36, S37, S38
20.	TEST39	S39, S40, N
21.	TEST41	S41, S42, N
22.	TEST43	S43, N
23.	TEST44	S44, N
24.	TEST45	S45, N
25.	TEST46	S46, N
26.	TEST47	S47, N
27.	TEST48	S48, N
28.	TEST49	S49, N
29.	TEST50	S50, S51, S52
30.	TEST53	S53, S54, S55
31.	TEST56	S56, S57, S58, S59
32.	TEST60	S60, S61

Note: The symbol "N" denotes a "normal" attribute. It means that a test may fail to reveal any of the other attributes listed. Thus, for TEST41, the possible results are S41 or S42 or neither S41 nor S47 (N).

chapter, the abbreviations for diseases and attributes presented in Table 5 and Table 6 will be used. In the initial set of experiments, all tests were assigned unit cost and the cost of all misdiagnoses (e.g. deciding the tumor is CS when it is really GC) was assumed to be 100,000. This number is quite arbitrary, and is used simply to make the decision losses much greater than the testing losses.

Experiment 1. Diagnosis Based on All Attributes

Each of the twelve case histories was presented to the diagnostic program by inputting all the attributes for the case. The diagnostic program processed the attributes through the inference function and obtained a posterior distribution for the type of tumor. The results of this experiment are presented in Table 7 along with the diagnosis of a pathologist provided with each case history. The latter is traditionally accepted as the definitive diagnosis in cases of this type.

Experiment 2. Sequential Diagnoses--Actual Case Histories

The second experiment exercised the sequential capabilities of the diagnostic program. Again, all diseases were taken to be equally serious (l_{ij} = 100,000, $i \neq j$) and all tests were assigned unit cost. The same twelve cases were analyzed by the program. For each case, the program was presented with a set of initial attributes. This set was obtained by collecting the results of the

TABLE 7

Diagnoses Based on all Available

Attributes for Actual Bone Tumor Case Histories

Case	Posterior Distribution*	Pathology
1	CB 0.12 GC 0.87	GC
2	OS 0.65 CS 0.35	OS
3	CB 1.00	CB
4	CS 0.99	CS
5	OS 1.00	OS
6	ES .33 RC .67	RC
7	CS 0.78 FS 0.22	CS
8	ES 0.04 ES 0.02 RC 0.94	ES
9	ES 1.00	ES
10	CS 1.00	CB
11	GC 0.65 CF 0.35	GC
12	PS 0.99	PS

* Only types with posterior probability greater than or equal to 0.01 are shown in the tables in this chapter.

first ten tests listed in Table 6 from the case histories. Thus each diagnostic problem was defined by approximately ten attributes. (In certain cases this number was smaller, because some tests are not relevant to specific bones.)

After processing the initial attributes, for the case, the program employed the test selection function to select a test to be run. The results of the test selected were determined by consulting the given case history. The attribute or attributes resulting from this test were given to the program and the inference-test selection cycle repeated. Throughout this experiment the test selection function searched the decision tree to a depth of one and limited the breadth of search to those tests relevant to the most likely disease type.

For each case, this sequential diagnosis was continued until the diagnostic program terminated the process. This termination occurred when the program determined the expected reduction in loss for the best test at the current decision node was less than the cost of the test.

An example of a sequential diagnosis is presented in Table 8 and the results of the experiment are summarized in Table 9.

The results of Experiment 2 underscore the potential advantage of sequential analysis of attributes in diagnosis. Since all diseases were taken to be equally serious for this experiment, the program found the best terminal decision to be the most probable disease. Since these same conditions held in Experiment 1, it is easy to make comparisons between the results of the two experiments.

TABLE 8

Sequential Diagnosis--An Example

(Actual Case History 12)

	Test	Resulting Attributes	Posterior Distribution
1.	--	S05, S10, S12, S15 S16, NOT S20, NOT S21 NOT S22, S23, S24	CS 0.42 ES 0.13 FS 0.10 PS 0.31 RC 0.02
2.	TEST29	S29	CS 0.06 FS 0.02 OS 0.01 PS 0.91
3.	TEST50	S50	CS 0.06 FS 0.01 PS 0.92
4.	TEST56	S56	CS 0.05 FS 0.02 PS 0.93

Terminal decision -- PS
Pathology report -- PS

TABLE 9

Sequential Diagnosis of Bone Tumor Cases
Summary of Results for Actual Case Histories

Case and Pathology	Number of Tests Selected by Program	Distribution at Point of Terminal Decision	Distribution When all Attributes Considered
1. (GC)	9	CB 0.21 GC 0.78	CB 0.12 GC 0.87
2. (OS)	12	CS 0.79 OS 0.21	CS 0.65 OS 0.35
3. (CB)	0	CB 1.00	CB 1.00
4. (CS)	4	CS 0.80 ES 0.08 FS 0.08 OS 0.04	CS 0.99
5. (OS)	4	CS 0.03 ES 0.02 OS 0.94 RC 0.03	OS 1.00
6. (RC)	13	ES 0.30 FS 0.01 RC 0.68	ES 0.33 RC 0.67
7. (CS)	4	CS 0.74 FS 0.26	CS 0.78 FS 0.22
8. (ES)	11	ES 0.05 FS 0.07 RC 0.87	ES 0.04 FS 0.02 RC 0.94
9. (ES)	5	CS 0.02 ES 0.88 OS 0.05 RC 0.05	ES 1.00
10. (CB)	3	CB 0.96 CF 0.04	CB 1.00
11. (GC)	5	CS 0.10 ES 0.01 GC 0.81 CF 0.08	GC 0.65 CF 0.35

12.	(PS)	3	CS 0.05 FS 0.02 PS 0.93	PS 0.99	

Average number of initial attributes 9.4
Average number of test by program 7.1

With regard to "accuracy," it can be seen that the lists of terminal decisions from the two experiments are identical and these decisions are the same as those of the pathologist in ten of the twelve cases. The major difference between the two sets of results is the average number of tests performed per diagnosis. In the first case this average is 30. (The average is less than 32 because some test results were not available or were not relevant for a given case and the test was not counted.) Sequential analysis of the given cases required an average of 16.7 tests per case. This average includes 9.4 tests on the average to obtain the initial attributes. Thus, by employing sequential analysis, the program in each case obtained the same diagnostic decision as it obtained using all attributes, but with only slightly more than half as many tests.

The nature of diagnosis of bone tumors makes this saving seem immaterial. That is, almost all attributes are obtained from a radiograph, and once the radiograph has been obtained, the marginal cost of the tests considered here is essentially zero. One can easily imagine a situation, however, in which tests are completely independent of one another. In such a situation, the savings from sequential diagnosis might be quite significant. The fact that the performance of a diagnostician should be assessed in terms of both accuracy and cost favors the sequential mode of operation for the program. The question of how to assess the performance of a diagnostician will be considered at greater length later.

Another difference between the results of the two experiments is found in the posterior distributions at the points of a terminal decision. The average value of the maximum likelihood probability for the terminal decisions can be taken as an indication of the equivocation or uncertainty in the average decision. For Experiment 1 this value is 0.85 while for Experiment 2, it is 0.80. Therefore, the sequential diagnoses terminate on slightly less "certain" decisions.

Experiment 3. Sequential Analysis--Simulated Case Histories

Table 10 presents the results of the sequential diagnoses of ten simulated case histories. The generator function was used to develop the cases and the diagnostic program employed as usual. Again, all diseases were taken to be equally serious and all tests were assigned unit cost.

Again, the marked advantage of sequential diagnosis is evident. The average number of tests required for diagnosis was 17.0. Based on a maximum likelihood terminal decision, the diagnostic program's terminal decision was correct in nine of ten cases.

On the average, the diagnostic program was more certain of its terminal decisions than in the previous experiments (average probability of terminal decision = 90.5).

TABLE 10

Sequential Diagnosis of Simulated Case Histories

Histological Type	Number of Initial Attributes	Number of Tests Selected by Program	Distribution at Point of Terminal Decision
1. FS	14	9	CS 0.26 FS 0.73
2. ES	7	9	ES 0.88 OS 0.01 RC 0.11
3. OS	11	0	OS 1.00
4. GC	5	11	CS 0.01 GC 0.79 CF 0.20
5. ES	12	6	CS 0.01 ES 0.94 OS 0.04
6. RC	5	8	CS 0.05 FS 0.78 RC 0.16
7. CB	11	8	CB 0.93 GC 0.02 CF 0.05
8. OS	11	8	OS 0.98 CS 0.02
9. FS	5	12	CS 0.11 FS 0.88
10. GC	10	8	CB 0.04 FS 0.01 GC 0.94 CF 0.01
	Average number of initial attributes 9.1	Average number of tests by program 7.9	

Chapter 6

DIAGNOSIS OF CONGENITAL HEART DISEASE

A. The Nature of the Diagnostic Problem

A prolonged study of a group of thirty-four types of congenital heart disease has been conducted by Warner and his associates (R12, R13, R14). As a result of this study, they developed a disease-attribute probability matrix for thirty-five types (including "normal") and fifty-seven attributes. The attributes can be grouped into four main categories: murmurs, electrocardiogram findings, X-ray findings, and other symptoms and physical signs. The problem of diagnosing heart disease cases based on this matrix is more difficult than the bone tumor problem discussed in Chapter 5. One reason for the increased difficulty is simply the increased number of diseases. Also certain groups of diseases have quite similar attribute probabilities in the matrix.

As noted in Chapter 2, Warner developed a computer program to perform diagnosis of congenital heart disease patients based on a Bayesian analysis of their signs and symptoms. His program employs the matrix mentioned above, but in addition it must account for certain dependencies (such as mutual exclusion of signs or symptoms). From the performance measures presented in Chapter 2, it can be seen that Warner's program performs at the level of an experienced physician.

The experiments discussed here involved the use of the disease-attribute probability matrix prepared by Warner in the diagnosis of congenital heart disease. As before, the matrix was the basis for each of the disease types and the appropriate attribute lists created. Twenty-eight tests were also defined for the problem. Dr. Warner provided nine case histories, each with the correct diagnosis and the diagnosis obtained by his program. In this instance, the correct diagnoses were determined by follow-up studies such as heart catheterization or autopsy.

Table 11 presents the names of the thirty-five states of the information structure used in these experiments and the names of the corresponding diseases. Table 12 lists the attributes of the problem; and Table 13 the tests.

B. Experiments in Congenital Heart Disease Diagnosis

Experiment 4. Diagnosis Based on All Attributes

The first experiment tested the diagnostic capability of the program given all the known attributes for each of the actual case histories provided by Dr. Warner. The results of this experiment are summarized in Table 14. In each instance, the diagnostic program duplicated the results obtained by Warner's program for the given case history. (That is, both programs arrived at the same posterior probability distribution given all attributes.)

TABLE 11

Heart Disease Types

States	Diseases	States	Diseases
D01	Normal	D18	Patent ductus arteriosus
D02	Atrial septal defect	D19	Pulmonary arterio-venous Fistula
D03	Atrial septal defect with pulmonary stenosis	D20	Congenital metral disease
D04	Atrial septal defect with pulmonary hypertension	D21	Primary myocardial disease
D05	Atrio-ventricular communis	D22	Anomalous origin or coronary artery
D06	Partial anomalous pulmonary venous connection	D23	Congenital aortic disease
D07	Total anomalous pulmonary venous connection	D24	Ventricular septal defect with pulmonary flow $\leq$ 1.4 systemic flow
D08	Tricuspid atresia (without transposition)	D25	Coarctation of aorta
D09	Ebstein's anomaly	D26	Truncus arteriosus
D10	Ventricular septal defect with valvular pulmonary stenosis	D27	Transposition
D11	Ventricular septal defect with infundibular pulmonary stenosis	D28	Hypertrophic subaortic stenosis
D12	Pulmonary stenosis, valvular, gradient $\geq$ 40 mm. Hg.	D29	Absent aortic arch
D13	Pulmonary stenosis, infundibular, gradient $\geq$ 40 mm. Hg.	D30	Ventricular septal defect with pulmonary flow $>$ 1.4 systemic flow
D14	Pulmonary atresia	D31	Ventricular septal defect with pulmonary hypertension
D15	Peripheral pulmonary stenosis	D32	Patent ductus arteriosus with pulmonary hypertension
D16	Pulmonary hypertension	D33	Tricuspid atresia with transplantation
D17	Aortic pulmonary window	D34	Pulmonary stenosis gradient $<$ 40 mm. Gh.
		D35	Ruptured sinus Valsalva

TABLE 12

Attributes for Congenital Heart Disease

Sign	Meaning	Sign	Meaning
S01	Age, less than 1 year	S29	Post systolic
S02	Age, 1 year to 20 years	S30	Post continuous
S03	Age, 20 or more years	S31	Murmur louder than gr 3/6 (10 mm)
S04	Cyanosis, mild	S35	Accentuated P_2
S05	Cyanosis, severe (with clubbing	S36	Diminished P_2
S06	Cyanosis intermittent	S37	Fixed split P_2
S07	Cyanosis differential	S38	Femoral pulse less than brachial
S08	Squatting	S40	Atrial fibrillation or broad notched P wave
S09	Apex systolic	S41	Axis, right (more than 110°)
S10	Apex systolic, holo	S42	Axis, left (less than 0°)
S11	Apex systolic, mid	S43	R wave greater than 1.2 mv in lead V_1
S12	Apex diastolic		
S13	Apex diastolic, early	S44	rR' or qR in lead V_1
S14	Apex diastolic, late	S45	R wave greater than 2.5 mv in lead V_6
S15	L 4th systolic		
S16	L 4th systolic, holo	S46	T wave inversion in lead V_6
S17	L 4th systolic, mid	S47	Rib notching
S18	L 4th continuous	S48	Peripheral vessels increased
S19	L 4th diastolic	S49	Peripheral vessels decreased
S20	L 4th diastolic, holo	S50	Hilar vessels increased
S21	L 4th diastolic, early	S51	Hilar vessels decreased
S22	L 2nd systolic	S52	Main pulmonary artery large
S23	L 2nd systolic, holo	S53	Main pulmonary artery not seen
S24	L 2nd systolic, mid	S54	Aorta large
S25	L 2nd continuous	S55	Aorta small
S27	R 2nd systolic	S56	Cardiomegaly
S28	R 2nd diastolic	S57	Snowman

TABLE 13

Tests for Heart Disease Diagnosis

Tests	Possible Results
1. TEST1	S01, S02, S03
2. TEST4	S04, S05, S06, S07, N
3. TEST8	S08, N
4. TEST9	S09, N
5. TEST10	S10, S11, N
6. TEST12	S12, N
7. TEST13	S13, S14, N
8. TEST15	S15, N
9. TEST16	S16, S17, S18, N
10. TEST19	S19, N
11. TEST20	S20, S21, N
12. TEST22	S22, N
13. TEST23	S23, S24, S25, N
14. TEST27	S27, N
15. TEST28	S28, N
16. TEST29	S29, S30, N
17. TEST31	S31, N
18. TEST35	S35, S36, N
19. TEST37	S36, S37, N
20. TEST38	S38, N
21. TEST40	S40, N
22. TEST41	S41, S42, N
23. TEST43	S43, N
24. TEST44	S44, N
25. TEST45	S45, N
26. TEST46	S46, N
27. TEST47	S47, N
28. TEST48	S48, S49, N
29. TEST50	S50, S51, N
30. TEST52	S52, S54, N
31. TEST54	S54, S55, N
32. TEST56	S56, N
33. TEST57	S57, N

TABLE 14

Diagnoses Based on All Available Attributes
for Actual Heart Disease Case Histories

Case	Posterior Distribution*	Definitive Diagnosis
1	DO3 0.91 NORMAL 0.04 D34 0.03	DO9
2	DO5 0.84 DO2 0.09 D31 0.03 DO4 0.03	DO4
3	D32 1.00	DO2
4	D20 0.41 D28 0.38 NORMAL 0.22 D24 0.04 D34 0.02 D11 0.01	NORMAL
5	DO8 0.94 D33 0.05	D33
6	D32 0.98 D29 0.02	D32
7	D31 0.47 D30 0.37 DO5 0.08 DO2 0.03 D32 0.02	D31
8	D30 0.87 DO2 0.12	D30
9	D31 0.70 D27 0.20 D26 0.10	D27

* Only diseases with probability greater than or equal to 0.01 are shown.

Experiment 5. Sequential Diagnosis of Heart Disease Cases

The actual heart disease cases were also diagnosed by the program using the sequential mode of operation. In each case, the initial attributes presented to the program were the results from a set of seven tests relating to physical signs. The diseases were assumed to be equally serious ($l_{ij} = 100{,}000$, $i \neq j$) and all tests were assigned unit cost. The search depth in the test selection function was one in each case.

A summary of the results of this experiment is presented in Table 15. Again, the advantage of sequential diagnosis is apparent. The program required an average of 5.8 tests to obtain a diagnosis compared to the thirty-three tests required to determine all attributes. This small number of tests is interesting. Recall the sequential diagnosis of the bone tumor cases required an average of 6.7 tests per case, although the problem involves only one quarter as many states as the heart disease problem. Several reasons might be advanced to account for this. First, the tests associated with heart disease may include a number which have little value in differentiating groups of diseases. Thus, in a given problem, the test selection function may choose a terminal decision after relatively few tests have been run. A second reason may be the relevance of more inter-attribute relationships in the heart disease problem. Such relationships may be quite useful in diagnosis, but the testing sequences for them are not examined since the

TABLE 15

Sequential Diagnosis of Actual Heart Disease Cases

Case and Definitive Diagnosis	Number of Tests Selected by Program	Distribution at Terminal Decision	Distribution Based on all Attributes
1. DO9	10	NORMAL 0.04	NORMAL 0.04
		DO2 0.06	DO3 0.91
		DO3 0.69	D34 0.03
		D11 0.02	
		D18 0.05	
		D26 0.03	
		D34 0.03	
2. DO4	4	DO2 0.08	DO2 0.09
		DO4 0.17	DO4 0.03
		DO5 0.62	DO5 0.83
		D31 0.10	D31 0.03
3. DO2	1	D27 0.03	D32 1.00
		D32 0.96	
4. NORMAL	10	NORMAL 0.07	NORMAL 0.22
		D10 0.03	D28 0.38
		D11 0.07	D24 0.04
		D12 0.02	D20 0.41
		D20 0.67	D34 0.02
		D24 0.01	D11 0.01
		D28 0.10	
5. D33	3	DO8 0.92	DO8 0.94
		D33 0.01	D33 0.05
6. D32	0	D32 0.98	D32 0.98
		D29 0.01	D29 0.02
7. D31	10	DO4 0.01	D31 0.47
		DO5 0.09	D30 0.37
		D31 0.86	DO5 0.08
		D32 0.02	D32 0.02
8. D30	8	DO2 0.03	D30 0.87
		DO5 0.02	DO2 0.12
		D20 0.01	
		D30 0.89	

9. D27	6	D11	0.02	D31	0.70
		D19	0.01	D27	0.20
		D24	0.06	D26	0.10
		D26	0.06		
		D31	0.77		
		D33	0.03		

Average number of initial attributes = 7
Average number of tests by program = 5.8

depth of the tree search is limited to one level. Unfortunately, an increase in the depth of search leads to prohibitive amounts of computation in the heart disease problem. A deeper search may be possible if more powerful breadth-limiting heuristics are developed.

On the whole, the performance of the program with sequential diagnosis is comparable to that when all attributes are available. The one apparent exception to this involves case 9. Here the sequential diagnosis failed to assign a probability of greater than 0.01 to disease D27. The seriousness of this failure depends on medical considerations which are not discussed here. The general problem of measuring diagnostic performance, however, will be discussed in Chapter 8.

Chapter 7

FURTHER EXPERIMENTS WITH THE DIAGNOSTIC SYSTEM

In order to explore the potential value of the diagnostic system as a tool for the study of a variety of diagnostic problems and strategies, some further experiments were performed. The results of these experiments are reported in this chapter.

Experiment 6. The Effect of a Very Serious State

In the experiments discussed in Chapters 5 and 6, it was assumed that the loss for misdiagnosis was the same for all pairs of diseases. For each experiment, the elements of the loss function matrix were taken to be 0 for l_{ii} and 100,000 for l_{ij}, $1 \neq j$. For this reason, the diagnostic program always selected the most likely disease as its terminal decision. One can easily imagine situations, however, in which the assumption of a constant loss for misdiagnosis independent of the actual disease is unrealistic. For example, it may be far more serious to diagnose pneumonia as a common cold than *vice versa*. Since the diagnostic program incorporates such considerations in its rules for selecting a terminal decision, changes in the loss function matrix can result in pronounced changes in its decisions.

This effect was observed in two different situations. In the first, the loss function matrix is presented in Table 16. Note that it is very costly to miss the diagnosis of CB. The misdiagnosis of either CS or ES as a disease other than one of these two or CB is quite serious, but it is not particularly serious to diagnose CS as ES or CB or ES as CS or CB. Failure to diagnose one of the remaining diseases results in a loss which is independent of the diagnosis made.

The generator was used to generate seven case histories of bone tumor cases. Each case was diagnosed by the diagnostic program in the light of the new loss function. The results of this experiment are summarized in Table 17. From this table, it can be seen that the new loss function affects only one decision, that of case 3. In this case, the diagnostic program selected CB as the terminal decision in spite of the fact that GC (the actual disease) was more than three times as probable. The loss for diagnosing CB as GC is 1,000 times that of diagnosing GC as CB, however, and this fact dominates the decision of the program. The relative seriousness of CB does not affect the diagnoses of the remaining cases because the observed attributes excluded CB as a possibility in each case.

The effect of a serious disease on diagnosis can be made even more pronounced if the serious disease is not easily distinguished from other less serious ones. For example, the disease CS often

TABLE 16

Loss Function Matrix for Bone Tumor Diagnosis
(in thousands)

	Actual Disease								
Diagnosis	CB	CS	ES	FS	GC	OS	PS	RC	CF
CB	0	0.1	0.1	1	1	1	1	1	1
CS	100	0	0.1	1	1	1	1	1	1
ES	100	0.1	0	1	1	1	1	1	1
FS	100	10	10	0	1	1	1	1	1
GC	100	10	10	1	0	1	1	1	1
OS	100	10	10	1	1	0	1	1	1
PS	100	10	10	1	1	1	0	1	1
RC	100	10	10	1	1	1	1	0	1
CF	100	10	10	1	1	1	1	1	0

TABLE 17

Sequential Diagnosis of Cases for Loss Function of Table 16

Case and Disease	Number of Initial Attributes	Number of Tests Selected by Program	Distribution at Terminal Decision
1. (PS)	15	1	PS* 1.00
2. (GC)	8	7	GC* 0.90 FS 0.09 CS 0.01
3. (GC)	9	3	CB* 0.24 GC 0.76
4. (ES)	10	0	ES* 0.99 RC 0.01
5. (ES)	8	2	ES* 0.96 CS 0.02
6. (OS)	13	0	OS* 1.00
7. (GC)	8	12	GC* 0.89 FS 0.09 CS 0.02

* Terminal decision by program.

appears in a terminal distribution when the actual disease is another. This means that CS has not been excluded as a possible diagnosis when a terminal decision is made. By making CS very serious relative to the other diseases, the decisions of the program can be strongly influenced.

The loss function matrix presented in Table 18 represents just this situation. A series of simulated cases was diagnosed by the program using this loss function. The results of this experiment are summarized in Table 19. Here the seriousness of CS dominates all decisions, and the terminal decision is CS in all cases. Note also that the terminal decision is made after relatively few tests have been run and while the posterior distribution is relatively diffuse. The predominance of terminal decisions for disease CS is a result of the seriousness of that disease. The decrease in the number of tests per case and the diffuse terminal distributions reflect the difficulty finding a single test which promises to significantly alter the expected loss. Since the diagnostic program employed a one level look ahead in searching the decision tree for these cases, it did not consider possible sequences of several tests to resolve this problem. This point will be discussed in more detail later in the thesis.

The above example is but one in which the loss function has a significant effect on the terminal decisions made by the diagnostic program. Because the test selection strategy also accounts for the loss function, it, too, is affected by changes in the matrix. There-

TABLE 18

Loss Function Matrix for Bone Tumor Diagnosis
(in thousands)

	Actual Disease								
Diagnosis	CB	CS	ES	FS	GC	OS	PS	RC	CF
CB	0	1	1	1	1	1	1	1	1
CS	100	0	1	1	1	1	1	1	1
ES	100	1	0	1	1	1	1	1	1
FS	100	1	1	0	1	1	1	1	1
GC	100	1	1	1	0	1	1	1	1
OS	100	1	1	1	1	0	1	1	1
PS	100	1	1	1	1	1	0	1	1
RC	100	1	1	1	1	1	1	0	1
CF	100	1	1	1	1	1	1	1	0

TABLE 19

Sequential Diagnoses of Cases
for Loss Function of Table 18

Case and Disease	Number of Initial Attributes	Number of Tests Selected by Program	Distribution at Terminal Decision
1. (FS)	14	1	CS* 0.56 ES 0.02 FS 0.34 CF 0.07
2. (CS)	8	2	CS* 0.96 FS 0.02 ES 0.02
3. (CS)	8	4	CS* 0.11 FS 0.55 GC 0.03 OS 0.02 RC 0.30
4. (OS)	7	3	CS* 0.08 OS 0.91
5. (CB)	6	2	CS* 0.16 CB 0.21 ES 0.03 FS 0.11 GC 0.48
6. (GC)	7	2	CS* 0.15 CB 0.04 FS 0.07 GC 0.53 OS 0.12 PS 0.06
7. (OS)	15	5	CS* 0.01 OS 0.88 FS 0.06

* Terminal decision by program.

fore, an important facility in the study of diagnostic strategies for a particular application is the ability to assess the sensitivity of these strategies to the loss function. Although the current version of the diagnostic system restricts the loss function to a matrix form, it is still possible to employ wide ranges of the values of the matrix elements in a given application study. This facility coupled with the capabilities of the generator makes it possible to study the performance of different versions of the diagnostic program with a variety of matrix loss function.

Experiment 8. Studies of a Test-Selection Heuristic

The experiments discussed in Chapters 5 and 6 indicate the value of sequential diagnosis in reducing the number of tests required for a diagnosis. Therefore, it is worth some effort to improve the operation of the test-selection function.

One problem which can arise in the use of the test-selection function of the current system is the appreciable amounts of computation required to evaluate all the relevant tests at a given decision node. It would be quite desirable to reduce the amount of computation devoted to test selection provided that the diagnostic capability of the program were not impaired. As an example of the amount of computation involved in test selection, consider the following. In the diagnosis of congenital heart disease, there can be as many as thirty-five states with non-zero probabilities in the current distribution. If there are twenty relevant tests at a given

decision node, each with two possible results, a one-level evaluation of these tests could require the creation of forty distributions, each requiring the computation of thirty-five updated probabilities. This is a significant amount of processing for a highly interactive program, and the example cited does not represent a particularly large set of alternatives. Since the test-selection function may be performed many times during a diagnosis, there is a good reason to reduce the time required to perform it. An obvious approach is to improve the efficiency of the code for the function. While this would no doubt lead to improvements, it was not attempted. Attention was focused on attempting to reduce the number of tests considered, rather than reducing the time devoted to the evaluation of an individual test.

This approach was motivated by the results of the experiments with sequential diagnosis. There it was observed that relatively few tests were required for diagnosis by the program. The particular set of tests employed for a given diagnosis is determined dynamically by the program, and varies from one diagnosis to another. If one could guess which tests would be relevant to a particular diagnosis, the total number of tests considered could be reduced significantly. A guess about the relevance of certain tests must not be irreversible, however, because the value of some tests will become apparent only after other tests have been run.

At any stage in a diagnosis, the current distribution provides

the most logical basis for a hypothesis about the future relevance of particular tests. One heuristic which incorporates this view is the one which restricts the set of tests considered to those which are relevant to the state which is the best terminal decision at the current node. This heuristic favors those tests which tend to confirm or disprove the current "best guess" about the problem. It also had the property of reversibility mentioned above. When the terminal decision changes, the set of relevant tests changes correspondingly.

This heuristic was employed in a number of experiments with both congenital heart disease problems and bone tumor problems. In the cases studied it resulted in the same number of tests selected as the standard function which employs no such heuristic. This heuristic does reduce the average number of decision nodes considered per diagnosis. This reduction is not great, however, because in both problem areas the diseases share many attributes in common, and hence many relevant tests. Thus, at any decision node, almost almost all the tests are relevant to the state determined to be the best terminal decision.

A second heuristic which offered a potentially greater reduction in the number of decision nodes considered per diagnosis was also considered. This heuristic employs the current distribution to "guess" which tests will <u>not</u> be useful in the remainder of the diagnosis. Tests which are thought to have little value are temporarily removed

from consideration. At a later point in the diagnosis these tests may be released for further consideration.

The actual operation of this heuristic is as follows. At a given decision node, the set of relevant tests is evaluated by the test selection function. Then the set of tests is partitioned into two disjoint subsets. In the first are all those tests with the property that the sum of the cost of the test plus the expected loss of a terminal decision after the test has been run exceeds the expected loss of the current terminal decision. These tests are said to be dominated. The second set consists of all the remaining undominated tests. The heuristic hypothesizes that the tests in the dominated set will remain dominated for the remainder of the diagnosis. This set of tests is placed on the top of a push-down stack. At each decision node the push-down stack is examined prior to evaluating each test. If the test is found in the stack it is not considered at the decision node.

In general, then, each iteration of the test selection function produces a new set of dominated tests which are pushed onto the stack. This means the set of relevant tests is generally decreased at each stage of the diagnosis. Whenever there are no undominated tests at a given decision node (i.e. whenever the terminal decision is selected), the program releases the set of dominated tests (if one exists) on the bottom of the stack. This corresponds to re-evaluating those tests which were tentatively discarded earliest in the diagnosis.

The reason for this choice is that it is desirable to reconsider tests which were dominated when the distribution was quite different from the present one. If the distribution has changed little, tests which were formerly dominated are apt to be currently dominated. Actually, there is no guarantee that this method will produce the desired effect. It is used primarily as an example of a possible approach, and additional discussion will be devoted to the subject below.

The "dominated-test" heuristic was tested in the sequential diagnosis of both the heart cases and bone tumor cases. The nine heart disease cases and the twelve bone tumor cases were used as the testing sample. The same initial attributes for a given case were given to both the "dominated-test" heuristic and the standard version of the diagnostic program. The number of tests by the program, the number of decision nodes considered during diagnosis, and the distribution at the terminal decision were all recorded. These results are summarized in Tables 20 through 23. A number of these results have an interesting interpretation.

In both the heart disease cases and the bone tumor cases, the dominated-test heuristic results in a substantial reduction in the average number of decision nodes considered per diagnosis. In the heart disease problem, this heuristic results in a larger average number of tests performed per diagnosis. In situations when the cost of an average test exceeds the value of the computation saved, this is an undesirable effect. The reason for this reduction in diagnostic efficiency can be seen from the following interpretation of

TABLE 20

Sequential Diagnosis of Heart Disease Cases--
Standard Test Selection Function

Case and Diagnosis	Initial Attributes	Number of Tests Selected by Program	Number of Decision Nodes Considered	Distribution at Terminal Decision
1. DO9	7	10	541	NORMAL 0.04
				DO3 0.69
				D34 0.03
				DO2 0.06
				D18 0.05
				D26 0.03
2. DO4	7	4	287	DO2 0.08
				DO4 0.17
				DO5 0.62
				D31 0.10
3. DO2	7	1	133	D27 0.03
				D32 0.96
4. NORMAL	7	10	523	NORMAL 0.07
				F10 0.03
				D11 0.07
				D12 0.02
				D20 0.67
				D24 0.01
				D28 0.10
5. D33	7	3	248	DO8 0.92
				D33 0.01
6. D32	7	0	66	D32 0.98
				D29 0.01
7. D31	7	10	513	DO4 0.01
				DO5 0.09
				D31 0.86
				D32 0.02

8. D30	7	8	457	D02	0.03
				D05	0.02
				D20	0.01
				D30	0.89
9. D27	7	6	379	D11	0.02
				D19	0.01
				D24	0.06
				D26	0.06
				D31	0.77
				D33	0.03

Average number of tests by program = 5.8
Average number of decision nodes considered = 350

TABLE 21

Sequential Diagnosis of Heart Disease Cases
Dominated Test Heuristic

Case and Diagnosis	Initial Attributes	Number of Tests Selected by Program	Number of Decision Nodes Considered	Distribution at Terminal Decision
1. DO9	7	11	283	NORMAL 0.04 DO3 0.70 DO5 0.01 DO2 0.06 D11 0.02 D18 0.05 D26 0.03 D34 0.03
2. DO4	7	5	163	DO2 0.08 DO4 0.16 DO5 0.63 D31 0.03
3. DO2	7	1	66	D27 0.03 D32 0.96
4. NORMAL	7	16	345	NORMAL 0.50 D11 0.02 D15 0.02 D20 0.24 D28 0.02
5. D33	7	3	176	DO8 0.98 D33 0.01
6. D32	7	0	66	D32 0.98 D29 0.01
7. D31	7	11	269	DO4 0.06 DO5 0.14 D30 0.01 D31 0.71 D32 0.07

8. D30	7	10	301	D02	0.04
				D04	0.02
				D05	0.02
				D18	0.03
				D20	0.04
				D30	0.70
				D31	0.08
				D32	0.02
9. D27	7	6	216	D11	0.02
				D31	0.77
				D24	0.06
				D26	0.06
				D33	0.03

Average number of tests by program = 7
Average number of decision nodes considered = 208

TABLE 22

Sequential Diagnosis of Bone Tumor Cases
Standard Test Selection Function

Case and Pathology	Initial Attributes	Number of Tests Selected by Program	Number of Decision Nodes Considered	Distribution at Terminal Decision
1. (GC)	7	9	269	GC 0.78 CB 0.21
2. (OS)	10	12	425	OS 0.35 CS 0.65
3. (CB)	9	0	0	CB 1.00
4. (CS)	70	4	223	CS 0.99
5. (OS)	10	4	194	OS 1.00
6. (RC)	10	13	406	RC 0.68 ES 0.30 FS 0.01
7. (CS)	8	4	228	CS 0.78 FS 0.22
8. (ES)	8	11	475	ES 0.05 FS 0.07 RC 0.87
9. (ES)	6	5	278	ES 0.88 RC 0.05 OS 0.05 CS 0.02
10. (CB)	10	3	109	CB 0.96 CF 0.04
11. (GC)	10	5	169	GC 0.81 CS 0.10 CF 0.08 ES 0.01
12. (PS)	10	3	142	PS 0.93 FS 0.02 CS 0.05

Average number of tests by program = 7.1
Average number of decision nodes considered = 243

TABLE 23

Sequential Diagnosis of Bone Tumor Cases
Dominated Test Heuristic

Case and Diagnosis	Initial Attributes	Number of tests Selected by Program	Number of Decision Nodes Considered	Distribution at Terminal Decision
1. (GC	7	7	151	CB 0.73 GC 0.26
2. (OS)	10	17	211	CS 0.66 OS 0.34
3. (CB)	9	0	0	CB 1.00
4. (CS)	10	5	148	CS 0.82 ES 0.09 FS 0.05 OS 0.05
5. (OS)	10	3	139	OS 0.92 ES 0.02 CS 0.03 RC 0.03
6. (RC)	10	14	218	RC 0.70 ES 0.29
7. (CS)	8	4	180	CS 0.74 FS 0.26
8. (ES)	8	15	294	RC 0.90 FS 0.05 ES 0.03
9. (ES)	6	5	137	ES 0.87 CS 0.03 FS 0.01 OS 0.04 RC 0.04
10. (CB)	10	3	97	CB 0.96 CF 0.04

11. (GC)	10	5	119	GC 0.81 CS 0.10 ES 0.01 CF 0.08
12. (PS)	10	3	106	PS 0.92 CS 0.05 FS 0.02

Average number of tests by program = 6.6
Average number of decision nodes considered = 150

the heuristic.

This heuristic simulates to a certain extent the diagnostic strategy of one who seizes upon an initial view of the problem and later yields that view with considerable reluctance. Thus, the program makes a guess as to which tests will prove important at an early stage in the diagnosis, and thereafter restricts its attention to those tests as long as some appear to be useful. The difficulty is that the view on which the guess was made may not be an accurate one. Although the tests being considered may be of some value, there may be other tests, temporarily disregarded, which may be of greater value. Unfortunately, the heuristic is not sufficiently sensitive to changes in the current distribution, and it may cause relatively unfruitful paths to be pursued to an unnecessary extent. When it eventually abandons such a path and re-evaluates the formerly dominated tests, it may already have incurred unnecessary testing costs. The heuristic exhibits a "single-mindedness" which results in less than satisfactory performance.

In the bone tumor cases, this heuristic reduced both the average number of decision nodes considered and the average number of tests run. Here its failing is a loss of accuracy. This effect is extremely interesting. Apparently in its pursuit of an informative series of tests, the program succeeds in obscuring much of the information implicit in the initial attributes. As a result, when the undominated tests are finally released for consideration, the

current distribution is sufficiently altered that the program does not find additional tests worthwhile. This effect may be the cause of the results for case 1. Here the dominated test heuristic selected fewer tests in arriving at a less satisfactory diagnosis than the standard test selection function.

While the heuristic in question has some shortcomings, it does indicate a certain amount of promise. What it seems to lack is an awareness of changes in the current distribution which should cause certain dominated tests to be released for consideration. One possible solution is to save the current distribution with a set of dominated tests. This would allow the program to compare the present distribution with one in the stack to determine whether the view of the problem has changed sufficiently to warrant the release of the tests. This comparison could also account for the relative seriousness of states in deciding whether a given change were significant.

This example is but one of a number of heuristics which can be studied in the diagnostic system. Because very large decision trees may be encountered in future applications, a variety of tree-pruning heuristics should be studied.

Experiment 9. Exercise of the Pattern-Sorting Capability

A small example was constructed with which the pattern-sorting capability could be tested. This example consisted of six states and fifteen attributes. The matrix for the example is presented in

Table 24. The states in this example can be partitioned into two sets which have the property that certain attributes are specific to the states in a group and other attributes are shared by the two groups. The generator was employed to simulate case histories with noise attributes. That is, a case history for a state in the first group included one or more attributes selected from those specific to the states in the second group.

Consider the following diagnostic problem with the loss function as specified in Table 25. The initial attributes are S10, S12, S13 and S04. These attributes cannot be attributed to a single state, and so the pattern-sorting function produces more than one pattern. In this case the patterns formed are (S04) and (S10, S12, S13). For each of these patterns the distribution over states is obtained assuming that the given pattern is the only one. These distributions are:

1)	(S04):	DONE	0.24	2)	(S10,S12,S13):	DFOUR	0.42
		DTWO	0.11			DFIVE	0.02
		DTHREE	0.65			DSIX	0.57

Based on these distributions, the pattern-sorting function selects the current pattern. Here the choice is pattern 1 although it contains only one attribute. From the loss function matrix, it can be seen that state DONE is very serious. Since state DONE can exhibit S04, the posterior probability of DONE given S04 is non-zero (0.24). By considering both posterior probabilities and losses, the pattern-

TABLE 24

Artificial Structure

A priori Probability	Disease	SO1	SO2	SO3	SO4	SO5	SO6	SO7	SO8	SO9	S10	S11	S12	S13	S14	S15
0.15	DONE	.10	.50	.40	.30	.70	.10	.20	.10	.05						
0.20	DTWO	.15	.35	.50	.10	.20	.70	.05	.90	.10						
0.15	DTHREE	.60	.20	.20	.80	.05	.00	.05	.50	.60						
0.15	DFOUR	.10	.80	.10							.30	.10	.25	.30	.05	.20
0.20	DFIVE	.25	.35	.40							.15	.25	.05	.10	.60	.00
0.15	DSIX	.40	.35	.25							.35	.90	.25	.35	.55	.60

TABLE 25

Loss Function Matrix for Six State Problem
(in thousands)

	DONE	DTWO	DTHREE	DFOUR	DFIVE	DSIX
DONE	0	1	1	1	1	1
DTWO	100	0	1	1	1	1
DTHREE	100	1	0	1	1	1
DFOUR	100	1	1	0	1	1
DFIVE	100	1	1	1	0	1
DSIX	100	1	1	1	1	0

sorting function selects pattern 1 as the more serious, and hence it becomes the current pattern. Tests are selected relative to this pattern, but any new attributes are processed through the entire pattern stack as discussed in Chapter 4. In this particular example, the program continued diagnosis until the following situation was obtained:

1. (SO2, SO4, NOT SO6, SO7, NOT SO8, NOT SO9)

DONE	0.92
DTHREE	0.08

2. (S10, S12, S13, SO2)

DFOUR	0.62
DFIVE	0.01
DSIX	0.37

The program then tentatively attributed pattern 1 to state DONE. This left S10, S12, and S13 unaccounted for. At this point, the user terminated the diagnosis. Had he wished, he could have pursued the investigation, the original pattern was shown to be invalid, the attributes in it would be returned to the unaccounted-for set and the pattern would be removed from the stack.

A variety of such experiments were run with the pattern-sorting function and the results indicated that the particular scheme embodied in the function exhibits the desired properties. This function needs to be studied more extensively, however, especially in more complicated situations. Although this area was somewhat slighted in this research the environment provided by the diagnostic system should be a good one in which to pursue such a study.

Chapter 8

DISCUSSION OF THE RESEARCH

The research discussed in the preceding chapters suggests a number of questions and issues which merit additional comment. In this chapter an attempt is made to draw together a number of results and to consider their potential generality. Also of interest here are some of the possible extensions of this research which aim at developing a more sophisticated system for the study and performance of diagnosis.

One of the more obvious questions involves the evaluation of the performance of the current diagnostic program. This question is important for two reasons. First, one of the principal hypotheses considered in this research was that in a variety of problem areas, a computer program could prove a competent or superior diagnostician. The current program has been applied to a number of cases, simulated and actual, of bone tumor and congenital heart disease. Hence a reasonable question is how well did it perform. A second reason for establishing a meaningful performance measure is so that it can be used in studies of various diagnostic strategies. If one test selection heuristic is to be judged superior to another, the judgment must be based on a measure of performance, and that measure should reflect diagnostic capability. So there is a very

real need for a good measure of diagnostic performance.

Unfortunately, while the need for a performance measure is clear, the precise nature of such a measure is open to a number of questions. Perhaps the best way to approach the problem is to catalog those qualities for which a diagnosis is generally judged to be a good one. The most obvious of these qualities is the accuracy of the diagnosis. The object of diagnosis as stated in the beginning of this thesis is to ascertain the state of a system. All other things being equal, the more accurate the determination of the state of the system, the better the diagnosis. By itself, however, this quality has relatively little meaning. One desires to know the state of a system in a diagnostic problem because this knowledge is an input to a subsequent decision (e.g. the decision about a treatment plan for a medical problem). Accuracy is not sought for its own sake, but rather for its improvement of decisions which result from the diagnosis. If these latter decisions are independent of any particular alternative in a group of diagnostic decisions, then there is no benefit to be accrued from distinguishing one of this group from another. From the point of view of further decisions, the states corresponding to these decision alternatives constitute an equivalence class. If a doctor knows that a patient has one of three viruses, all of which would be treated in the same manner, there may be no value attempting to deduce the "actual" virus.

If one were interested in accuracy as the chief quality of good

diagnosis, he could contend that in the above example, the doctor was accurate in diagnosing the problem as one of three viruses and that this can be thought of in identifying the state of the patient. A simple extension of this example makes this objection less forceful, however. Suppose that each of the three viruses are treated in a different manner and that there is a loss of diagnosing any one as another, but in each case this loss is less than the testing loss required to distinguish one from another. Again the identification of the goal of diagnosis as accuracy seems incomplete. The point is that accuracy is sought only to an extent commensurate with the expected consequences of a diagnostic decision about the system and the expected cost of obtaining greater accuracy.

This view of the diagnostic process has been the basis for this research. From the point of view of the diagnostician, the goal of diagnosis is to minimize the sum of the testing loss and the expected decision loss. Conceivably a diagnostician could correctly ascertain the state of a system at such a testing cost that his diagnosis would be judged inferior.

While it is appropriate for a diagnostician to consider expected loss for misdiagnosis as a factor in determining the course of a diagnosis this quantity is not necessarily relevant to the judgment of his diagnostic performance. The principal reason for this is that the expected loss depends on the probability distribution over states which is held by the diagnostician at the time of a terminal decision.

Since the diagnostician chooses tests, this distribution reflects his testing strategy as well as the actual problem. Basing a performance measure on expected loss ignores the relative merits of different testing strategies. It is as though a doctor were to be given a high performance rating simply because he believed very strongly that he had discovered the patient's problem. This strong belief may well be founded on incomplete or irrelevant information.

A more satisfactory way of assessing diagnostic performance is to simply add the testing loss to the actual decision loss. That is, judge the act rather than the intent. Ideally, one could determine the actual decision loss by comparing the actual state of the system (when it becomes known) with the diagnostic decision and determining the loss attributable solely to the difference between the two. By this standard, a diagnostician who consistently minimized the sum of testing and decision losses would be judged to be superior. Some of the problems inherent in this measure are rather obvious. First, the actual state of the system may never be known with certainty. A patient who is diagnosed and treated may never return for further examination, and hence a serious misdiagnosis may never be uncovered. A second problem is the difficulty in apportioning the decision loss to various diagnostic decisions. Also, the loss itself may be very difficult to ascertain. Nonetheless, this measure does seem to subsume the desired properties, and although it may be difficult to apply, it does seem to be a standard to be sought.

Another consideration in evaluating diagnostic decisions couched in terms of probabilities is the interpretation of probability distributions. For example, what are the implications of a diagnosis of (0.75, 0.25) for the states S1 and S2 for a performance measure? To a large extent, it depends on the actions which are taken based on this diagnosis. Suppose the actual state is S2. How does this affect the evaluation of this diagnosis? If only a single action can be taken on this diagnosis and it is based on the belief that the state is S1, the problem is even more difficult. The influence of such a distribution on a human decision maker may be quite subtle. If individuals react differently to such distributions, the problems will be compounded.

Finally, some effort should be made to normalize performance measures. Certain problems may be inherently more difficult to diagnose than others. For this reason, it is important to obtain an understanding of the limitations placed upon even the most expert diagnostician by the very nature of the problem before him.

The evaluation of the performance of the diagnostic program in the particular problem areas of bone tumors and congenital heart disease is made more difficult by the lack of well-defined loss structure for these problems. This precludes the use of the total loss measure discussed above. An alternative approach is to compare the program performance with standards based on the performance of experienced doctors. Even this approach is somewhat indirect in this case. Since no studies of doctor performance with the particular

case histories used were performed, no immediate comparisons based solely on the results of this research are possible. Some indication of program performance, however, can be obtained in the following way. The problems of bone tumor diagnosis and heart disease diagnosis have been studied extensively by Lodwick and Warner respectively. Both developed computer programs to perform diagnosis and have compared the performance of these programs with that of experienced physicians. These comparisons suggested that the programs performed diagnosis of a quality comparable to that of an experienced physician when all attributes were presented to both physician and program. The fact that the current diagnostic program duplicates the results of these programs on the cases studied suggests that the current program would fare equally well in a comparison with physicians. In the absence of a performance measure, this is the strongest statement which the experimental evidence will support.

If one tentatively accepts this suggestion, then a second significant conclusion can be derived from the results of these experiments. The diagnostic program was able to solve problems in two different areas of medical diagnosis. These areas differ in both the number of diseases and the complexity of inter-attribute relationships which are considered. The latter aspect is particularly important because it was handled without changing the program. Since the experiments involved only two problem areas and both were medical, the applicability of the program for a wide class of problems has

not been established. Its success in the two areas mentioned, however, strengthens the belief that it does have wider applicability.

The fact that the program is independent of the content of the information structure might be of significant value in the use of the program with hierarchical structures. Consider, for example, the problem of diagnosing a very large set of diseases. One possibility would be to create a hierarchical structure in which many sub-structures exist. The structures for bone tumors and congenital heart disease might such sub-structures. At the higher levels, the states would be classes of diseases, such as heart disease. The goal of diagnosis at higher levels would be to determine the proper class of disease. When this determination had been made, a more detailed sub-structure for that disease class would be employed for a "finer" diagnosis. The same diagnostic program could deal with all sub-structures. This would be a great improvement over a large set of programs, one for each sub-structure.

Again, considering the results of diagnosing actual case histories, one can readily appreciate the advantage of sequential diagnosis. In the particular problems studied, the program was able to arrive at a diagnosis with the use of relatively few tests. This capability is very important since the testing cost for a diagnosis may be a significant part of the total cost. Tests which are unnecessary or uninformative may exact a high price, and an effort

should be made to restrict the tests run to those essential to the diagnosis. The sequential test selection facility permits the program to dynamically assess the potential usefulness of each possible test. This results in efficient testing strategies, an important component of good diagnosis.

In a problem area in which the tests relevant to different groups of states are relatively disjoint, the value of sequential testing should be even greater. Once the appropriate group of states has been established, the tests considered can be restricted to the set of tests associated with that group. In the absence of a sequential testing capability, it may be necessary to perform all tests to obtain information which could have been obtained from a few. The striking reduction in the number of tests required for diagnosis of bone tumors and congenital heart disease effected by sequential testing strongly suggests the potential value of this approach in other diagnostic problems.

The existence of a diagnostic system rather than just a diagnostic program has proved quite important in this research. Many of the strategies which were considered are quite complicated, and it is difficult to predict a priori the manner in which they will perform. The generator has been very useful in testing these strategies under a variety of problem conditions. Also of use has been the facility for selectively monitoring particular diagnostic functions such as pattern-sorting and test selection by collecting detailed data on

their operations.

One virtue of the inclusion of a generator in the diagnostic system is that it makes it possible to study the performance of the diagnostic program in problems derived from a wide range of information structures. The simulation capability frees the researcher from dependence on actual case histories. Thus he can create structures and simulated cases specifically designed to test some aspect of the diagnostic program. The use of the simulation facility with an information structure corresponding to an actual diagnostic problem may also be very useful in the study of that particular problem.

Complementing this capability is that of operating the diagnostic program in an interactive mode. Thus a user can employ the program in actual diagnostic problems. This "open end" of the system permits the independent testing of strategies developed through research, as well as making the diagnostic program a practical aid to problem solving. The experience gained in this research indicated the value of such a system which permits the study of both actual and artificial diagnostic problems. It seems that this type of system would prove must useful in further development of sophisticated strategies for computer-aided diagnosis.

Finally, the modularity of the system is very important. On the one hand, the insulation of the system functions from one another permits one to study a wide variety of diagnostic strategies since the functions can be changed independently of one another. Also as

better versions of these functions are developed, they can be incorporated into the system without restructuring it. In this sense, the performance of the system can be improved as additional experience with it is obtained.

The experience obtained with the diagnostic system has pointed to a number of areas for further research. A number of these areas are discussed here. Some pertain to specific improvements in the diagnostic capabilities of the program, while others have more general ramifications.

In Chapter 7, certain experiments to study the effect of the loss function on diagnosis were discussed. While these experiments are by no means exhaustive, they do indicate the strong effect the loss function can exert on diagnoses obtained by the program. Two major questions need to be investigated in this regard. The first is how such a loss function can be developed for a particular problem area, and the second is in what ways is diagnosis sensitive to the actual values of a loss function.

The first question is a very difficult one to answer. Assuming for the moment that the matrix form of the loss function is retained, the problem is to determine the "seriousness" of each possible misdiagnosis in some appropriate units. For example, in the context of medical diagnosis, one must answer questions such as "How serious is the diagnosis of pneumonia as influenza and vice versa?" This answer must be in such terms as to permit the comparison of a wide variety

of misdiagnoses in an orderly manner. If one considers the extreme range of consequences resulting from misdiagnoses in medicine, he can appreciate the magnitude of this task. As stated, the problem required the establishment of a common scale for such extremes as the failure to diagnose a simple cold and the failure to diagnose cancer.

In many instances, the loss for a misdiagnosis depends on many extraneous factors, such as whether a patient will return to the doctor when his symptoms persist. The loss may also depend on decisions made after the diagnosis which are difficult to predict. Compounding the problem of the loss function is the need to convert the testing loss to the same scale. In particular areas, one may be confronted with further complications in this regard. For example, the question of a loss function for medical diagnosis is also a question of <u>whose</u> loss function should be employed. One could answer that the loss function should be that of the patient. The loss function of the doctor, and that of society, however, are also possible answers to this question. If a diagnostic system were created for general use in medical diagnosis, questions such as these would have to be considered.

Although the problems of determining the loss function for an area as complex as medical diagnosis would be very great, they may well prove worth the effort of solution. If the value of a program for diagnosis in a given area can be clearly demonstrated to be considerable, this would be strong motivation for work on an ap-

propriate loss function. As currently conceived, such a diagnostic program would make extensive use of losses in directing a diagnosis. These losses should reflect the best understanding of the consequences of possible decisions. In some areas, the development of a loss function might be a valuable exercise independent of the implementation of a diagnostic program. In areas where sophisticated diagnosis is currently being performed by human beings, a loss function is often implicit. The attempt to quantify this loss function may reveal inconsistencies and reveal implicit losses of questionable merit. To the extent that this situation obtains in a particular area, there is additional motivation for research into this problem.

Such research would involve investigation of means of quantifying and scaling diverse consequences as well as considerations of the best form which the loss function should take. To a large extent, a framework for these investigations has already been established. A number of workers in the areas of statistical decision theory, game theory, and economics (R21, R22) have considered many of the problems associated with the attempt to scale decision alternatives. While this work is far from complete, it does provide a reasonable basis for some of the initial studies. This whole area is rich with problems of interest and importance.

Another important area for research is the development of a diagnostic program which includes improved solutions to a number of different problems, some of which are discussed here.

As previously noted, the test selection function merits particular attention. This function serves a central purpose in the overall diagnostic strategy of the program, and as a result, significant improvements in this area would directly promote the diagnostic capability of the program. More sophisticated test selection heuristics are required if the program is to deal successfully with problems involving large numbers of decision and testing alternatives. All the test selection heuristics employed in this research is "fixed-depth" in the sense that they explore all branches away from a given decision node to a fixed depth in the decision tree. Most likely a better test selection function would explore branches to varying depth, pursuing further those branches which appeared more promising. The difficulty yet to be overcome in this regard is the establishment of some measure of "promise" for branches in the decision tree. This problem has been encountered in other applications of heuristic programming, and it can be expected that significant results in the diagnostic problem would be of more general applicability. Similarly, if powerful test selection heuristics can be developed, they might be of considerable value in a variety of sequential decision problems.

Another improvement to the diagnostic program would allow it to take advantage of various relationships among tests. For example, if one is going to perform a certain test, it may be advantageous to perform another test as well because it is inexpensive when run

in conjunction with the first test. The inclusion of more complete information about tests in the information structure might allow the program to exploit various inter-test relationships and to select groups of tests to be run during diagnosis.

The pattern-sorting function needs to be bolstered by the addition of facilities for assessing the accuracy of the attributes provided it by the user. Just as it is important to detect noise attributes, it is equally important that the presence of false information be discovered. Undoubtedly only partial solutions to this problem are possible, but additional capabilities of this kind, even if somewhat limited, would be of considerable value in applications of the program to actual diagnostic problems. For example, the program could include a means for incorporating estimates of the reliability of tests into both the pattern-sorting and inference functions.

A number of improvements can be made in the inference function of the program. One of these is the incorporation of a learning scheme within this function. Such a scheme would permit the program to learn the a priori probabilities for the various states as well as the conditional probabilities of attributes of given states. Bayesian framework provides a convenient structure within which a learning scheme can be developed. Learning of this type is especially important if the relevant probabilities vary with the specific application. For example, if the information structure for congenital

heart disease were employed in a region of the country other than that in which it was developed, the probabilities might require adjustment to reflect changes in the characteristics of the population of potential patients. The program can obtain the information required for such an adjustment from the actual diagnoses which it performs on patients from the new population provided that other means of obtaining diagnoses are available. Thus in certain applications, the diagnostic program may require a training period in which it can alter the contents of the information structure to more accurately reflect the relevant behavior of the given system. A variety of learning schemes should be investigated to develop a scheme which will be suited for this problem.

Some of the considerations involved in research of this kind are apparent at the outset. If the probabilities of interest are relatively stable, then a rather prolonged learning period may be acceptable in the hope that these probabilities will be learned accurately. On the other hand, if the probability structure of the problem is relatively dynamic, then more rapid learning may be required. One difficulty with the latter situation is that rapid learning implies a greater weighting of recent experiences and if the environment is noisy, this may lead to poor probability estimates, and hence to poor diagnosis. One possibility is to exploit the ability of the human diagnostician to perceive patterns and trends by allowing him to influence probability estimates dynamically. For

instance, a doctor might be better able to detect the early stages of an epidemic and hence adjust the a priori probability of the prevalent disease to reflect its increased incidence.

Some Comments on the Diagnostic Model

When one devotes considerable attention to the problem of diagnosis, he may experience a tendency to generalize his definition of the problem so as to encompass an increasingly wide circle of problems. The danger of this tendency is that it may result in the extensive discussion of diagnostic programs and systems of impressive capabilities which are founded more on wishful thinking than on practical experience. Because the appeal of such an intellectual exercise is strong, it is important to consider carefully the model of the diagnostic problem being employed in order to obtain a realistic view of both its potential and limitations. Some of the important characteristics of the model employed in this research are investigated here with this intention.

A diagnostic model based on attribute-state relationships has understandable appeal. In many diagnostic problems the most visible aspect of an expert's attack on a problem is his gathering of attributes on which to base his decision. In many instances he may appear to relate these attributes directly to the possible states of the system. When the difficulty of diagnostic problems in general is considered, however, it seems unlikely that the human expert performs only a simple association of attributes and states to arrive

at a diagnosis. Diagnosis, as performed by humans, seems to be a subtle and often complex process of association and deduction.

The model employed in this research, on the other hand, is very explicit in the way in which it relates attributes and states. Associations in the information structure are relatively direct, and deduction is performed in a uniform manner for all problems. In one sense, the model employed by the diagnostic program appears quite rigid and simple. Even this brief comparison with human diagnosis suggests an important question. Can this relatively simple model be sufficient for a diagnostic program to perform effectively? A derivative of this question is the following. To what extent can a program based on this model be successful in performing diagnosis in a variety of problem areas? Although the evidence gathered from this research is far from sufficient to allow definitive answers to these questions, it does permit some insights into the problems to which these questions are addressed.

The author believes that the basic functions developed in this work reflect aspects of a diagnostic program which has both potential generality and power. At present, the functions are quite crude in their structure and capabilities, but the conception of diagnosis in terms of these functions (or their more sophisticated successors) is believed to be both a useful and viable one. One problem may be that the current separation of functions is somewhat restrictive, but this has the advantage of emphasizing the principal objectives and problems of each. This emphasis is very important in the initial phases of

research in this area, and the separation permits the study of different versions of one function more or less independently of the others.

In broad outline, the model incorporates the principal features of diagnosis as performed by human beings. The inference function coupled with the information structure allows the consideration of both past experience and current information in a particular diagnosis. Bayesian inference provides an orderly way for balancing these two elements in the deductive process. The test selection function provides the program with a rational means for choosing tests which accounts both for their cost and their potential value in furthering the diagnosis. Finally, the pattern-sorting function provides a means for performing diagnosis in the presence of noise.

While it is unlikely that the human diagnostician employs this particular division of the diagnostic function, the total capability incorporated in the functions seems to approximate that required. It is also important to note that there is no particular reason to require a diagnostic program to simulate the processes employed by humans. A more appropriate requirement is that a diagnostic program should allow the exploitation of the comparative advantages of a computer in order that the total diagnostic capability of a man-machine partnership may exceed that attainable by either above.

For example, it has been noted that doctors do not organize their diagnostic experience into large lists of symptoms and diseases, but rather associate their experience with and through their under-

standing of the human body and its processes. It would be extreme to conclude from this that such an organization is a necessary one for diagnosis, particularly if the diagnostician is a computer program. The fact that a doctor does not order his experience primarily in terms of attribute-disease lists may simply be evidence of the difficulty he encounters in attempting to deal with and maintain information of this form. A computer program would have less of a problem in this regard, and, in fact, this may be a useful structure to impose on the experience employed by a diagnostic program.

While in very general terms, the functions of the program correspond to those apparently required for diagnosis, there remain certain questions about limitations arising from their current realizations. In a sense these are questions about the generality of the model. Since the program was designed to solve the model diagnostic problem, it is reasonable to expect that the generality of the program will be determined by the extent to which real diagnostic problems can be described by the model. (Also, the appropriate statistical data must be available.)

For example, a major difficulty in applying the program to program debugging is developing a proper characterization of states. One can see in theory how this can be accomplished, but a practical solution would be extremely difficult. Also, an extremely useful strategy in program debugging is changing the state of the program (by

changing instructions, etc.) Here tests may very well change the state of the system. Because one can save a copy of the program, one can also use destructive testing. While one could probably change the model (and program) to reflect these possibilities, the current model does not account for them. Hence, the use of the program in this area is severely limited.

Also, there may be areas in which the diagnostic experience may not fit the statistical model employed in this work. In these areas, the inference function would have to be redone for non-Bayesian inference.

On the other hand, there seem to be a number of real problems which can be described by the model, including many machine failure and medical diagnosis problems. While the evidence is limited, the performance of the current diagnostic program in the areas of congenital heart disease and bone tumors should not be overlooked. At the very least these results must be termed promising. The model on which the program was based and the program itself were developed independently of considerations of these particular diagnostic problems, and yet the program demonstrated potential value in both areas. There seems reason to believe that other problems of medical diagnosis will also prove susceptible to such a program. The diagnostic system permits the study of alternative strategies developed in the light of such experiments, and this, too, should ease the problems of increasing the extent of its capabilities.

Some of the difficulty in applying the program to new areas can be traced more directly to a lack of adequate data for an information structure than to an inherent intractability to this approach. If continued research yields further indications of the value of a computer program for diagnosis, it may well be worth the considerable effort required to reformulate a number of diagnostic problems in terms of this model or an extension of it. Certainly, the results of this research do not preclude this possibility.

References

1. R. S. Ledley & L. B. Lusted, "The Use of Electronic Computers to Aid Medical Diagnosis," Prox. IRE Vol. 47, pp. 1970-1977, November 1959.

2. R. S. Ledley & L. B. Lusted "The Reasoning Foundations of Medical Diagnosis," Science, Vol. 130, pp. 9-21, July 3, 1959.

3. L. B. Lusted & R. S. Ledley "Mathematical Models in Medical Diagnosis," J. Med. Ed., Vol. 35, pp. 213-220, March 1960.

4. S. Rush, "A Logical Structure for Diagnosis Based on Probability," IRE National Convention Record, part 9, p. 10, 1959.

5. K Brodman, "Diagnostic Decisions by Machine," IRE Transactions on Medical Electronics, Vol. ME-7, 216, 1960.

6. K. Brodman, et al, "Interpretation of Symptoms With a Data Processing Machine," AMA Archives of Internal Medicine, 103: 776, 1959.

7. A. J. Lerner, "Formal Methods of Diagnostics in Engineering and Medicine," Second International Conference on the Development of Science and Technology and Their Impact on Society, Herceg Novi, Yugoslavia, 1966.

8. R. A. Bruce & S. R. Yarnall, "Computer-Aided Diagnosis of Cardiovascular Disorders," J. Chron. Dis., Vol. 19, pp. 473-484, 1966.

9. Overall, J. E. Williams, "Conditional Probability Program for Diagnosis of Thyroid Function," JAMA 183, No. 5, p. 307, 1963.

10. J. A. Rinaldo, et al, "Symptom Diagnosis: A Mathematical Analysis of Episgastric Pain," Ann. Int. Med., Vol. 59, No. 2, p. 145.

11. C. A. Nugent, "The Diagnosis of Cushing's Syndrome," The Diagnostic Process, Ed. J. A. Jacquez, Malloy Litho., Ann Arbor, Mich., 1964.

12. H. R. Warner, et al, "Experience with Bayes' Theorem for Computer Diagnosis of Congenital Heart Disease," Ann. N. Y. Acad Science, 115, p. 558, 1964.

13. A. F. Toronto, et al, "Evaluation of a Computer Program for Diagnosis of Congenital Heart Disease," Progr. Card. Diseases, Vol. 5, p. 362, 1962.

14. H. R. Warner, et al, "A Mathematical Approach to Medical Diagnosis," JAMA 177, No. 3, p. 144, 1961.

15. B. S. Sanders, "Completeness and Reliability of Diagnosis in Therapeutic Practice," quoted in M. L. Gross, The Doctors, p. 33, Random House, N. Y. 1966.

16. L. Clendening & E. H. Hashinger, "Methods of Diagnosis," C. V. Mosby Co., St. Louis, Mo., p. 59, 1947.

17. L. B. Lusted, "Computer Programming of Diagnostic Tests," IRE Trans. Med. Elect. M 7, p. 255, 1960.

18. E. G. Manning, "Self-Diagnosis of Electronic Computers--An Experimental Study," University of Illinois, Coordinated Science Laboratory Report R-259, July 1965.

19. H. Y. Chang, "An Algorithm for Selecting an Optimum Set of Diagnostic Tests," IEEE Trans. on Elec. Comp EC-14, No. 5, p. 706, 1965.

20. J. Weizenbaum, "Symmetric List Processor," Comm. ACM, Vol. 6, No. 9, September 1963.

21. D. Luce & H. Raiffa, Games and Decisions, John Wiley, N.Y.1957.

22. J. W. Pratt et al, Introduction to Statistical Decision Theory, McGraw-Hill, N. Y. 1965.

Appendix 1

Sample of an Input File

(STATE DO1 0.05 SO1 0.01 SO2 0.10 . . . S17 0.90)

(CLUSTR DO1 EXOR 0.05 SO6 0.07 SO7)

.
.
.

(ATTRIB (SO1 SO2 SO3) TEST1 SO4 TEST4 . . . (16 S17) TEST16)

.
.
.

(TESTS TEST1 10. . . . TEST16 15.)

Appendix 2

Trace of a Session with the Diagnostic Program

User responses = small letters
Program responses = capital letters

1. r system

2. NAME OF DIAGNOSTIC AREA PLEASE

3. bone tumors

4. NAME OF LOSS STRUCTURE FILE

5. bone losses

6. INFORMATION STRUCTURE ESTABLISHED

7. generate brief

8. YOU OR ME

9. me

10. HISTORY FILE

11. bone case

12. CODES

13. 3 2 2 2 3 2 3

14. NEW CASE
    WHAT ARE THE INITIAL ATTRIBUTES OF THE PROBLEM. Q.

15. s05 s07 s11 s14 s17 s20 not s21

16. CONDITIONAL PRIOR STATE PROB
    CB 0.26
    CS 0.09
    GC 0.62
    CF 0.02
    TRACE 0.01

17. ANY IDEAS. Q. TYPE 'DONE' IF SATISFIED.

18. c.r.

19. SET SEARCH DEPTH, THRESHOLD, AND HEURISTIC CONTROL

20. 1 0.10 0

21. THE TEST SELECTED IS TEST43

22. s43

23. CONDITIONAL PRIOR STATE PROB

CB	0.55
CS	0.04
GC	0.37
CF	0.04

.

.

.

24. THE TEST SELECTED IS TEST50

25. s50

26. CONDITIONAL PRIOR STATE PROB

CB	0.21
GC	0.78
TRACE	0.01

27. GC TENTATIVE DIAGNOSIS FOR THIS PATTERN

28. CONSISTENT DIAGNOSIS FOR ALL ATTRIBUTES

Notes

A. Line 7 through line 14. The user sets controls for the run. These controls include a history file and instructions as to what information is to be collected in this file during the run (line 13).

B. Line 15. These are the initial attributes of the problem.

C. Line 16. The inference function reports the current distribution.

D. Lines 17 and 18. The user is given the option of testing his hypothesis about the problem. He declines this option (line 18).

E. Lines 19 and 20. Here the user sets the depth and threshold for the test selection function. He also chooses the standard version of this function.

F. Lines 21 and 22. The program selects a test and the user responds. This dialogue continues through line 25.

G. Line 27 and line 28. The program makes a terminal decision for the pattern. This decision accounts for all attributes and the case is completed.

Appendix 3

Listings of Diagnostic System

```
COMMON   MAD
         P'N BUF1,BUF2,CBIT,SIGNS,CPAT,ALLPAT,CPRIOR
        1 ,CTEST,ALLTST,DISEAS,STAND,FILE1,FILE2,
        2 DEPTH,THRESH,NINITS,NOISE,NODES,BASE,TREE,CURLST
        3 ,PATLST,STRUCT,SYMCNT,SYMLST,UNACTD,TSTRUN,PATSTK
        4 ,CODE,OPSTCK,STACK,UFUNC,ARGS,PRIM,CONST,SPOINT
        5 ,NPRIM,CELL
         D'N BUF1(432),BUF2(432),STACK(20),UFUNC(20),ARGS(10)
        1, PRIM(30),NPRIM(30),CONST(30),CELL(20)


MASTER   MAD
          N'R
          INSERT FILE COMMON
          D'N VAULT(100)
          INITAS.(0)
          SET LIST TO VAULT
          LIST.(TREE)
          NEWVAL.($VALUES$,LIST.(9),LIST.(CELL(8)))
          LIST.(OPSTCK)
          LIST.(TSTRUN)
          LIST.(CELL(11)
          LIST.(SYMLST)
          LIST.(PATSTK)
          LIST.(UNACTD)
          LIST.(TEMP)
          LIST.(STRUCT)
          VAULT=0
          UFUNC=-1
          PRIM(1)=PLUS.
          PRIM(2)=MINUS.
          PRIM(3)=TIMES.
          PRIM(4)=DIVIDE.
          PRIM(5)=L.
          PRIM(6)=LE.
          PRIM(7)=EQ.
          PRIM(8)=GE.
          PRIM(9)=G.
          PRIM(10)=AND.
          PRIM(11)=OR.
          PRIM(12)=EQV.
          PRIM(13)=NOT.
          PRIM(14)=ATTRIB.
          PRIM(15)=PRES.
          V'S NPRIM=15,$PLUS$,$MINUS$,$TIMES$,$DIVIDE$,
         1 $L$,$LE$,$EQ$,$GE$,$G$,$AND$,$OR$,$EQV$,
         2 $NOT$,$ATTRIB$,$PRES$
          PRINT COMMENT $NAME OF DIAGNOSTIC AREA PLEASE$
          RDLOWL.(TEMP)
          N1=POPTOP.(TEMP)
          N2=POPTOP.(TEMP)
          SETUP.(N1,N2)
          PRINT COMMENT $NAME OF LOSS STRUCTURE FILE$
```

```
          RDLONL.(TEMP)
          N1=POPTOP.(TEMP)
          N2=POPTOP.(TEMP)
          SETLOS.(RJUST.(N1),RJUST.(N2))
          PRINT COMMENT $INFORMATION STRUCTURE ESTABLISHED.$
TOP       RDLONL.(TEMP)
          CODE=POPTOP.(TEMP)
          W'R CODE.E.$DEFINE$
             LST=DEFINE.(TEMP)
          O'R CODE.E.$GENER8$.OR. CODE.E.$GEN$
             GENER8.(TEMP)
             PRINT COMMENT $RETURN FROM GENER8.$
          O'R CODE.E.$CLUSTR$
          LST=CLUSTR.(TEMP)
             PRINT OCTAL RESULTS LST
          E'L
          MTLIST.(TEMP)
          T'O TOP
          E'M


 DIAG  MAD
          EXTERNAL FUNCTION (CONTRL)
          N'R
          F'T P,SELECT,FANS,FANS1
          B'N LEMPTY
          INSERT FILE COMMON
          EQUIVALENCE (IP,P),(FANS,ANS),(FANS1,ANS1)
         R
         R
         R THIS FUNCTION IS THE CONTROL ROUTINE FOR THE
         R DIAGNOSIS.IT MANAGES THE MACRO ASPECTS OF
         R THE DIAGNOSIS.
         R
         R
          E'O DIAG.
          COUNT=0
          MTLIST.(CELL(1))
          LIST.(TEMP)
          W'R STAND.E.1.OR.STAND.E.3, OUTPUT.(STAND,0,BLNK)
         R
         R GET AND PROCESS THE INITIAL SYMPTOMS WHICH DEFINE
         R THE PROBLEM
         R
          W'R CBIT.E.1, P'T ILINE
          V'S ILINE=$H/WHAT ARE THE INITIAL SIGNS OF THE PROBLEM/*$
          W'R GETSYM.(TEMP).E.0, F'N
          W'R LEMPTY.(TEMP), F'N
          MTLIST.(PATSTK)
          MTLIST.(UNACTD)
          MTLIST.(TSTRUN)
          MTLIST.(SYMLST)
          MTLIST.(TREE)
          V'S BLNK=$/,H/   NEW CASE/,/*$
         R
         R PROCESS THESE SYMPTOMS TO FORM SYMPTOM PATTERNS.
         R
LOOP      W'R LEMPTY.(TEMP), T'O GETPAT
          SYMP=POPTOP.(TEMP)
```

```
          TEST=BOT.(SYMP)
          NEWBOT.(TEST,TSTRUN)
          TEST=ITSVAL.($EXCLUS$,TEST)
          W'R TEST.NE.0, NEWBOT.(TEST,TSTRUN)
          SYMSAV.(SYMP,$INIT$)
          T'O LOOP
          R
          R
          R HERE IS WHERE THE MOST SERIOUS PATTERN IS
          R CHOSEN. DURING THIS ITERATION, TESTS WILL BE EVALUATED
          R RELATIVE TO THIS PATTERN.
          R
GETPAT    P=SELECT.(0)
          OUTPUT.(CPAT,0,INIFRM)
          V'S INIFRM=$/,H/THE CURRENT PATTERN IS.../*$
          V'S CLINE=$H/THE WEIGHT OF THIS PATTERN IS /,F4.2,1H.*$
          PDUMP1.(CPAT,PATLST)
          OUTPUT.(CPAT,1,CLINE,R)
          DDUMP1.(CPRIOR,CURLST)
          OUTPUT.(ALLPAT,0,OTFMT)
          V'S OTFMT=$/,H/OTHER PATTERNS.../,*$
          DUMPR.(ALLPAT)
          R
          R
          R HERE CHECK THE CURRENT STATE PRIOR FOR A SUCCESSFUL
          R DIAGNOSIS OF THE CURRENT PATTERN.
          R
          R=LADROV.(CURLST)
TL        IP=ADVLER.(R,F)
          W'R F.E.1
              IRARDR.(R)
              T'O DOTEST
          O'R P.L..99
              T'O TL
          E'L
          NAME=ITSVAL.($PNAME$,CONT.(LNKL.(CONT.(LPNTR.(R)))+1))
          OUTPUT.(STAND,1,ANSFRM,NAME)
          V'S ANSFRM=$H/THE CURRENT PATTERN IS ATTRIBUTED TO /,
          1 C6,//*$
          R
          R CHECK FOR MORE SYMPTOMS TO EXPLAIN.
          R
SUCC      GOTPAT.(0)
          W'R CODE.E.0, T'O GETPAT
          OUTPUT.(STAND,0,OKVS)
          V'S OKVS=$H/CONSISTENT DIAGNOSIS FOR ALL SIGNS./*$
          IRALST.(TEMP)
          F'N $OK$
          R
          R
          R
DOTEST    LCOUNT=0
          W'R CBIT.E.0.OR.CPRIOR.E.2
              NSTATE=0
              WORD=0
              T'O SEEK
          E'L
          PRINT COMMENT $ANY IDEAS.Q. 'DONE' IF SATISFIED.$
RRD       CONTINUE
          R'T C6, WORD
          V'S C6=$C6*$
```

```
            W'R WORD.E.$DONE$
               OUTPUT.(STAND,0,UTERM)
               V'S UTERM=$H/USER TERMINATED DIAGNOSIS OF PATTERN/*$
               T'O SUCC
            O'R WORD.E.$NO$.OR.WORD.E.$$
               NSTATE=0
               WORD=0
            O'E
               NSTATE=TRANS.(WORD,1)
               W'R NSTATE.E.0
                 PRINT COMMENT $NOT RECOGNIZED. TRY AGAIN.$
                 T'O RRD
               E'L
            E'L
            R
            R 'SEQDEC' IS THE TEST SELECTION ROUTINE.
            R
            PRINT COMMENT $SET DEPTH,THRESHOLD, AND HEURISTIC CONTROL.$
            W'R .NOT. LEMPTY.(RDLONL.(TEMP))
            DEPTH=POPTOP.(TEMP)
            THRESH=POPTOP.(TEMP)
            CONTRL=POPTOP.(TEMP)
            E'L
SEEK        NODES=0
            STATE=NSTATE
            W'R STAND.GE.2, OUTPUT.(2,2,CFRM,DEPTH,THRESH)
            SEQDEC.(TREE,0,STATE)
            W'R ALLTST.E.0, T'O GETTST
            RDR=SEQRDR.(ITSVAL.($VALUES$,TREE))
            STATE=SEQLR.(RDR,I)
            IP=SEQLR.(RDR,I)
            W'R WORD.NE.0
               W'R STATE.E.$DUMMY$
                  PRE=$NOT$
               O'E
                  PRE=$$
               E'L
            O'E
               WORD=ITSVAL.($PNAME$,STATE)
               PRE=$$
            E'L
            OUTPUT.(ALLTST,3,TRMD,PRE,WORD,P)
            V'S TRMD=$H/ BEST TERMINAL DECISION AT THIS POINT IS /,C3,/
           1,C6,H/ WITH EXPECTED LOSS /,F8.2*$
            OUTPUT.(ALLTST,0,THEAD)
            V'S THEAD=$H/ TEST      COST        E(LOSS)/*$
TLOOP       TEST=SEQLR.(RDR,I)
            W'R I.NE.1
               ANS=SEQLR.(RDR,I)
               ANS1=BOT.(TEST)
               NAME=ITSVAL.($PNAME$,TEST)
               OUTPUT.(ALLTST,3,TLIN,NAME,FANS1,FANS-FANS1)
               V'S TLIN=$C6,3S,F5.1,5S,F8.2*$
               T'O TLOOP
            E'L
            OUTPUT.(ALLTST,1,SCORE,NODES)
            V'S SCORE=$I6,H, DECISION NODES CONSIDERED.,//*$
            R
            R SELECT THE BEST TEST
            R
GETTST      W'R CONTRL.G.0
```

```
            LC=TOPTH.(TEST,STATE)
            COUNT=COUNT+LC
            T'O CKS
        E'L
AGAIN   TOPT.(TEST,STATE)
CKS     W'R STATE.NE.0
            LCOUNT=LCOUNT+LC
            W'R LCOUNT.L.COUNT
              POPBOT.(CELL(1))
              COUNT=COUNT-1
            MTLIST.(ITSVAL.($VALUES$,TREE))
              T'O SEEK
            E'L
DECIDE      OUTPUT.(STAND,1,DF,ITSVAL.($PNAME$,STATE))
            V'S DF=$C6,H/ TENTATIVE DECISION FOR THIS PATTERN./*$
            T'O SUCC
        O'E
            NEWBOT.(TEST,TSTRUN)
            TTEST=ITSVAL.($EXCLUS$,TEST)
            W'R TTEST.NE.0, NEWBOT.(TTEST,TSTRUN)
        E'L
        R
       R TEST HAS BEEN SELECTED. NOW RUN IT.
        R
        MTLIST.(TEMP)
        W'R CBIT.E.0, NEWTOP.(TEST,TEMP)
        OUTPUT.(CTEST,1,TFRM,ITSVAL.($PNAME$,TEST))
        GETSYM.(TEMP)
        W'R LEMPTY.(TEMP), T'O AGAIN
TRES1   SYMP=POPTOP.(TEMP)
        SYMSAV.(SYMP,TEST)
        TEST=BOT.(SYMP)
        NEWBOT.(TEST,TSTRUN)
        TEST=ITSVAL.($EXCLUS$,TEST)
        W'R TEST.NE.0, NEWBOT.(TEST,TSTRUN)
        W'R .NOT. LEMPTY.(TEMP), T'O TRES1
        T'O GETPAT
        V'S CFRM=$H/DEPTH=/,I2,H/ AND THRESH=/,E4.2*$
        V'S TFRM=$H/THE TEST SELECTED IS /,C6*$
        E'N


GENER8  MAD
        EXTERNAL FUNCTION (X)
        N'R
        F'T RANNO,OLDP,P,PR,TESTP
        B'N DMARK,LEMPTY
        INSERT FILE COMMON
        EQUIVALENCE (IPR,PR)
        E'O GENER8.
       R
       R GENER8 IS THE SIMULATOR FOR THE
       R DIAGNOSTIC SYSTEM.
        R
        LIST.(WORK)
        W'R .NOT.LEMPTY.(X)
            POPTOP.(X)
            NORUNS=FAST.(CONTRL)
            T'O OKTOGO
```

```
          E'L
          PRINT COMMENT $IN THE GENERATOR. WHO IS CONTROL.Q.$
          R'T C6, ANS
          W'R ANS.NE.$YOU$
             CBIT=1
             PRINT COMMENT $DO YOU WISH A HISTORY TO BE KEPT.Q.$
             R'T C6, ANS
             W'R ANS.E.$YES$
               T'O GETFIL
             O'E
             FILE1=0
             FILE2=0
               T'O GETINC
             E'L
          E'L
          R
          R GENERATOR CONTROL HERE. SET CONTROLS.
          R
          CBIT=0
          PRINT COMMENT $HOW MANY CASES IN THIS RUN.Q.$
          NORUNS=POPTOP.(RDLONL.(WORK))
          PRINT COMMENT $NAME OF ONE DISEASE OR C.R. FOR RANDOM DRAW$
RSTAT     R'T C6, ANS
          W'R ANS.E.$$
             DMARK=0B
          O'E
             DMARK=1B
             DISEAS=TRANS.(ANS,1)
             W'R DISEAS.E.0
               PRINT COMMENT $NOT RECOGNIZED. TRY AGAIN.$
               T'O RSTAT
             E'L
             IPR=ITSVAL.($PROB$, DISEAS)
          E'L
          PRINT COMMENT $PLEASE SPECIFY (IN THE ORDER GIVEN) THE$
          PRINT COMMENT $FOLLOWING CONTROL PARAMETERS FOR THE RUN$
          PRINT COMMENT $1. DEPTH OF THE TREE SEARCH.$
          PRINT COMMENT $2. BREADTH LIMITING PROBABILITY.$
          PRINT COMMENT $3. NO. OF INITIAL SIGNS PER CASE.$
          PRINT COMMENT $4. NO. OF NOISE SIGNS PER CASE.$
          PRINT COMMENT $5. HEURISTIC CONTROL FOR TEST SELECTION.$
          R
          R
          DEPTH=POPTOP.(RDLONL.(WORK))
          THRESH=POPTOP.(WORK)
          NINITS=POPTOP.(WORK)
          NOISE=POPTOP.(WORK)
          CONTRL=POPTOP.(WORK)
          R
          R
GETFIL    PRINT COMMENT $NAME HISTORY FILE.$
          FILE1=RJUST.(POPTOP.(RDLONL.(WORK)))
          FILE2=RJUST.(POPTOP.(WORK))
          ASSIGN.(FILE1,BUF1,BUF2)
          T'O GETINF
          R
          R
GETINC    P'T CDF
          V'S CDF=$H/FOR EACH OF THE FOLLOWING, TYPE '1' FOR A/,/,
          1H/CONSOLE TRACE, '0' OTHERWISE./*$
          T'O RDINF
```

```
GETINF      PRINT COMMENT $FOR EACH OF THE FOLLOWING, RESPOND$
            PRINT COMMENT $'1' IF YOU WISH A CONSOLE TRACE,$
            PRINT COMMENT $'2' IF YOU WISH A HISTORY RECORD,$
            PRINT COMMENT $ '3' IF YOU WISH BOTH, AND '0' IF NEITHER.$
RDINF       PRINT COMMENT $1. CURRENT DISTRIBUTIONS$
            PRINT COMMENT $2. CURRENT PATTERNS$
            PRINT COMMENT $3. PATTERN STACK$
            PRINT COMMENT $4. TESTS AND VALUES$
            PRINT COMMENT $5. TEST SELECTED$
            PRINT COMMENT $6. SIGNS OF THE PROBLEM$
            PRINT COMMENT $7. STANDARD INFORMATION$
            CPRIOR=POPTOP.(RDLONL.(WORK))
            CPAT=POPTOP.(WORK)
            ALLPAT=POPTOP.(WORK)
            ALLTST=POPTOP.(WORK)
            CTEST=POPTOP.(WORK)
            SIGNS=POPTOP.(WORK)
            STAND=POPTOP.(WORK)
OKTOGO      PRINT COMMENT $$
            W'R FILE1.NE.0, DWRITE.(FILE1,HEAD,CPRIOR,CPAT,ALLPAT,
           1 ALLTST,CTEST,SIGNS,STAND)
            W'R CBIT.E.1
               T'O DODIAG
            O'R DMARK
               T'O START
            E'L
           R
           R SET UP DISEASE SELECTION LIST.
           R
            P=0.
            LIST.(GENLST)
            RDR=SEQRDR.(TOP.(STRUCT))
SLOOP       HSHLST=SEQLR.(RDR,I)
            W'R I.NE.1
            R=SEQRDR.(HSHLST)
HLOOP       NEXT=SEQLR.(R,F)
            W'R F.E.1, T'O SLOOP
            IPR=ITSVAL.($PROB$,NEXT)
            P=P+PR
            MANY.(GENLST,NEXT,P)
            T'O HLOOP
            E'L
           R
           R WARM UP RANNO.
            T'H RIN, FOR J=1,1,J.G.20
RIN         RANNO.(X)
           R
           R CONTROL LOOP FOR THE GENERATOR.
           R
START       T'H GEND, FOR J=1,1,J.G.NORUNS
            OLOP=0.
            P'T CON, J
            V'S CON=$H/CASE /,I2=$
            W'R DMARK, T'O GOTIT
            TESTP=RANNO.(X)
            RDR=SEQRDR.(GENLST)
GLOOP       DISEAS=SEQLR.(RDR,I)
            W'R I.E.1
               OUTPUT.(STAND,0,BUG)
               V'S BUG=$H/BUG IN GENLST/=$
               CHNCOM.(0)
```

```
          O'E
             IPR=SEQLR.(RDR,I)
             W'R OLDP.LE.TESTP.AND.TESTP.L.PR, T'O GOTIT
             OLDP=PR
             T'O GLOOP
          E'L
          R
          R
GOTIT     OUTPUT.(STAND,3,HEAD1,J,ITSVAL.($PNAME$,DISEAS),PR-OLDP)
          CURLST=0
DODIAG    DIAG.(CONTRL)
          W'R CBIT.E.1
             PRINT COMMENT $ANOTHER.Q.$
             R'T C6, ANS
             W'R ANS.E.$YES$, T'O DODIAG
             T'O FINI
          E'L
GEND      CONTINUE
          OUTPUT.(STAND,0,TERM)
          IRALST.(GENLST)
          V'S TERM=$//,H/RUN COMPLETED./*$
FINI      FILE.(FILE1)
          IRALST.(WORK)
          F'N
          R
          R
          V'S I1=$I1*$
          V'S C6=$C6*$
          V'S HEAD=$H/SWITCHES FOR THIS RUN/,/,
         1 7HCPRIOR=,I1,1S,5HCPAT=,I1,1S,7HALLPAT=,
         2 I1,1S,7HALLTST=,I1,1S,6HCTEST=,I1,1S,6HSIGNS=,
         3 I1,1S,6HSTAND=,I1,/*$
          V'S HEAD1=$//,H/**********************************/,
         1 //,H/CASE /,I3,H/. DISEASE IS /,C6,2H (,
         2 F3.2,2H)./*$
          E'N


GETSYM    MAD
          EXTERNAL FUNCTION (LST)
          N'R
          F'T TESTP,PR,POLD,PNEW,PIJ,RANNO
          INSERT FILE COMMON
          B'N LEMPTY,NAMTST,SPTEST
          R
          R THIS FUNCTION HANDLES ALL THE SIGN RETRIEVAL
          R ACTIVITY FOR THE DISEASE GENERATOR AND THE
          R DIAGNOSTIC PROGRAM.
          R
          E'B GETSYM.
          RET=1
          LIST.(WORK)
          W'R CBIT.E.1
          W'R SIGNS.G.1
             DS=2
          O'E
             DS=0
          E'L
          R
```

```
          R HERE THE USER IS IN CONTROL. SIMPLY RETRIEVE
          R THE NEXT SYMPTOM FROM HIM ( WITH TRANSLATION
          R AND CHECKING). RECORD SYMPTOM AS CALLED FOR.
          R
           OUTPUT.(DS,0,FIRST)
           RDLONL.(WORK)
LOOP       W'R .NOT. LEMPTY.(WORK)
             NAME=POPTOP.(WORK)
             W'R NAME.E.$NOT$.OR.NAME.E.$NO$
             NL=POPTOP.(WORK)
             W'R  NAMTST.(NL)
               STRANS.(0)
             O'E
               R=SEQRDR.(NL)
SL             NL=SEQLR.(R,F)
               W'R F.NE.1
                 STRANS.(0)
                 T'O SL
               E'L
             E'L
           INTERNAL FUNCTION (DM)
           E'O STRANS.
           WORD=TRANS.(NL,2)
           W'R WORD.E.0, T'O ERRMRK
           NEWTOP.(-WORD,LST)
           OUTPUT.(DS,1,NRM,NL)
           F'N
           E'N
             O'R NAME.E.$NORMAL$
               R=SEQRDR.(POPTOP.(WORK))
TL             NEXT=SEQLR.(R,F)
               W'R F.NE.1
               OUTPUT.(DS,1,NT,NEXT)
               V'S NT=$H/NORMAL /,C6=$
                 R1=SEQRDR.(ITSVAL.($MEMBER$,TRANS.(NEXT,3)))
TL1              SYMP=SEQLR.(R1,F1)
                 W'R F1.E.1, T'O TL
                 NEWTOP.(-SYMP,LST)
                 T'O TL1
               E'L
             O'E
               WORD=TRANS.(NAME,2)
               W'R WORD.E.0, T'O ERRMRK
               NEWTOP.(WORD,LST)
               OUTPUT.(DS,1,POS,NAME)
             E'L
             T'O LOOP
           E'L
          R
          R WHEN THE CURRENT LIST IS EMPTY, INITIAL SYMPTOMS
          R MUST BE GENERATED.
          R
           O'R CURLST.E.0
             OUTPUT.(SIGNS,0,INIFRM)
             MANY.(LIST.(TEMP),DISEAS,1.0)
             COUNT=RELIST.(WORK,TEMP)
             IRALST.(TEMP)
          R
          R HERE THE INITIAL TESTS ARE CHOSEN AT RANDOM
          R TO OBTAIN THE INITIAL SYMPTOMS.
          R
```

```
            SWITCH=0
            T'H TGLOOP, FOR J=1,1,J.G.NINITS
            W'R COUNT.E.0, T'O OUT
               KTH=COUNT*RANNO.(X)+1
               K=0
               RDR=SEQRDR.(WORK)
GET1           TEST=SEQLR.(RDR,I)
               K=K+1
               W'R K.L.KTH, T'O GET1
               COUNT=COUNT-1
               NEWTOP.(TEST,LST)
               SYMGEN.(LST)
               REMOVE.(LPNTR.(RDR))
               W'R TOP.(LST).G.0, SWITCH=1
TGLOOP      CONTINUE
OUT         W'R SWITCH.E.0
               OUTPUT.(STAND,0,NOS)
               RET=0
            E'L
            R
            R HERE A RESPONSE TO A PARTICULAR TEST IS
            R REQUIRED. THE TEST IS ON THE TOP OF 'LST'.
            R
            O'E
               SYMGEN.(LST)
            E'L
            R
BACK        IRALST.(WORK)
            F'N RET
ERRMRK      OUTPUT.(STAND,0,ERR)
            T'O LOOP
            R
            R THIS FUNCTION SELECTS A RESPONSE AT RANDOM
            R TO THE TEST ON THE TOP OF THE 'LST' GIVEN
            R THE KNOWN DISEASE 'DISEAS'.
            R
            INTERNAL FUNCTION (X)
            E'O SYMGEN.
            TEST=POPTOP.(LST)
            W'R ITSVAL.($PTEST$,TEST).E.$YES$
               SPTEST=1B
            O'E
               SPTEST=0B
            E'L
            TESTP=RANNO.(X)
            POLD=0.
            R=SEQRDR.(ITSVAL.($MEMBER$,TEST))
GLOOP       NEXT=SEQLR.(R,S)
            W'R S.E.1
               W'R SPTEST, F'N
GLOOP1         NEXT=SEQLR.(R,S)
               W'R S.E.1, F'N
               NAME=ITSVAL.($PNAME$,NEXT)
               OUTPUT.(SIGNS,1,NRM,NAME)
               NEWTOP.(-NEXT,LST)
               T'O GLOOP1
            O'E
               LOC=MEMBER.(DISEAS,ITSVAL.($MEMBER$,NEXT),0)
               W'R LOC.E.0, T'O GLOOP
               PR=PIJ.(NEXT,CONT.(LNKR.(CONT.(LOC))+1))
               PNEW=POLD+PR
```

```
            W'R POLD.L.TESTP.AND.TESTP.L.PNEW
              NAME=ITSVAL.($PNAME$,NEXT)
              OUTPUT.(SIGNS,1,POS,NAME)
              NEWTOP.(NEXT,LST)
              W'R SPTEST
                POLD=0.
                T'O GLOOP
              E'L
              F'N
            O'E
              POLD=PNEW
              T'O GLOOP
            E'L
          E'L
          E'N
        R
        R
          V'S FIRST=$H/USER RESPONSE /,C6=*$
          V'S NOS=$H/INITIAL SIGNS ARE ALL 'NORMAL' SIGNS./*$
          V'S ERR=$H/SIGN NOT RECOGNIZED. IGNORED./*$
          V'S INIFRM=$H/THE INITIAL SIGNS OF THE PROBLEM ARE/*$
          V'S NRM=$H/OBSERVED SIGN 'NOT /,C6,2H'.*$
          V'S POS=$H/OBSERVED SIGN '/,C6,2H'.*$
          E'N


PDUMP   MAD
          EXTERNAL FUNCTION (MARK)
          N'R
          F'T WGT
          EQUIVALENCE (IWGT,WGT)
          INSERT FILE COMMON
          E'O DUMPP.
          W'R ALLPAT.E.0, F'N
          OUTPUT.(ALLPAT,0,BLANK)
          R=SEQRDR.(PATSTK)
          COUNT=0
LOOP      NEXT=SEQLR.(R,F)
          W'R F.E.1
            W'R COUNT.E.1, OUTPUT.(ALLPAT,0,ONLY)
            OUTPUT.(ALLPAT,0,BLANK)
            F'N
          E'L
          COUNT=COUNT+1
          W'R NEXT.L.0, T'O LOOP
          W'R NEXT.E.CURLST.AND.CPAT.E.ALLPAT,T'O LOOP
          IWGT=ITSVAL.($WEIGHT$,NEXT)
          PDUMP1.(ALLPAT,ITSVAL.($SYMPS$,NEXT))
          OUTPUT.(ALLPAT,1,CLINE,WGT)
          DDUMP1.(ALLPAT,NEXT)
          T'O LOOP
        R
        R
          V'S BLANK=$/*$
          V'S ONLY=$H/CURRENT PATTERN IS THE ONLY ONE./*$
          V'S CLINE=$H/PATTERN WEIGHT=/,F4.2*$
          E'N
```

```
DUMP1    MAD
           EXTERNAL FUNCTION (MARK,LST)
           N'R
           F'T P,PTOT
           EQUIVALENCE (IP,P)
           INSERT FILE COMMON
           E'O PDUMP1.
           W'R MARK.E.0, F'N
           CNT=0
           R=SEQRDR.(LST)
LOOP       SYMP=SEQLR.(R,F)
           W'R F.E.1
              W'R CNT.G.0, OUTPUT.(MARK,CNT,SYLIN,$ARRAY$)
              OUTPUT.(MARK,0,BLANK)
              F'N
           E'L
           W'R SYMP.L.0
              CNT=CNT+1
              STACK(CNT)=$NOT $
           E'L
           CNT=CNT+1
           STACK(CNT)=ITSVAL.($PNAME$,SYMP).V.$000,00$
           W'R CNT.G.17
              OUTPUT.(MARK,CNT,SYLIN,$ARRAY$)
              CNT=0
           E'L
           T'O LOOP
         R
         R
           E'O DDUMP1.
           W'R MARK.E.0, F'N
           PTOT=0.
           R=SEQRDR.(LST)
           OUTPUT.(MARK,0,DLINE)
LOOP1      STATE=SEQLR.(R,F)
           W'R F.E.1
              OUTPUT.(MARK,2,LINE,TRACE,1.-PTOT)
              OUTPUT.(MARK,0,BLANK)
              F'N
           E'L
           IP=SEQLR.(R,F)
           W'R P.L.1.E-2, T'O LOOP1
           OUTPUT.(MARK,2,LINE,ITSVAL.($PNAME$,STATE),IP)
           PTOT=PTOT+P
           T'O LOOP1
         R
         R
           V'S BLANK=$/*$
           V'S TRACE=$TRACE$
           V'S SYLIN=$18C4**$
           V'S DLINE=$H/CONDITIONAL PRIOR    STATE  PROB/,/*$
           V'S LINE=$20S,C6,F4.2*$
           E'N


OUTPUT   MAD
           EXTERNAL FUNCTION (MARK,NARGS,FMT,A1,A2,A3)
```

```
            N'R
            INSERT FILE COMMON
            E'O OUTPUT.
            W'R A1.E.$ARRAY$, T'O DO(MARK)
            STACK(1)=A1
            STACK(2)=A2
            STACK(3)=A3
            T'O DO(MARK)
DO(3)       CONTINUE
DO(2)       W'R NARGS.E.0
               DWRITE.(FILE1,FMT)
            O'E
               DWRITE.(FILE1,FMT,STACK(1)...STACK(NARGS))
            E'L
            T'O DO(MARK-2)
DO(1)       W'R NARGS.E.0
               P'T FMT
            O'E
               P'T FMT, STACK(1)...STACK(NARGS)
            E'L
DO(0)       F'N
            E'N


SELECT    MAD
            EXTERNAL FUNCTION (DUMMY)
            R
            R THIS FUNCTION EXAMINES ALL THE PATTERNS IN THE
            R PATTERN STACK. IT RETURNS THE NAME OF THE PATTERN
            R WHICH HAS MINIMUM EXPECTED LOSS AS 'PATLST'.
            R THE DISTRIBUTION CORRESPONDING TO THIS PATTERN BECOMES
            R 'CURLST', THE CURRENT DISTRIBUTION.
            R
            N'R
            INSERT FILE COMMON
            F'T WEIGHT,WGT,PSAVE
            E'O SELECT.
            PSAVE=0.
            FIRMUP.(PATSTK)
            RDR=SEQRDR.(PATSTK)
            SAVLST=CURLST
TLOOP       NEXT=SEQLR.(RDR,I)
            W'R I.E.1
            R
            R UPDATE THE TREE
            R
               MTLIST.(TREE)
               NEWVAL.($VALUES$,LIST.(9),TREE)
               NEWVAL.($PRIOR$,CURLST,TREE)
               W'R SAVLST.NE.CURLST, MTLIST.(CELL(1))
               F'N PSAVE
            E'L
               WGT=WEIGHT.(NEXT)
               NEWVAL.($WEIGHT$,WGT,NEXT)
               W'R NEXT.L.0, T'O TLOOP
               W'R WGT.G.PSAVE
                  PSAVE=WGT
                  CURLST=NEXT
                  PATLST=ITSVAL.($SYMPS$,NEXT)
```

```
            E*L
            T*0 TLOOP
          R
          R
          R THIS FUNCTION UPDATES THE UNACCOUNTED-
          R FOR LIST AFTER A SUCCESSFUL DIAGNOSIS OF A PATTERN.
          R IT CONTROLS THE FORMATION OF NEW PATTERNS FROM THE
          R SYMPTOMS REMAINING ON THE UNACTD LIST.
          R
            E*0 GOTPAT.
            CODE=1
            RDR=SEQRDR.(UNACTD)
LOOP        SYMP=SEQLR.(RDR,I)
CHECK       W*R I.E.1, T*0 PRUNE
            LOC=MEMBER.(SYMP,PATLST,0)
            W*R LOC.E.0
              W*R SYMP.G.0, CODE=0
              T*0 LOOP
            O*E
              ADD=LPNTR.(RDR)
              SYMP=SEQLR.(RDR,I)
              REMOVE.(ADD)
              T*0 CHECK
            E*L
          R
          R
PRUNE       W*R CODE.E.1, F*N
            RDR=SEQRDR.(PATSTK)
            LIST.(TEMP)
LOOP1       NULST=SEQLR.(RDR,I)
            W*R I.E.1
              T*0 RESTOR
            O*R NULST.L.0
              NEWBOT.(NULST,TEMP)
            O*R NULST.E.CURLST
              NEWBOT.(-NULST,TEMP)
            E*L
            T*0 LOOP1
          R
          R
RESTOR      RDR=SEQRDR.(UNACTD)
            MTLIST.(PATSTK)
LOOP2       SYMP=SEQLR.(RDR,I)
            W*R I.E.1
              INLSTR.(TEMP,PATSTK)
              IRALST.(TEMP)
              F*N
            E*L
            W*R SYMP.G.0, PATFRM.(SYMP)
            T*0 LOOP2
            E*N


PATFRM  MAD
            EXTERNAL FUNCTION (SYMP)
          R
          R THIS FUNCTION FORMS ALL THE DISTINCT PATTERNS
          R FOR A GIVEN SYMPTOM, *SYMP*. IT PROCESSES
          R ALL PATTERNS SO FORMED AGAINST THE CURRENT
```

```
          R PATTERN STACK. IF THE PATTERN IS A NEW ONE, IT
          R IS RETAINED. OTHERWISE IT IS DISCARDED.
          R
          N'R
           B'N SUBSET
           INSERT FILE COMMON
           F'T P, UPD1
           E'O PATFRM.
           MEMLST=ITSVAL.($MEMBER$,SYMP)
          R
          R PROCESS THE SYMPTOM PATTERN FOR EACH STATE ON THE
          R MEMBER LIST OF 'SYMP'.
          R
           RDR=SEQRDR.(MEMLST)
LOOP       STATE=SEQLR.(RDR,I)
           W'R I.E.1, F'N
          R
          R CHECK FOR THIS STATE IN THE CURRENT PATTERN STACK.
          R IF IT IS THERE THEN ITS SYMPTOM PATTERN MUST ALSO
          R BE THERE, AND IT SHOULD BE IGNORED.
          R
           SEQLR.(RDR,I)
           W'R MEMBER.(STATE,PATSTK,1).NE.0, T'O LOOP
          R
          R STATE NOT FOUND IN PATTERN STACK. SYMPTOM PATTERN
          R FOR THIS STATE MAY BE A NEW PATTERN.
          R GET THE 'PARTIAL SYMPTOM PATTERN' FOR THIS
          R STATE GIVEN THE CURRENT SYMPTOM LIST.
          R
           INSECT.(UNACTD,STATE,LIST.(TEMP))
          R IS THIS PARTIAL PATTERN A SUBSET OF AN EXISTING PATTERN.Q.
           R=SEQRDR.(PATSTK)
CLOOP      NEXT=SEQLR.(R,F)
           W'R F.NE.1
               W'R SUBSET.(TEMP,ITSVAL.($SYMPS$,NEXT))
                   IRALST.(TEMP)
                   T'O LOOP
               O'E
                   T'O CLOOP
               E'L
           E'L
          R
          R 'TEMP' NOW CONTAINS THE PARTIAL SYMPTOM
          R PATTERN. CREATE THE STATE PRIOR FOR THIS PATTERN
          R AND ADD IT TO THE PATTERN STACK.
          R
           NULST=CONT.(NEWBOT.(LIST.(9),PATSTK)+1)
           NEWVAL.($SYMPS$,TEMP,NULST)
           IRDR=SEQRDR.(MEMLST)
INLOOP     STATE1=SEQLR.(IRDR,II)
           W'R II.E.1, T'O PROC
           SEQLR.(IRDR,II)
           W'R MEMBER.(STATE1,PATSTK,1).NE.0, T'O INLOOP
           MANY.(NULST,STATE1,ITSVAL.($PROB$,STATE1))
           T'O INLOOP
          R
          R 'NULST' NOW CONTAINS THE STATES AND A PRIORI
          R PROBABILITIES FOR THE PATTERN IN 'TEMP'.
          R UPDATE THIS PRIOR BASED ON THE SYMPTOMS IN 'TEMP'.
          R
PROC       P=0.
```

```
          PRDR=SEQRDR.(TEMP)
PLOOP     SYMP1=SEQLR.(PRDR,PI)
          W'R PI.NE.1
             UPD1.(SYMP1,NULST,NULST)
             T'O PLOOP
          E'L
          IRALST.(TEMP)
          T'O LOOP
          E'N


   UPD    MAD
          EXTERNAL FUNCTION (SYMP)
          R
          R THIS FUNCTION SUPERVISES THE UPDATING
          R OF THE PATTERN STACK GIVEN THE NEW 'SYMP'.
          R EACH OF THE STATE PRIOR LISTS IN THE STACK
          R IS UPDATED (PROVIDED THAT THE 'SYMP' IS
          R RELEVANT TO SOME STATE IN THE LIST).
          R WHENEVER THE PROBABILITY OF A PATTERN GOES
          R TO ZERO, THE PATTERN IS DELETED FROM THE
          R PATTERN STACK.
          R
          N'R
          B'N LEMPTY, RELEV
          INSERT FILE COMMON
          F'T P,UPD1
          E'O UPD.
          NEWBOT.(SYMP,UNACTD)
          RDR=SEQRDR.(PATSTK)
LOOP      STLST=.ABS.SEQLR.(RDR,I)
CHECK     W'R I.E.1, T'O FINISH
          R
          R CHECK THE RELEVANCE OF THE SYMPTOM TO
          R THE PRIOR IN STLST.
          R
          W'R .NOT. RELEV.(SYMP,STLST)
             W'R SYMP.L.0, NEWBOT.(SYMP,ITSVAL.($SYMPS$,STLST))
             T'O LOOP
          E'L
          R
          R THE SYMPTOM IS RELEVANT. USE IT TO
          R UPDATE THE PRIOR.
          R
          P=UPD1.(SYMP,STLST,STLST)
          W'R P.E.0
             W'R STLST.L.0, UNDO.(STLST)
             ADD=LPNTR.(RDR)
             STLST=SEQLR.(RDR,I)
             SYMPS=ITSVAL.($SYMPS$,REMOVE.(ADD))
PLOOP        W'R LEMPTY.(SYMPS), T'O CHECK
             TSYMP=POPTOP.(SYMPS)
             W'R TSYMP.G.0, PATFRM.(TSYMP)
             T'O PLOOP
          O'E
             NEWVAL.($PROB$,P,STLST)
             NEWBOT.(SYMP,ITSVAL.($SYMPS$,STLST))
             T'O LOOP
          E'L
```

```
          R
          R HERE THE SYMPTOM IS PROCESSED BY 'PATFRM'
          R TO SEE IF ANY NEW PATTERNS CAN BE FORMED.
          R
FINISH     W'R SYMP.G.0, PATFRM.(SYMP)
           F'N
           E'N



 UPD1   MAD
           EXTERNAL FUNCTION (SYMP,LST1,LST2)
          R
          R THIS FUNCTION UPDATES THE STATE PRIOR
          R IN 'LST1' TO ACCOUNT FOR THE NEW
          R SYMPTOM 'SYMP'. THE SIGN OF 'SYMP' DENOTES
          R THE PRESENCE OR ABSENCE OF 'SYMP'.
          R 'LST2' IS WHERE THE UPDATED PRIOR IS STORED.
          R
           N'R
           V'S EPSI=1.E-4
           INSERT FILE COMMON
           F'I P,PIJ,EPSI,PR,PROB
           EQUIVALENCE (IPROB,PROB)
           B'N SAME
           E'O UPD1.
           W'R LST1.E.LST2
              SAME=1B
           O'E
              SAME=0B
           E'L
           P=0.
           MEMLST=ITSVAL.($MEMBER$,SYMP)
          R
          R PROCESS EACH STATE ON THE MEMBER LIST OF 'SYMP'.
          R
           RDR=SEQRDR.(LST1)
LOOP       STATE=SEQLR.(RDR,I)
CHECK      W'R I.E.1
              W'R P.L.EPSI, F'N 0.
              RDR=LRDROV.(LST2)
AGAIN         IPROB=ADVLER.(RDR,I)
              W'R I.E.1, F'N P
              ADD=LPNTR.(RDR)
              SUBST.(PROB/P,ADD)
              T'O AGAIN
           E'L
           IPROB=SEQLR.(RDR,I)
           LOC=MEMBER.(STATE,MEMLST,0)
           W'R LOC.E.0
              PR=0.
           O'E
              PR=PIJ.(SYMP,CONT.(LNKR.(CONT.(LOC))+1))
           E'L
           W'R SYMP.L.0
              PROB=PROB*(1.-PR)
           O'E
              PROB=PROB*PR
           E'L
          R
```

```
          R CHECK FOR 'ZERO' POSTERIOR FOR THIS
          R STATE. IF ZERO, DELETE IT FROM THE LIST.
          R
           W'R PROB.L.EPSI, T'O SGRAP
           P=P+PROB
           W'R SAME
              ADD=LPNTR.(RDR)
              SUBST.(PROB,ADD)
           O'E
              MANY.(LST2,STATE,PROB)
           E'L
           T'O LOOP
          R
          R HERE IS WHERE A STATE IS REMOVED FROM 'LST'
          R
SGRAP     W'R .NOT.SAME, T'O LOOP
           ADD=LPNTR.(RDR)
           ADD1=LNKL.(CONT.(ADD))
           STATE=SEQLR.(RDR,I)
           REMOVE.(ADD)
           REMOVE.(ADD1)
           T'O CHECK
           E'N


  PIJ   MAD
           EXTERNAL FUNCTION (SYMP,CLUSTR)
          N'R
           INSERT FILE COMMON
          B'N LEMPTY,NAMTST
          F'T P1,P2
          EQUIVALENCE (P1,IP1),(P2,IP2)
          E'O PIJ.
          R
          R THIS FUNCTION OBTAINS THE PROBABILITY OF SYMPTOM
          R 'SYMP'. 'CLUSTR' IS EITHER THIS PROBABILITY OR
          R THE NAME OF A CLUSTER WHICH CONTAINS 'SYMP'.
          R
           W'R  NAMTST.(CLUSTR), F'N CLUSTR
          LIST.(TEMP)
           LIST.(OPSTCK)
          RDR=SEQRDR.(CLUSTR)
LOOP      NEXT=SEQLR.(RDR,I)
          W'R T.E.1
          R
          R SHOULD NEVER GET HERE
          R
              F'N -1.
           O'R NEXT.E.LPAREN
              T'O LOOP
           O'R NEXT.E.RPAREN
          R
          R END OF A TRIPLE, PROCESS OPERATOR AGAINST 'TEMP'
          R
              IP1=POPTOP.(TEMP)
              FIRST=POPTOP.(TEMP)
              W'R LEMPTY.(OPSTCK)
          R
          R END OF THE EVALUATION
```

```
      R
            IRALST.(TEMP)
            IRALST.(OPSTCK)
            E'N P1
          E'L
          IP2=POPTOP.(TEMP)
          SECOND=POPTOP.(TEMP)
          OPER=POPTOP.(OPSTCK)
      R
      R PROCESS OPERATOR HERE
      R
          W'R OPER.E.$OR$
            W'R FIRST.E.1.OR.SECOND.E.1
              P1=P1+P2
              BMARK=1
            O'E
              P1=0.
              BMARK=0
            E'L
          O'E
            BMARK=FIRST+SECOND
            W'R BMARK.G.1
              P1=0.
              BMARK=0
            O'E
              P1=P1+P2
            E'L
          E'L
      R
      R CHECK FOR AN OPERATOR HERE
      R
       O'R NEXT.E.$OR$.OR.NEXT.E.$EXOR$
          NEWTOP.(NEXT,OPSTCK)
          I'O LOOP
      R
      R PROCESS SUBCLUSTER HERE
      R
       O'E
          BMARK=INTERP.(NEXT)
          W'R SYMCNT.E.0
            P1=0.
            BMARK=0
          O'R BMARK.E.0
            P1=0.
          O'R MEMBER.(SYMP,NEXT,0).NE.0
            W'R SYMCNT.E.1
              IP1=ITSVAL.($PROB$,NEXT)
            O'E
              P1=1.
            E'L
          O'E
            P1=0.
          E'L
       E'L
       MANY.(TEMP,P1,BMARK)
       I'O LOOP
       V'S LPAREN=$($
       V'S RPAREN=$)$
       E'N
```

```
NSCOMP    MAD
          EXTERNAL FUNCTION (TEST,PRIOR,LST)
          N'R
          F'T P,UPD1
          E'O NSCOMP.
          R=SEQRDR.(ITSVAL.($MEMBER$,TEST))
LOOP      SYMP=SEQLR.(R,F)
          W'R F.NE.1
             P=UPD1.(-SYMP,PRIOR,LST)
             W'R P.G.0., T'O LOOP
          E'L
          F'N P
          E'N


DEFINE    MAD
          EXTERNAL FUNCTION (TEMP)
          N'R
          B'N LEMPTY, OPER
          INSERT FILE COMMON
          E'O DEFINE.
          LST=0
          LIST.(RDRSTK)
          NAME=POPTOP.(TEMP)
          NUM=UFUNC(0)
          T'H LOOK, FOR J=1,2,J.G.NUM
          W'R UFUNC(J).NE.NAME, T'O LOOK
             W'R UFUNC(J+1).NE.0
             PRINT COMMENT $RELATION ALREADY DEFINED. REPLACE.Q.$
             R'T $C6*$,ANS
             W'R ANS.NE.$YES$, T'O DONE
             LST=LIST.(UFUNC(J+1))
             T'O START
             E'L
LOOK      CONTINUE
          NUM=NUM+2
          UFUNC(NUM)=NAME
          LST=LIST.(UFUNC(NUM+1))
          PRINT OCTAL RESULTS LST
          UFUNC(0)=NUM
START     PCOUNT=0
          OPER=18
          ARGLST=POPTOP.(TEMP)
          RDR=SEQRDR.(TEMP)
LOOP      ELEM=SEQLR.(RDR,I)
          W'R I.E.1
             PCOUNT=PCOUNT-1
             W'R LEMPTY.(RDRSTK)
               W'R PCOUNT.G.0
                PRINT COMMENT $NOT WELL FORMED. TRY AGAIN.$
                UFUNC(J+1)=0
               E'L
               T'O DONE
             O'E
               NEWBOT.($)$, LST)
               RDR=POPTOP.(RDRSTK)
               T'O LOOP
```

```
            E'L
        O'R I.L.0
            W'R OPER
              OPER=08
              W'R ELEM.E.$QUOTE$
                ELEM=SEQLR.(RDR,I)
                T'O CDEF
            E'L
            CODE=ARGTST.(0)
            W'R CODE.NE.0, ELEM=CODE.V.54K10
          NEWBOT.(ELEM,LST)
            O'E
              CODE=ARGTST.(0)
              W'R CODE.NE.0
            ELEM=CODE.V.54K10
            NEWBOT.(ELEM,LST)
              O'E
CDEF            CNUM=CONST(0)
                CNUM=CNUM+1
                CONST(CNUM)=ELEM
                CONST(0)=CNUM
                NEWBOT.(CNUM,LST)
              E'L
            E'L
      O'E
        OPER=1B
        NEWTOP.(RDR,RDRSTK)
        RDR=SEQRDR.(ELEM)
        NEWBOT.($($,LST)
        PCOUNT=PCOUNT+1
      E'L
      T'O LOOP
    R
    R
     INTERNAL FUNCTION (DUMMY)
    R
     E'O ARGTST.
     ARDR=SEQRDR.(ARGLST)
     ACOUNT=1
ALOOP ATEMP=SEQLR.(ARDR,AI)
     W'R AI.E.1, F'N 0
     W'R ATEMP.E.ELEM, F'N ACOUNT
     ACOUNT=ACOUNT+1
     T'O ALOOP
     E'N
    R
    R
DONE  IRALST.(RDRSTK)
     F'N LST
     E'N


CLUSTR  MAD
     EXTERNAL FUNCTION (LST)
     N'R
     B'N LEMPTY
     F'T P, PSAVE
     EQUIVALENCE (IP,P)
     E'O CLUSTR.
```

```
          LIST.(TEMP)
          STATE=TRANS.(POPTOP.(LST),1)
          W'R STATE.E.0, T'O ERR
          NULST=CONT.(NEWBOT.(LIST.(9),STATE)+1)
          NEWVAL.($RELAT$,$CLUSTR$,NULST)
          RDR=SEQRDR.(LST)
          NEWBOT.(LPAREN,NULST)
          SUB=0
LOOP      IP=SEQLR.(RDR,I)
          W'R I.E.1
             NEWBOT.(RPAREN,NULST)
             W'R LEMPTY.(TEMP)
               IRALST.(TEMP)
               F'N NULST
             E'L
             RDR=POPTOP.(TEMP)
             SUB=0
          O'R I.E.0
             W'R SUB.E.1
               DO1.(IP)
             O'E
               NEWTOP.(RDR,TEMP)
               RDR=SEQRDR.(IP)
             NEWBOT.(LPAREN,NULST)
             E'L
          O'R IP.E.$OR$.OR.IP.E.$EXOR$
             NEWBOT.(IP,NULST)
          O'E
             SUB=1
             PSAVE=P
          E'L
          T'O LOOP
        R
        R
          INTERNAL FUNCTION (SLST)
          E'O DO1.
          SUBLST=CONT.(NEWBOT.(LIST.(9),NULST)+1)
          NEWVAL.($PROB$,PSAVE,SUBLST)
          NEWVAL.($RELAT$,POPTOP.(SLST),SUBLST)
          DRDR=SEQRDR.(SLST)
DLOOP     NEXT=SEQLR.(DRDR,DI)
          W'R DI.E.1, F'N
          NEXT=TRANS.(NEXT,2)
          W'R NEXT.E.0, T'O ERR
          LOC=MEMBER.(NEXT,STATE,0)
          REMOVE.(LOC)
          LOC=MEMBER.(STATE,ITSVAL.($MEMBER$,NEXT),0)
          ADD=LNKR.(CONT.(LOC))
          SUBST.(NULST,ADD)
          NEWBOT.(NEXT,SUBLST)
          T'O DLOOP
          E'N
        R
        R
ERR       PRINT COMMENT $ERROR IN FORMAT$
          F'N -1
          V'S LPAREN=$($
          V'S RPAREN=$)$
          E'N
```

```
INTERP    MAD
          EXTERNAL FUNCTION (CLUSTR)
          N'R
          INSERT FILE COMMON
          E'O INTERP.
          SYMCNT=0
          BASE=0
          SPOINT=0
          STACK(0)=0
          RDR=SEQRDR.(CLUSTR)
LOOP      ELEM=SEQLR.(RDR,I)
          W'R I.E.1
             PUSH.(ITSVAL.($RELAT$,CLUSTR))
             CODE=EVAL.(0)
             W'R STACK(0).E.$*INC*$
                STACK(0)=1
             O'R CODE.L.0.OR.STACK(0).E.$FALSE$
                STACK(0)=0
             E'L
             F'N STACK(0)
          O'E
             PUSH.(ELEM)
             T'O LOOP
          E'L
          E'N


EVAL      MAD
          EXTERNAL FUNCTION (DUMMY)
          N'R
          INSERT FILE COMMON
          E'O EVAL.
          POP.(NAME)
          R
          R CHECK FOR PRIMITIVE
          R
          T'H LOOK, FOR J=1,1,J.G.NPRIM(0)
          W'R NAME.NE.NPRIM(J), T'O LOOK
             CODE=PRIM(J).(0)
             F'N CODE
LOOK      CONTINUE
          R
          R USER DEFINED FUNCTION
          R
          T'H ULOOK, FOR J=1,2,J.G.UFUNC(0)
          W'R UFUNC(J).NE.NAME, T'O ULOOK
             LST=UFUNC(J+1)
             W'R LST.E.0, T'O ERR
             T'O PROC
ULOOK     CONTINUE
ERR       P'T ERRM, NAME
          V'S ERRM=$C6,H/ NOT DEFINED./*$
          F'N -1
PROC      RDR=SEQRDR.(LST)
LOOP      NEXT=SEQLR.(RDR,I)
          W'R I.E.1
             POP.(NEXT)
```

```
            RETAC.(NEXT)
            F'N STACK(0)
         O'R NEXT.E.$($
            PUSH.(0)
            NEWTOP.(SPOINT,OPSTCK)
            NEXT=SEQLR.(RDR,I)
            NEWTOP.(NEXT,OPSTCK)
         O'R NEXT.E.$)$
            NEXT=POPTOP.(OPSTCK)
            W'R NEXT.A.77K10.E.54K10
              ARGF.(NEXT)
              T'O KEEP
            E'L
            PUSH.(NEXT)
KEEP        SAVE DATA RDR, BASE
            SAVE RETURN
            BASE=POPTOP.(OPSTCK)
            CODE=EVAL.(0)
            RESTORE RETURN
            RESTORE DATA BASE, RDR
            W'R CODE.L.0.OR.CODE.E.$FALSE$, F'N CODE
         O'R NEXT.A.77K10.E.54K10
            ARGF.(NEXT)
      O'E
            PUSH.(CONST(NEXT))
      E'L
      T'O LOOP
      E'N


CONTRL   MAD
         EXTERNAL FUNCTION (SPOT)
         N'R
         INSERT FILE COMMON
         E'O RETAC.
         STACK(BASE)=SPOT
         SPOINT=BASE
         F'N
         E'O PUSH.
         SPOINT=SPOINT+1
         STACK(SPOINT)=SPOT
         F'N
         E'O POP.
         SPOT=STACK(SPOINT)
         SPOINT=SPOINT-1
         F'N
         E'N


 APRIM   MAD
         EXTERNAL FUNCTION (DUMMY)
         N'R
         STATEMENT LABEL X
         B'N FIRST, SECOND,ACHECK,8V
         V'S NRMBIT=4000000000K
         INSERT FILE COMMON
         E'O L.
```

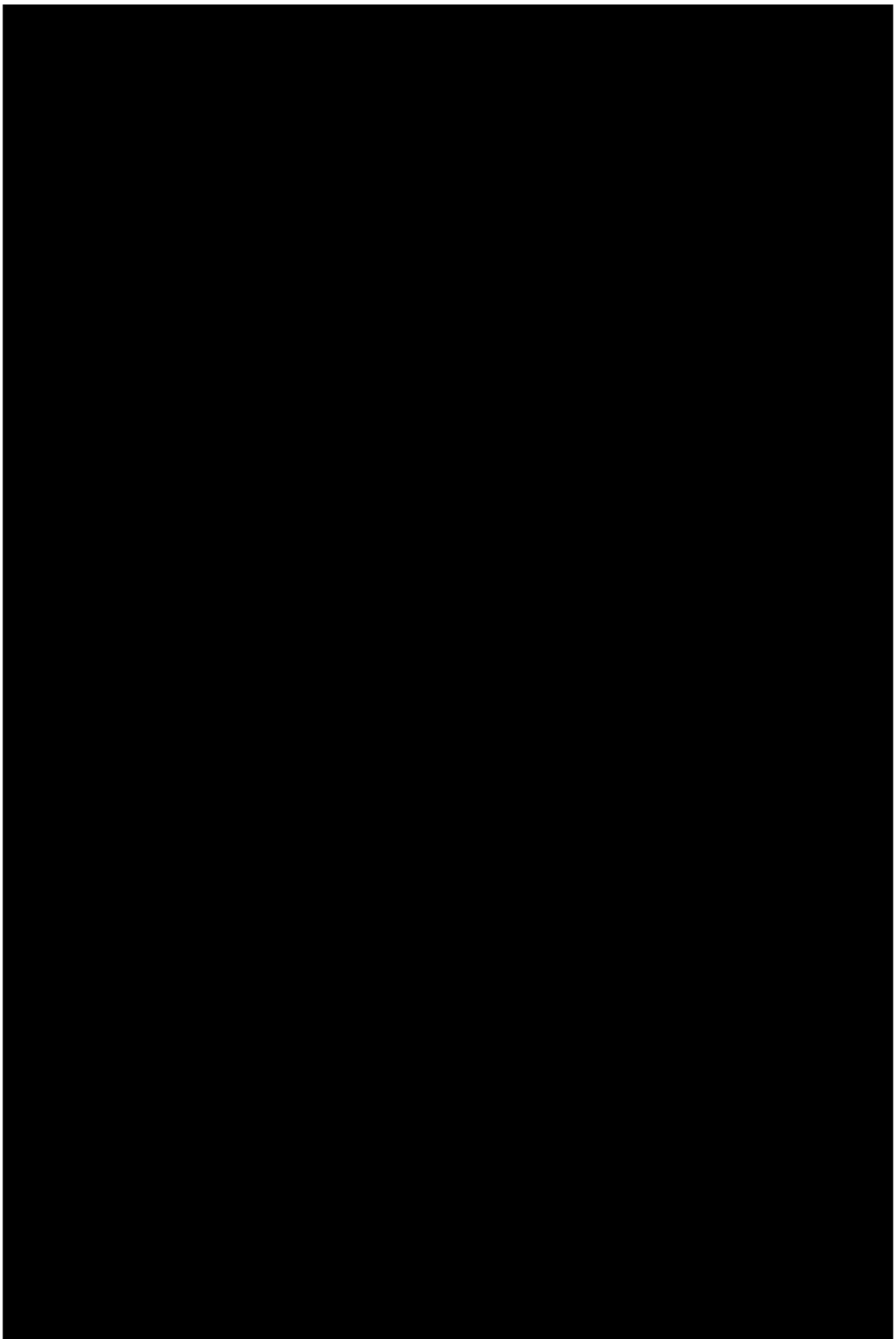
```
            W'R ACHECK.(BSTORE)
               BV=FTEMP2.L.FTEMP1
            O'E
               BV=TEMP2.L.TEMP1
            E'L
            T'O BSTORE
            E'O LE.
            W'R ACHECK.(BSTORE)
               BV=FTEMP2.LE.FTEMP1
            O'E
               BV=TEMP2.LE.TEMP1
            E'L
            T'O BSTORE
            E'O EQ.
            W'R ACHECK.(BSTORE)
               BV=FTEMP1.E.FTEMP2
            O'E
               BV=TEMP1.E.TEMP2
            E'L
            T'O BSTORE
            E'O GE.
            W'R ACHECK.(BSTORE)
               BV=FTEMP2.GE.FTEMP1
            O'E
               BV=TEMP2.GE.TEMP1
            E'L
            T'O BSTORE
            E'O G.
            W'R ACHECK.(BSTORE)
               BV=FTEMP2.G.FTEMP1
            O'E
               BV=TEMP2.G.TEMP1
            E'L
BSTORE      RETAC.(BV)
            F'N
            E'O PLUS.
            W'R ACHECK.(BACK)
               FTEMP1=FTEMP1+FTEMP2
            O'E
               TEMP1=TEMP1+TEMP2
            E'L
            T'O BACK
            E'O MINUS.
            W'R ACHECK.(BACK)
               FTEMP1=FTEMP2-FTEMP1
            O'E
               TEMP1=TEMP2-TEMP1
            E'L
            T'O BACK
            E'O TIMES.
            W'R ACHECK.(BACK)
               FTEMP1=FTEMP1*FTEMP2
            O'E
               TEMP1=TEMP1*TEMP2
            E'L
            T'O BACK
            E'O DIVIDE.
            W'R ACHECK.(BACK)
               FTEMP1=FTEMP2/FTEMP1
            O'E
               TEMP1=TEMP2/TEMP1
```

```
          E'L
BACK      RETAC.(TEMP1)
          F'N 0
          R
          R
          INTERNAL FUNCTION (X)
          E'O ACHECK.
          POP.(TEMP1)
          POP.(TEMP2)
          W'R TEMP1.E.$*INC**$.OR.TEMP2.E.$*INC**$
             TEMP1=$*INC**$
             BV=1B
             T'O X
          E'L
          W'R TEMP1.A.NRMBIT.E.0
             FIRST=0B
          O'E
             FIRST=1B
          E'L
          W'R TEMP2.A.NRMBIT.E.0
             SECOND=0B
          O'E
             SECOND=1B
          E'L
          W'R FIRST.AND.SECOND
             F'N 1B
          O'R FIRST.EQV.SECOND
             F'N 0B
          O'R FIRST
             FTEMP2=TEMP2
          O'E
             FTEMP1=TEMP1
          E'L
          F'N 1B
          E'N
          E'N


LPRIM     MAD
          EXTERNAL FUNCTION (DUMMY)
          N'R
          INSERT FILE COMMON
          B'N TEMP1,TEMP2,BTEST,BV
          E'O AND.
          W'R .NOT.BTEST.(0), F'N -1
          BV=TEMP1.AND.TEMP2
          T'O STORE
          E'O OR.
          W'R .NOT.BTEST.(0), F'N -1
          BV=TEMP1.OR.TEMP2
          T'O STORE
          E'O EQV.
          W'R .NOT.BTEST.(0), F'N -1
          BV=TEMP1.EQV.TEMP2
          T'O STORE
          E'O NOT.
          POP.(TEMP1)
          BV=.NOT.TEMP1
          T'O STORE
```

```
STORE       RETAC.(BV)
            F'N 0
           R
           R
            INTERNAL FUNCTION (X)
            E'0 BTEST.
            T1=STACK(SPOINT)
            T2=STACK(SPOINT-1)
            W'R T1.E.0.OR.T1.E.1
               POP.(TEMP1)
            0'R T1.E.$*INC*$
               POP.(TEMP1)
               TEMP1=1B
            0'E
               F'N 0B
            E'L
NEXT        W'R T2.E.0.OR.T2.E.1
               POP.(TEMP2)
            0'R T2.E.$*INC*$
               POP.(TEMP2)
               TEMP2=1B
            0'E
               F'N 0B
            E'L
            F'N 1B
            E'N
            E'N


SYMATS   MAD
            EXTERNAL FUNCTION (DUMMY)
            N'R
            B'N BV
            INSERT FILE COMMON
            E'0 PRES.
            POP.(TEMP)
            LOC=MEMBER.(TEMP,SYMLST,0)
            W'R LOC.E.0
               RETAC.(MARK)
            0'E
               BV=TEMP.G.0
               SYMCNT=SYMCNT+1
               RETAC.(BV)
            E'L
            F'N
            V'S MARK=$*INC*$
           R
           R
            E'0 ATTRIB.
            CODE=0
            LIST.(LST)
            POP.(SYMP)
            POP.(ATT)
            W'R SYMP.L.0
               CODE=$FALSE$
               VAL=$FALSE$
               T'0 RET
            E'L
            LOC=MEMBER.(SYMP,SYMLST,0)
```



```
            CHECKP.(SYMP)
            T'O INLOOP
           R
           R HERE THE 'NORMAL' RESULT OF A TEST IS
           R PROCESSED.
           R
NCOMP       LIST.(SAVE)
            P=NSCOMP.(TEST,PRIOR,SAVE)
            CHECKP.($NORM$)
            IRALST.(RLST)
            F'N RLST
           R
           R
            INTERNAL FUNCTION (MARK)
            E'O CHECKP.
            W'R P.G.0
            LIST.(SCRAT)
            MANY.(SCRAT,$PROB$,P,$PRIOR$,SAVE,$RESULT$,MARK)
            NEWBOT.(LIST.(9),RLST)
            MAKEDL.(SCRAT,BOT.(RLST))
            IRALST.(SCRAT)
            E'L
            IRALST.(SAVE)
            F'N
            E'N
            E'N



RELTST    MAD
            EXTERNAL FUNCTION (LST,PRIOR)
            N'R
            INSERT FILE COMMON
            B'N LEMPTY
            STATEMENT LABEL SWITCH
            F'T P,THRESH
            EQUIVALENCE (IP,P)
           R
           R THIS FUNCTION DETERMINES ALL THE TESTS WHICH
           R ARE RELEVANT TO THE CURRENT STATE LIST OF 'PRIOR'.
           R TESTS WHICH HAVE ALREADY BEEN RUN ARE IGNORED.
           R
            E'O RELTST.
            COUNT=0
            LIST.(RDRSTK)
            W'R THRESH.G.1.
               SWITCH=RET
               STATE=0
               DLOSS.(PRIOR,STATE)
               T'O DO1
            O'E
               SWITCH=LOOP
            E'L
            SR=SEQRDR.(PRIOR)
LOOP        STATE=SEQLR.(SR,SI)
            W'R SI.E.1
RET            IRALST.(RDRSTK)
               F'N COUNT
            E'L
            IP=SEQLR.(SR,SI)
```

```
            W'R P.LE.THRESH, T'O LOOP
DO1         SYR=SEQRDR.(STATE)
INLOOP      SYMP=SEQLR.(SYR,SYI)
            W'R SYI.E.1
               W'R .NOT.LEMPTY.(RDRSTK)
                  SYR=POPTOP.(RDRSTK)
                  T'O INLOOP
               O'E
                  T'O SWITCH
               E'L
            O'R SYI.L.0
               T'O INLOOP
            O'R ITSVAL.($RELAT$,SYMP).NE.0
               NEWTOP.(SYR,RDRSTK)
               SYR=SEQRDR.(SYMP)
               T'O INLOOP
            O'E
               TEST=BOT.(SYMP)
               W'R MEMBER.(TEST,CELL(1),1).NE.0, T'O INLOOP
               W'R MEMBER.(TEST,LST,0).NE.0, T'O INLOOP
               W'R MEMBER.(TEST,TSTRUN,0).NE.0, T'O INLOOP
               NEWBOT.(TEST,LST)
               COUNT=COUNT+1
               T'O INLOOP
            E'L
            E'N


  TOPT   MAD
            EXTERNAL FUNCTION (ATEST,STATE)
            N'R
            B'N LEMPTY
            INSERT FILE COMMON
            F'T LSAVE,DSAVE,LS
            EQUIVALENCE (ILSAVE,LSAVE),(ILS,LS)
            E'O TOPTH.
            SWITCH=1
            LIST.(NOGOOD)
            T'O START
            E'O TOPT.
            SWITCH=2
START       R=SEQRDR.((ITSVAL.($VALUES$,TREE)))
            RET=0
            STATE=SEQLR.(R,F)
            ILSAVE=SEQLR.(R,F)
            DSAVE=LSAVE
            ADD=LPNTR.(R)
LOOP        TEST=SEQLR.(R,F)
            W'R F.E.1, T'O END(SWITCH)
            ILS=SEQLR.(R,F)
            W'R LS.G.DSAVE
               T'O SAVE(SWITCH)
            O'R LS.L.LSAVE
               STATE=0
               LSAVE=LS
               ATEST=TEST
               ADD=LPNTR.(R)
            E'L
SAVE(2)     T'O LOOP
```

```
SAVE(1)     NEWTOP.(TEST,NOGOOD)
            T'O LOOP
END(1)      W'R .NOT. LEMPTY.(NOGOOD)
               RET=1
               NEWTOP.(NOGOOD,CELL(1))
            E'L
            IRALST.(NOGOOD)
END(2)      REMOVE.(LNKL.(CONT.(ADD)))
            REMOVE.(ADD)
            FIN RET
            E'N



 SETUP   MAD
            EXTERNAL FUNCTION (N1,N2)
            N'R
            F'T PROB
            EQUIVALENCE (IPROB,PROB)
            B'N LEMPTY,NAMTST
            INSERT FILE COMMON
            E'O SETUP.
            MANY.(STRUCT,LIST.(STLIST),LIST.(SYMS),LIST.(TESTS))
            NUM=ITSVAL.($HSHNUM$,STRUCT)
            T'H HL, FOR J=1,1,J.G.2.P.NUM
            NEWBOT.(LIST.(9),STLIST)
            NEWBOT.(LIST.(9),SYMS)
            NEWBOT.(LIST.(9),TESTS)
HL          CONTINUE
            LIST.(TEMP)
LOOP        W'R DSKLST.(N1,N2,TEMP).E.$DONE$, T'O ENDO
            WORD=POPTOP.(TEMP)
            W'R WORD.E.$STATE$
               T'O ST
            O'R WORD.E.$SYMPS$
               T'O SYMPL
            O'R WORD .E.$TESTS$
               T'O TESTL
            O'R WORD.E.$DEFINE$
               DEFINE.(TEMP)
            O'R WORD.E.$SPTEST$
               T'O SPTST
            O'R WORD.E.$EXCLUS$
               T'O EXL
            O'E
               PRINT COMMENT $ERROR IN BCD TAPE$
               CHNCOM.(0)
            E'L
            R
            R
ST          NAME=POPTOP.(TEMP)
            STATE=LOOKUP.(NAME,1)
            NEWVAL.($PROB$,POPTOP.(TEMP),STATE)
STLOOP      W'R LEMPTY.(TEMP), T'O LOOP
            SYMP=POPTOP.(TEMP)
            IPROB=POPTOP.(TEMP)
            W'R PROB.E.0., T'O STLOOP
            SYMP=LOOKUP.(SYMP,2)
            MANY.(ITSVAL.($MEMBER$,SYMP),STATE,PROB)
            NEWBOT.(SYMP,STATE)
```

```
            T'O STLOOP
            R
            R
SYMPL       W'R LEMPTY.(TEMP), T'O LOOP
            P'T KK, TOP.(TEMP), NTHTOP.(TEMP,2)
            V'S KK=$2(1$,K12)*$
            NAME=TOP.(TEMP)
            TEST=LOOKUP.(NTHTOP.(TEMP,2),3)
            W'R NAMTST.(NAME)
            SYMP=LOOKUP.(NAME,2)
            NEWBOT.(SYMP,ITSVAL.($MEMBER$,TEST))
            NEWBOT.(TEST,SYMP)
            T'O SYMPL1
            E'L
            MEMLST=ITSVAL.($MEMBER$,TEST)
            R=SEQRDR.(NAME)
RLOOP       SYMP=SEQLR.(R,F)
            W'R F.E.1, T'O SYMPL1
            SYMP=LOOKUP.(SYMP,2)
            NEWBOT.(TEST,SYMP)
            NEWBOT.(SYMP,MEMLST)
            T'O RLOOP
TESTL       W'R LEMPTY.(TEMP), T'O LOOP
            NAME=TOP.(TEMP)
            COST=NTHTOP.(TEMP,2)
            W'R NAMTST.(NAME)
            NEWBOT.(COST,LOOKUP.(NAME,3))
            T'O TESTL1
            E'L
            R=SEQRDR.(NAME)
TLOOP       NEXT=SEQLR.(R,F)
            W'R F.E.1, T'O TESTL1
            NEWBOT.(COST,LOOKUP.(NEXT,3))
            T'O TLOOP
SPTST       W'R LEMPTY.(TEMP), T'O LOOP
            TEST=LOOKUP.(POPTOP.(TEMP),3)
            NEWVAL.($SPTEST$,$YES$,TEST)
            T'O SPTST
ENDO        IRALST.(TEMP)
            F'N
SYMPL1      POPTOP.(TEMP)
            POPTOP.(TEMP)
            T'O SYMPL
TESTL1      POPTOP.(TEMP)
            POPTOP.(TEMP)
            T'O TESTL
EXL         W'R LEMPTY.(TEMP), T'O LOOP
            T1=TRANS.(POPTOP.(TEMP),3)
            T2=TRANS.(POPTOP.(TEMP),3)
            NEWVAL.($EXCLUS$,T1,T2)
            NEWVAL.($EXCLUS$,T2,T1)
            T'O EXL
            E'N


LOOKUP  MAD

            EXTERNAL FUNCTION (WORD,LCODE)
            N'R
```

```
            INSERT FILE COMMON
            E'0 LOOKUP.
            HLIST=0
            ADD=LOCATE.(0)
            W'R ADD.E.0
            NEWBOT.(LIST.(9),HLIST)
            LST=BOT.(HLIST)
            W'R LCODE.G.1, NEWVAL.($MEMBER$,LIST.(9),LST)
            NEWVAL.($PNAME$,WORD,LST)
            E'L
            F'N LST
            E'0 TRANS.
            F'N LOCATE.(0)
            INTERNAL FUNCTION (X)
            E'0 LOCATE.
            HSHNUM=ITSVAL.($HSHNUM$,STRUCT)
            HLIST=NTHTOP.(NTHTOP.(STRUCT,LCODE),HASH.(WORD,HSHNUM)+1)
            RDR=SEQRDR.(HLIST)
LOOP        LST=SEQLR.(RDR,I)
            W'R I.E.1, F'N 0
            W'R ITSVAL.($PNAME$,LST).E.WORD, F'N LST
            T'0 LOOP
            E'N
            E'N



INSECT      MAD
            EXTERNAL FUNCTION (L1,L2,L3)
              N'R
            B'N LEMPTY
            E'0 INSECT.
            R
            R THIS FUNCTION DETERMINES THE INTERSECTION
            R OF L1 AND L2 AND PLACES THE ANSWER IN L3.
            R
            RDR=SEQRDR.(L1)
            LIST.(RDRSTK)
LOOP        ELEM=SEQLR.(RDR,I)
            W'R I.E.1
               W'R LEMPTY.(RDRSTK)
                 IRALST.(RDRSTK)
                 F'N
               O'E
                 RDR=POPTOP.(RDRSTK)
               E'L
            O'R I.L.0
            O'R ITSVAL.($RELAT$,ELEM).NE.0
               NEWTOP.(RDR,RDRSTK)
               RDR=SEQRDR.(ELEM)
            O'R MEMBER.(.ABS.ELEM,L2,0).NE.0
               NEWBOT.(ELEM,L3)
            E'L
            T'0 LOOP
            E'N



MEMBER      MAD 12/12/66 2321.6       92          00000
```

```
            EXTERNAL FUNCTION (GOAL,LST,LEVEL)
            N'R
            E'0 MEMBER.
            ADD=0
            RDR=LRDROV.(LST)
DESCND      NAME=ADVSWR.(RDR,I)
            W'R I.E.1, T'0 RETURN
            W'R LCNTR.(RDR).L.LEVEL, T'0 DESCND
COMPAR      W'R NAME.E.GOAL, T'0 FOUND
            NAME=ADVLWR.(RDR,I)
            W'R I.NE.1, T'0 COMPAR
ASCEND      W'R LCNTR.(RDR).E.0, T'0 RETURN
            LVLRV1.(RDR)
            ADVLNR.(RDR,I)
            W'R I.E.1, T'0 ASCEND
            T'0 DESCND
FOUND       ADD=LSPNTR.(RDR)
RETURN      IRARDR.(RDR)
            F'N ADD
            E'N


  UNDO    MAD
            EXTERNAL FUNCTION (LST)
            N'R
            INSERT FILE COMMON
            E'0 UNDO.
            RDR=SEQRDR.(ITSVAL.($SYMPS$,LST))
LOOP        SYMP=SEQLR.(RDR,I)
            W'R I.E.1, F'N
            RDR1=SEQRDR.(PATSTK)
LOOP1       NEXT=SEQLR.(RDR1,I1)
            W'R I1.E.1
               NEWBOT.(SYMP,UNACTD)
               T'0 LOOP
            O'R MEMBER.(SYMP,ITSVAL.($SYMPS$,NEXT),0).NE.0
               T'0 LOOP
            O'E
               T'0 LOOP1
            E'L
            E'N


DSKRD9    MAD 12/01/66 2024.2    144        00000
            EXTERNAL FUNCTION (FIRST,SECOND,LST)
            N'S INTEGER
            INSERT FILE COMMON
            D'N INT(21),NAME(1),OTHER(21)
            E'0 DSKLST.
            V'S MODE=1
            T'0 START(MODE)
START(1)    NAME(0)=RJUST.(FIRST)
            NAME(1)=RJUST.(SECOND)
            BFOPEN.($R$,NAME(0),NAME(1),BUF1(432),BUF2(432),-0,ERR)
            MODE=2
START(2)    BFREAD.(NAME(0),NAME(1),INT(0) ... 1,EOF,EOFCT,ERR)
            COUNT=LNKR.(INT(0))
```

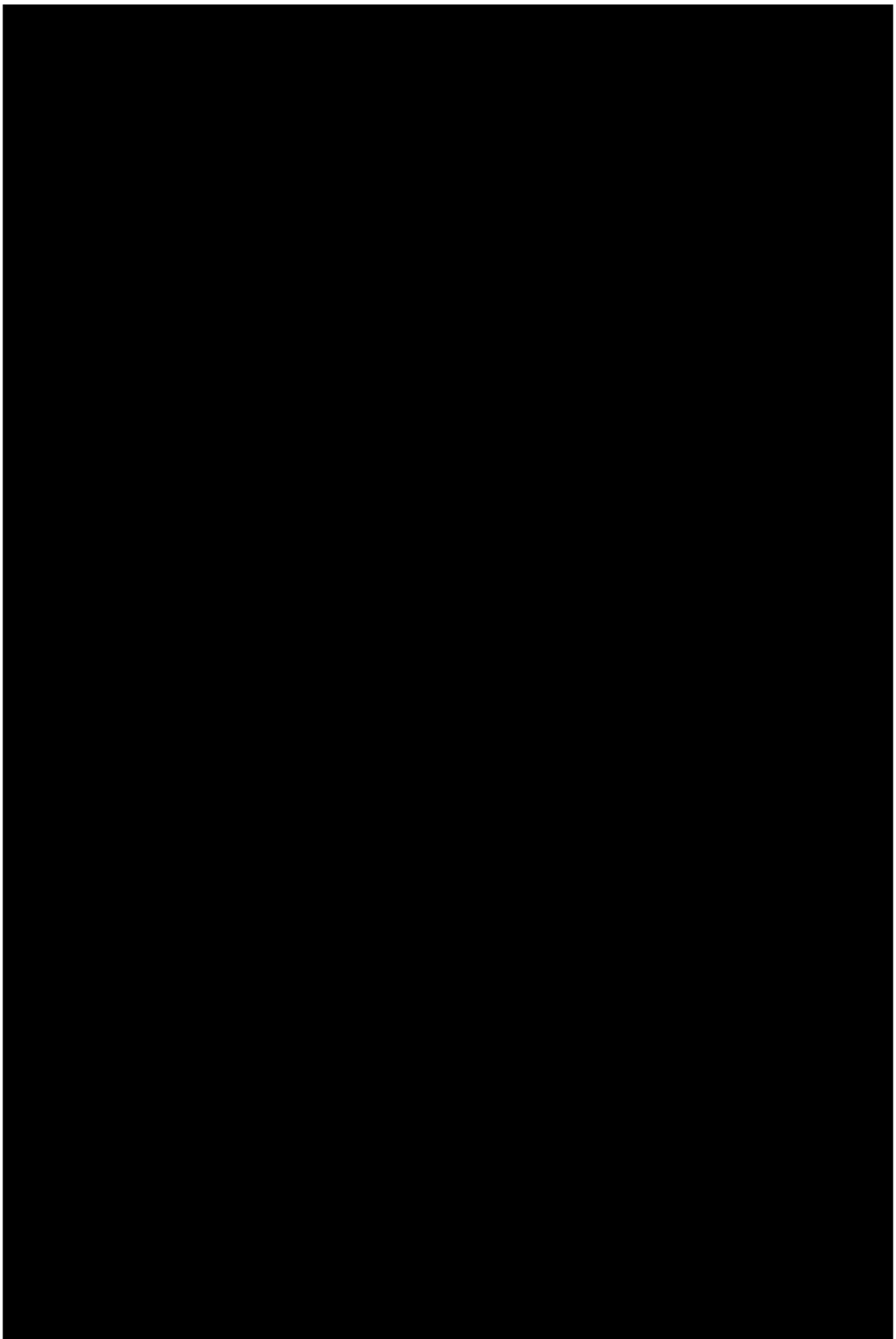
```
            BFREAD.(NAME(0),NAME(1),INT(COUNT) ... COUNT,EOF,EOFCT,ERR)
            T'H SWITCH, FOR I=COUNT,-1, I .E. 0
SWITCH      OTHER(COUNT-I)=INT(I)
            OTHER(21)=COUNT
            K=VLIST.(OTHER,LST)
            W'R K .E. $NOTYET$, T'O START(2)
            F'N K
EOF         BFCLOS.(NAME(0),NAME(1),ERR)
            MODE=1
            F'N $DONE$
ERR         PRINT COMMENT $GOOF ON READING FILE$
            MODE=1
            F'N $DONE$
            E'N



 RELEV      MAD
            EXTERNAL FUNCTION (SYMP,LST)
            N'R
            E'O RELEV.
            RDR=LRDROV.(ITSVAL.($MEMBER$,SYMP))
LOOP        TEST=ADVLNR.(RDR,1)
            W'R I.E.1, F'N OB
            W'R MEMBER.(TEST,LST,0).NE.0, F'N 1B
            T'O LOOP
            E'N



  LOSS      MAD
            EXTERNAL FUNCTION (A1,A2)
            N'R
            F'I PR,LOSS,WGT,SAVE,FCONS,WTOT,PI
            B'N LEMPTY
            EQUIVALENCE (IPR,PR),(IWGT,WGT)
            D'N PI(10),LOSS(100,AD)
            V'S AD=2,1,10
            V'S SUBS=0,0,10,20,30,40,50,60,70,80,90
            B'N LEMPTY
            INSERT FILE COMMON
            E'O SETLOS.
            RET=0
            LIST.(BUFFER)
            DSKLST.(A1,A2,BUFFER)
            SIZE=POPTOP.(BUFFER)
            DSKLST.(A1,A2,BUFFER)
            T'H LOOP, FOR J=1,1,J.G.SIZE
            W'R LEMPTY.(BUFFER), T'O ERL
            NUM=POPTOP.(BUFFER)
            NAME=TRANS.(POPTOP.(BUFFER),1)
            W'R NAME.E.0, T'O ERL
            IPR=ITSVAL.($PROB$,NAME)
            PI(NUM)=PR
LOOP        NEWVAL.($INDEX$,NUM,NAME)
            MAX=0.
            T'H LOOP1, FOR J=1,1,J.G.SIZE
            IND=SUBS(J)
            W'R DSKLST.(A1,A2,BUFFER).E.$DONE$, T'O ERL
```



```
WEIGHT   MAD
         EXTERNAL FUNCTION (LST)
         N'R
         F'T WGT,ANS,PR
         EQUIVALENCE (WGT,IWGT),(IPR,PR)
         E'O WEIGHT.
         R=SEQRDR.(LST)
         ANS=0.
LOOP     STATE=SEQLR.(R,F)
         W'R F.E.1, F'N ANS
         IPR=SEQLR.(R,F)
         IWGT=ITSVAL.($WEIGHT$,STATE)
         ANS=ANS+PR*WGT
         T'O LOOP
         E'N


  FAST   MAD
         EXTERNAL FUNCTION (CONTRL)
         N'R
         B'N LEMPTY
         INSERT FILE COMMON
         E'O FAST.
         LIST.(TEMP)
         PRINT COMMENT $YOU OR ME$
         R'T $C6*$, ANS
         W'R ANS.E.$YOU$
            CBIT=0
            PRINT COMMENT $CONTROL LIST$
            RDLONL.(TEMP)
            DEATH=PORTOP.(TEMP)
            THRESH=POPTOP.(TEMP)
            NINITS=PORTOP.(TEMP)
            NOISE=POPTOP.(TEMP)
            CONTRL=PORTOP.(TEMP)
            PRINT COMMENT $CASES$
            RDLONL.(TEMP)
            NUM=POPTOP.(TEMP)
         O'E
            CBIT=1
         E'L
         PRINT COMMENT $HISTRY FILE$
         RDLONL.(TEMP)
         W'R .NOT.LEMPTY.(TEMP)
         FILE1=RJUST.(POPTOP.(TEMP))
         FILE2=RJUST.(POPTOP.(TEMP))
         ASSIGN.(FILE1,BUF1,BUF2)
         E'L
         PRINT COMMENT $CODES$
         RDLONL.(TEMP)
         CPRIOR=POPTOP.(TEMP)
         CPAT=POPTOP.(TEMP)
         ALLPAT=POPTOP.(TEMP)
         ALLTST=POPTOP.(TEMP)
         CTEST=PORTOP.(TEMP)
         SIGNS=POPTOP.(TEMP)
         STAND=POPTOP.(TEMP)
         W'R CPRIOR.E.2.AND.CBIT.E.1
```

```
          PRINT COMMENT $DEPTH,THRESH,HEURISTIC CONTROL$
             DEPTH=POPTOP.(RDLONL.(TEMP))
             THRESH=POPTOP.(TEMP)
             CONTRL=POPTOP.(TEMP)
          E'L
          IRALST.(TEMP)
          F'N NUM
          E'N


FIRMUP    MAD
          EXTERNAL FUNCTION (PATSTK)
          N'R
          B'N SUBSET
          F'T P,PR
          EQUIVALENCE (PR,IPR)
          E'O FIRMUP.
          P=0.
          R=SEQRDR.(PATSTK)
          PATR=R
LOOP      NEXT=SEQLR.(R,F)
CHECK     W'R F.E.1, F'N P
          CURPAT=ITSVAL.($SYMPS$,NEXT)
          R1=PATR
LOOP1     CAND=SEQLR.(R1,I1)
          W'R I1.E.1
             IPR=ITSVAL.($PROB$,NEXT)
             P=P+PR
             T'O LOOP
          O'R CAND.E.NEXT
             T'O LOOP1
          O'R SUBSET.(CURPAT,ITSVAL.($SYMPS$,CAND))
             ADD=LPNTR.(R)
             NEXT=SEQLR.(R,F)
             REMOVE.(ADD)
             T'O CHECK
          O'E
             T'O LOOP1
          E'L
          E'N


SUBSET    MAD 12/26/66 1718.4      44          00000
          EXTERNAL FUNCTION (L1,L2)
          N'R
          E'O SUBSET.
          R=SEQRDR.(L1)
LOOP      NEXT=SEQLR.(R,F)
          W'R F.E.1
             F'N 18
          O'R MEMBER.(NEXT,L2,0).E.0
             F'N 08
          O'E
             T'O LOOP
          E'L
          E'N
```

```
SYMSAV    MAD
          EXTERNAL FUNCTION (SYMP,TEST)
          N'R
          INSERT FILE COMMON
          E'O SYMSAV.
          W'R MEMBER.(SYMP,SYMLST,0).NE.0, F'N
          NEWBOT.(SYMP,SYMLST)
          UPD.(SYMP)
          F'N
          E'N
```

```
COMMON   FAP
BUF1   COMMON  433
BUF2   COMMON  433
CBIT   COMMON  1
SIGNS  COMMON  1
CPAT   COMMON  1
ALLPAT COMMON  1
CPRIOR COMMON  1
CTEST  COMMON  1
ALLTST COMMON  1
DISEAS COMMON  1
STAND  COMMON  1
FILE1  COMMON  1
FILE2  COMMON  1
DEPTH  COMMON  1
THRESH COMMON  1
NINITS COMMON  1
NOISE  COMMON  1
NODES  COMMON  1
BASE   COMMON  1
TREE   COMMON  1
CURLST COMMON  1
PATLST COMMON  1
STRUCT COMMON  1
SYMCNT COMMON  1
SYMLST COMMON  1
UNACTD COMMON  1
TSTRUN COMMON  1
PATSTK COMMON  1
CODE   COMMON  1
OPSTCK COMMON  1
STACK  COMMON  21
UFUNC  COMMON  21
ARGS   COMMON  11
PRIM   COMMON  31
CONST  COMMON  31
SPOINT COMMON  1
NPRIM  COMMON  31
CELL   COMMON  21
```

```
MACROS   FAP
*
* STACK MANAGEMENT MACROS
PUSH   MACRO   ARGS
       IRP     ARGS
       TXI     *+1,1,1
       CLA     ARGS
       STO     STACK,1
       IRP
PUSH   END
*
POP    MACRO   ARGS
```

```
        IRP     ARGS
        CLA     STACK,1
        STO     ARGS
        TIX     *+1,1,1
        IRP
POP     END
*
* LIST READING MACROS HERE
*
SEQRDR  MACRO   A,B
        CLA*    A               GET LIST HEADER
        STO     B               STORE IN READER CELL
SEQRDR  END
*
SEQLR   MACRO   A,B,C
        LAC     B,4             READER LINK
        CLA     1,4             GET DATUM FOR CELL
        STO     A               SAVE DATUM
        CLA     0,4             ADVANCE READER
        STO     B
        ANA     =0700000        SET FLAG
        ARS     15
        SUB     =1
        STO     C
SEQLR   END


NTEST   FAP
        ENTRY   NAMTST
NAMTST  SXA     SV4,4
        CLA*    1,4
        STO     CAND
        TSX     $GETMEM,4
        TXH     *
        STO     LIMIT
        CLA     CAND
        SSP
        STA     LINK
        ARS     18
        CAS     LINK
        TRA     NO
        TRA     *+2
        TRA     NO
        CLA     LINK
        CAS     LIMIT
        TRA     NO
        TRA     *+1
        CLA*    LINK
        STO     HEAD
        ANA     =0700000
        CAS     =0200000
        TRA     NO
        TRA     *+2
        TRA     NO
        CLA     HEAD
        ARS     18
        CAS     LIMIT
        TRA     NO
        TRA     *+1
```

```
        STA     *+1
        CLA     **
        ANA     =077777
        CAS     LINK
        TRA     NO
        TRA     YES
NO      CLA     =1
        TRA     *+2
YES     CLA     =0
SV4     AXT     **,4
        TRA     2,4
CAND    PZE
HEAD    PZE
LINK    PZE
LIMIT   PZE
        END


   SLF   FAP
* DEPTH TO OBTAIN THE BEST TEST TO RUN. THE ROUTINE
* 'GROW1' IS USED TO GROW NEW BRANCHES ON THE TREE IF
* NECESSARY.
*
*
* STACK MANAGEMENT MACROS
PUSH    MACRO   ARGS
        IRP     ARGS
        TXI     *+1,1,1
        CLA     ARGS
        STO     STACK,1
        IRP
PUSH    END
*
POP     MACRO   ARGS
        IRP     ARGS
        CLA     STACK,1
        STO     ARGS
        TIX     **1,1,1
        IRP
POP     END
*
* LIST READING MACROS HERE
*
SEQRDR  MACRO   A,B
        CLA*    A               GET LIST HEADER
        STO     B               STORE IN READER CELL
SEQRDR  END
*
SEQLR   MACRO   A,B,C
        LAC     B,4             READER LINK
        CLA     1,4             GET DATUM FOR CELL
        STO     A               SAVE DATUM
        CLA     0,4             ADVANCE READER
        STO     B
        ANA     =0700000        SET FLAG
        ARS     15
        SUB     =1
        STO     C
SEQLR   END
```

```
*
*
       ENTRY   SEQDEC
SEQDEC SXA     RET,1
       SXA     RET+1,2
       SXA     RET+2,4
* INDEX REGISTER 1 IS THE POINTER TO THE TOP OF THE STACK
       LXA     ZERO,1
* INDEX REGISTER 2 IS THE LEVEL COUNTER FOR THE SEARCH
       LXA     ZERO,2
       CLA*    1,4
       STO     LIST
       CLA*    3,4
       STO     STATE
       CLA     DEPTH
       SUB     ONE
       ALS     18
       STO     LTEST
       TSX     $ITSVAL,4
       TXH     VALUEQ          VALUE LIST FOR TOP LEVEL
       TXH     LIST
       STO     VALUES
*
* THIS IS THE MAIN SEARCH LOOP.
*
* FIRST GET THE DECISION LOSS OF THE CURRENT PRIOR
LOOP   CLA     NODES           COUNT DECISION NODES
       ADD     =1
       STO     NODES
       TSX     $ITSVAL,4       GET DISTRIBUTION FOR THIS NODE
       TXH     PRIOR
       TXH     LIST
       STO     PLIST           SAVE NAME OF PRIOR LIST
       CLA     STATE
       STO     DECIDE
NOTERM TSX     $DLOSS,4        DECISION LOSS FOR DISTRIBUTION
       TXH     PLIST
       TXH     DECIDE          DECIDE NAME
       STO     LSAVE
       TXH     LTEST,2,0
       TSX     $MANY,4         SAVE DECISION VALUES IF AT LEVEL ZERO
       TXH     VALUES
       TXH     DECIDE
       TXH     LSAVE
LTEST  TXL     DOWN,2,**       CHECK LEVEL AGAINST DEPTH
       TXI     CONTIN,2,-1     RETURN
*
* HERE THE LEVEL IS LESS THAN THE REQUIRED DEPTH.
* THE TREE IS DEVELOPED TO THE NEXT LEVEL AND THE SEARCH
* CONTINUES.
*
DOWN   TSX     $RELTST,4       GET RELEVANT TESTS FOR THIS LEVEL
       TXH     LIST
       TXH     PLIST
*
* PROCESS THE BRANCHES AWAY FROM THE NODE DENOTED BY 'LIST'.
* EACH BRANCH CORRESPONDS TO A DIFFERNT TESTING ALTERNATIVE
* AT THE NODE DENOTED BY 'LIST'.
*
       SEQRDR  LIST,RDR        ESTABLISH READER FOR LIST
       SEQLR   TEST,RDR,I      GET NEXT TEST
```

```
READ    CLA     I
        CAS     ONE
        TRA     NOHEAD          NOT A HEADER
        TRA     *+2
        TRA     NOHEAD
        TXH     *+2,2,0
        TRA     RET             END
        TXI     CONTIN,2,-1     NOT THE END OF THE ANALYSIS
*
* PROCESS A SINGLE TEST BRANCH HERE
*
NOHEAD  CLA     ZERO
        STO     ELOSS           EXPECTED LOSS FOR THIS TEST
        TSX     $GROW1,4        'GROW' RESULT LIST FOR THIS TEST
        TXH     RDR
        TXH     PLIST
        STO     RESLST          NAME    OF RESULTS LIST
* SAVE VARIABLES HERE
        PUSH    (RDR,LSAVE,RESLST,PLIST)
        TSX     $NEWTOP,4       PUT THIS TEST ON TEST STACK
        TXH     TEST
        TXH     ISTRUN
*
* PROCESS ALL POSSIBLE RESULTS FOR THE TEST CURRENTLY BEING
* EVALUATED.
*
        SEQRDR  RESLST,RDR1     READER FOR RESULTS LIST
READ1   SEQLR   LIST,RDR1,I1    GET LIST FOR NEXT RESULT
        CLA     I1              CHECK FOR HEADER
        CAS     ONE
        TRA     GOON
        TRA     *+2             HEADER
        TRA     GOON
*
* ALL RESULTS FOR THIS TEST PROCESSED. RESTORE VARIABLES FOR TEST
* EVALUATION
*
        POP     (PLIST,RESLST,LSAVE,RDR)
        TSX     $POPTOP,4       GET THE TEST NAME
        TXH     ISTRUN
        STO     TEST
        TSX     $BOT,4          GET TEST COST
        TXH     TEST
        FAD     ELOSS           COMBINE WITH ELOSS
        STO     ELOSS
        TXH     CHECK,2,0
        TSX     $MANY,4         SAVE VALUES IF LEVEL IS ZERO
        TXH     VALUES
        TXH     TEST
        TXH     ELOSS
CHECK   CLA     ELOSS
        FSB     LSAVE           IS THIS THE BEST TO DATE
        TPL     DEL             NO
        CLA     ELOSS           BEST SO FAR
        STO     LSAVE
DEL     SEQLR   TEST,RDR,I      REMOVE THIS BRANCH
        LXD     RDR,4
        SXA     TEMP,4
        TSX     $REMOVE,4
        TXH     TEMP
        TRA     READ
```

```
*
* PROCESS A SINGLE TEST RESULT HERE
*
* SAVE VARIABLES
GOON    PUSH    (RDR1,ELOSS,LIST)
        TXI     LOOP,2,1        CYCLE
*
* FOLD THIS BRANCH BACK IN TERMS OF EXPECTED VALUE
*
CONTIN  POP     (LIST,ELOSS,RDR1) RESTORE VARIABLES
        TSX     $ITSVAL,4
        TXH     PROBQ           GET PROBABILITY OF THE BRANCH
        TXH     LIST
        STO     PROB
        LDQ     LSAVE           EXPECTED LOSS
        FMP     PROB
        FAD     ELOSS
        STO     ELOSS
        TRA     READ1
*
*
RET     AXT     **,1
        AXT     **,2
        AXT     **,4
        TRA     1,4
*
*
TESTQ   BCI     1,TEST
PLIST
PRIOR   BCI     1,PRIOR
TEMP
TEST
RDR
RDR1
I
I1
LIST
ELOSS
ONE     OCT     1
RESLST
LSAVE
ZERO    OCT     0
PROB
PROBQ   BCI     1,PROB
STATE
DECIDE
VALUEQ  BCI     1,VALUES
VALUES
        INSERT  COMMON          COMMON PACKAGE
        END
```

```
  UPD1   FAP
*
* THIS FUNCTION UPDATES THE PRIOR DISTRIBUTION IN
* 'LST1' BASED ON THE SIGN 'SYMP'. THE NEW DISTRIBUTION
* IS STORED IN 'LST2'.
*
        ENTRY   UPD1
```

```
        INSERT  MACROS
SUBST   MACRO   READER,DATUM
        LAC     READER,4
        CLA     0,4
        ARS     18
        PAC     0,4
        CLA     DATUM
        STO     1,4
SUBST   END
UPD1    SXA     RET,4
        CLA*    1,4
        STO     SYMP
        CLA*    2,4
        STO     LST1
        CLA*    3,4
        STO     LST2
        CAS     LST1
        TRA     DIF
        TRA     SAME
DIF     STZ     SWITCH
        TRA     *+2
SAME    STL     SWITCH
        CLA     FZERO
        STO     P
        TSX     $ITSVAL,4
        TXH     MEMQ
        TXH     SYMP
        STO     MEMLST
        SEQRDR  LST1,RDR
LOOP    SEQLR   STATE,RDR,I
CHECK   CLA     I
        CAS     ONE
        TRA     MORE
        TRA     *+2
        TRA     MORE
        CLA     P
        FSB     FZERO
        IPL     NOZERO
        CLA     P
RET     AXT     RET,4
        TRA     4,4
*
NOZERO  SEQRDR  LST2,RDR
AGAIN   SEQLR   STATE,RDR,I
        CLA     I
        CAS     ONE
        TRA     *+2
        TRA     RET-1
        SEQLR   PR,RDR,I
        CLA     PR
        FDP     P
        STO     PROB
        SUBST   RDR,PROB
        TRA     AGAIN
*
MORE    SEQLR   PROB,RDR,I
        TSX     $MEMBER,4
        TXH     STATE
        TXH     MEMLST
        TXH     ZERO
        TNZ     *+4
```

```
        CLA     FZERO
        STO     PR
        TRA     STEST
        PAC     0,4
        CLA     0,4
        PAC     0,4
        CLA     1,4
        STO     PR
        TSX     $PIJ,4
        TXH     SYMP
        TXH     PR
        STO     PR
STEST   CLA     SYMP
        TPL     MULT
        CLA     =1.E0
        FSB     PR
        STO     PR
MULT    LDQ     PROB
        FMP     PR
        STO     PROB
        FSB     FZERO
        TPL     *+2
        TRA     SCRAP
        CLA     P
        FAD     PROB
        STO     P
        ZET     SWITCH
        TRA     *+2
        TRA     DIFPRO
        SUBST   RDR,PROB
        TRA     LOOP
*
DIFPRO  TSX     $MANY,4
        TXH     LST2
        TXH     STATE
        TXH     PROB
        TRA     LOOP
*
*
SCRAP   NZT     SWITCH
        TRA     LOOP
        LAC     RDR,4
        CLA     0,4
        ARS     18
        STA     ADD
        CLA     RDR
        ARS     18
        STA     ADD1
        SEQLR   STATE,RDR,I
        TSX     $REMOVE,4
        TXH     ADD
        TSX     $REMOVE,4
        TXH     ADD1
        TRA     CHECK
*
*
FZERO   OCT     233000000000
ONE     OCT     1
ZERO    OCT     0
SWITCH
RDR
```

```
STATE
I
LST1
LST2
SYMP
MEMQ    BCI     1,MEMBER
ADD
ADD1
P
PR
PROB
MEMLST
        END


MEMBER  FAP
        ENTRY   MEMBER
        INSERT  MACROS
MEMBER  SXA     RET,4
        SXA     RET+1,1
        CLA*    1,4
        STO     GOAL
        CLA*    2,4
        STO     LIST
        CLA*    3,4
        TNZ     LEVEL1
        CLA     LIST
        STO     NEXT
        TSX     ONCE,1
RET     AXT     **,4
        AXT     **,1
        TRA     4,4
*
*
LEVEL1  SEQRDR  LIST,RDR
LOOP    SEQLR   NEXT,RDR,I
        CLA     I
        CAS     ONE
        TRA     GOON
        TRA     *+2
        TRA     GOON
        ZAC
        TRA     RET
GOON    TSX     ONCE,1
        TRA     LOOP
*
*
ONCE    SEQRDR  NEXT,R
OLOOP   SEQLR   CAND,R,F
        CLA     F
        CAS     ONE
        TRA     MORE
        TRA     *+2
        TRA     MORE
        ZAC
        TRA     1,1
MORE    CLA     CAND
        CAS     GOAL
        TRA     OLOOP
```

```
        TRA     *+2
        TRA     BLOOP
        LAC     R,4
        CLA     0,4
        ARS     18
        ANA     =077777
        TRA     RET
*
CAND
NEXT
RDR
I
R
F
GOAL
LIST
ONE     OCT     1
        END
```

```
    UN   FAP
        ENTRY   UPD1
        ENTRY   NSCOMP
        INSERT  MACROS
SUBST   MACRO   READER,DATUM
        LAC     READER,4
        CLA     0,4
        ARS     18
        PAC     0,4
        CLA     DATUM
        STO     1,4
SUBST   END
*
HCHECK  MACRO   LAB1,LAB2,FLAG
        CLA     FLAG
        CAS     ONE
        TRA     LAB1
        TRA     LAB2
        TRA     LAB1
HCHECK  END
*
* 'UPD1' DOES THE STANDARD UPDATE OF LST1 INTO LST2
* WHEN A SYMPTOM IS THE 'AGENT'.
*
UPD1    STI     INDIC
        RIR     17              SAVE AND SET INDICATORS
        TRA     START
*
* 'NSCOMP' DOES THE NORMAL UPDATE WITH A TEST AS THE AGENT.
*
NSCOMP  STI     INDIC
        RIR     17
        SIR     1
START   SXA     RET,4
        SXA     RET+1,1
        CLA*    1,4             'AGENT'
        STO     AGENT
        CLA*    2,4             FIRST LIST
        STO     LST1
```

```
        CLA*    3,4             SECOND LIST
        STO     LST2
        CAS     LST1            SAME LIST.Q.
        TRA     *+2             NO
        SIR     2               YES
        CLA     ZERO
        STO     P
        TSX     $ITSVAL,4
        TXH     MEMQ
        TXH     AGENT           GET MEMBER LIST OF AGENT
        STO     MEMLST
*
* PROCESS EACH STATE ON LST1.
*
        SEQRDR  LST1,RDR
LOOP    SEQLR   STATE,RDR,I
CHECK   HCHECK  MORE,NORM,I
*
        CLA     P
RET     AXT     **,4
        AXT     **,1
        LDI     INDIC
        TRA     4,4
*
*
NORM    SEQRDR  LST2,RDR        NORMALIZE LST2
AGAIN   SEQLR   STATE,RDR,I
        HCHECK  DIV,RET-1,I
DIV     SEQLR   PR,RDR,I
        CLA     PR
        FDP     P
        STO     PROB
        SUBST   RDR,PROB
        TRA     AGAIN
*
MORE    SEQLR   PROB,RDR,I
        RFT     1               TEST PROCESS SWITCH
        TRA     NC              'NSCOMP'
        TSX     GETP,1          GET P(AGENT/STATE)
        TXH     MEMLST
        TXH     PR
        CLA     AGENT           CHECK FOR NEGATIVE RESULT
        TPL     MULT
        CLA     =1.E0
        FSB     PR
        STO     PR
MULT    CLA     =1.E-6          CHECK FOR 'ZERO' PROB
        FSB     PR
        TMI     OK
RTEST   RFT     2
        TRA     SCRAP
        TRA     LOOP
OK      LDQ     PROB
        FMP     PR
        STO     PROB
        CLA     P
        FAD     PROB            ACCUMULATE PROBABILITY
        STO     P
        RFT     2               AGAIN TEST LISTS
        TRA     SAME
        TSX     $MANY,4
```

```
        TXH     LST2
        TXH     STATE
        TXH     PROB
        TRA     LOOP
SAME    SUBST   RDR,PROB
        TRA     LOOP
*
NC      CLA     =1.E0           INITIALIZE PR
        STO     PR
        SEQRDR  MEMLST,R        READ MEMBER LIST OF TEST
NLOOP   SEQLR   AGENT,R,F       NEXT SIGN
        MCHECK  GOON,MULT,F
GOON    TSX     $ITSVAL,4       MEMBER LIST OF SYMP
        TXH     MEMQ
        TXH     AGENT
        STO     SYMEM
        TSX     GETP,1
        TXH     SYMEM
        TXH     TEMP
        CLA     PR
        FSB     TEMP
        STO     PR
        FSB     =1.E-6          TEST FOR ZERO
        TPL     NLOOP
        TRA     RTEST
*
* GET THE PROBABILITY OF A SIGN GIVEN A SYMP
*
GETP    CLA*    1,1
        STO     HOLD
        TSX     $MEMBER,4
        TXH     STATE
        TXH     HOLD
        TXH     ZERO
        TNZ     *+4             FOUND
        CLA     ZERO
BACK    STO*    2,1             STORE RESULT
        TRA     3,1
        PAC     0,4             GET PROB CELL
        CLA     0,4
        PAC     0,4
        CLA     1,4
        STO     HOLD
        STA     RIGHT
        ARS     18              FAST CHECK FOR NAME
        ANA     =077777
        CAS     RIGHT
        TRA     NONAM           NOT A NAME
        TRA     *+2
        TRA     NONAM
        TSX     $PIJ,4          POSSIBLY A NAME
        TXH     AGENT
        TXH     HOLD
        TRA     BACK
NONAM   CLA     HOLD
        TRA     BACK
*
SCRAP   LXD     RDR,4
        SXA     ADD,4
        SEQLR   STATE,RDR,I
        LXD     RDR,4
```

```
        SXA     ADD1,4
        TSX     $REMOVE,4
        TXH     ADD
        TSX     $REMOVE,4
        TXH     ADD1
        TRA     CHECK
*
SYMEM
ADD
ADD1
INDIC
RDR
R
I
F
STATE
LST1
LST2
RIGHT
HOLD
TEMP
MEMLST
MEMQ    BCI     1,MEMBER
ZERO    OCT     0
AGENT
P
PR
PROB
ONE     OCT     1
        END
```

Biographical Note

George Anthony Gorry was born in Glens Falls, New York on November 16, 1940. He attended public schools there, graduating from Glens Falls High School in June, 1958. He entered Yale University in September, 1958, where he studied chemical engineering. He received a Bachelor of Engineering degree with high honors in June, 1962. He entered the University of California at Berkeley in September, 1963, and received a Master of Science degree in chemical engineering in September, 1963. In September, 1963, he entered the Sloan School of Management at M.I.T. In September, 1965, he was married to the former Lucinda Jean Paulsen of Belmont, Massachusetts.

Mr. Gorry joined the staff at M.I.T. as a teaching assistant in the Sloan School in September, 1964, and was appointed as Instructor in Management in July, 1965. He has taught courses in operations research and heuristic programming. During the summer of 1964, Mr. Gorry worked at Project MAC, and in the summer of 1965, he became associated with the Boston Programming Center of the IBM Corporation. Since 1966, he has been a consultant to a number of organizations concerned with computer technology.

CS-TR Scanning Project
Document Control Form

Date : 12/14/95

Report # LCS-TR-44

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 254 (260-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☒ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:
IMAGE MAP! (1-254) UNNUMBERED TITLE PAGE, ii-ix, UNNUMBERED BLANK, 1-244
(255-260) SCANCONTROL, COVER, DOD, TRGTS(3)

Scanning Agent Signoff:
Date Received: 12/14/95 Date Scanned: 12/19/95 Date Returned: 12/28/95

Scanning Agent Signature: Michael W. Cook

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Massachusetts Institute of Technology Project MAC	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP None

3. REPORT TITLE

A System for Computer-Aided Diagnosis

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Ph.D Thesis, Sloan School of Management, June 1967

5. AUTHOR(S) *(Last name, first name, initial)*

Gorry, George A.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1967	253	22

8a. CONTRACT OR GRANT NO. Office of Naval Research, Nonr-4102(01) b. PROJECT NO. NR 048-189 c. RR 003-09-01 d.	9a. ORIGINATOR'S REPORT NUMBER(S) MAC-TR-44 (THESIS) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D. C. 20301

13. ABSTRACT This thesis describes a model diagnostic problem and a computer program designed to deal with this problem. The model diagnostic problem is an abstract problem. A major contention of this thesis, however, is that this problem subsumes the principal features of a number of ostensibly different real diagnostic problems including certain problems of medical diagnosis and the diagnosis of machine failures. A second major contention of this thesis is that strategies for the solution of the model diagnostic problem can be formulated in terms sufficiently explicit to permit their incorporation in a computer program.

The diagnostic program was implemented on the time-sharing system at Project MAC. It was applied to two medical problems, the diagnosis of congenital heart disease, and the diagnosis of primary bone tumors. The results obtained here suggest 1) that a computer program can be of considerable value as a diagnostic tool, and 2) that it is quite advantageous for such a program to perform sequential diagnosis as it interacts with the user.

14. KEY WORDS

Computers	Multiple-access computers	Real-time computers
Computer-aided diagnosis	On-line computers	Time-sharing
Machine-aided cognition	On-line diagnosis	Time-shared computers

DD FORM NOV 44 1473 (M.I.T.)

Unclassified

Security Classification

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**