APPLICATIONS OF FUZZY LOGIC TO MEDICAL DIAGNOSIS

Harry Wechsler

Technical Report #62

Information and Computer Science

APPLICATIONS OF FUZZY LOGIC TO

MEDICAL DIAGNOSIS

Harry Wechsler
Information and Computer Science Department
University of California, Irvine

1. INTRODUCTION

Automated medical diagnosis has been of continual interest to a variety of people. A lot of work has been done in the field, but except for the work of Shortliffe et al [8], much of it is technically based on statistical decisions models, and the results were not too encouraging for further investigation.

A summary of diagnostic models is presented by Craft [14], and by far the Bayes Criterion is the most common. It is based on the notion that the best set of pattern assignments is the one resulting in the least expected probability of misclassification. This criterion has been used by Garry and Barrett [16]. Craft concludes his paper showing that the accuracy obtained by using several different statistics was little better than 50-60%. He concludes that further refinements of the statistical approach were unlikely to provide a major improvement and that the diagnosis models should point a new direction.

The design of all the systems presented by Craft share the following weaknesses:

a) the statistical dependencies require acquisition of high order conditional probabilities;

b) it is hard to get a good data base, i.e., to get the probability of any disease with respect to any subset of symptoms;

c) there is no way of adding procedural knowledge.

The knowledge such a system can have can be expressed either in a static form, a descriptive one or in a constructive form, a procedural one. A descriptive knowledge base system will merely look for the probability of having a specific disease given a set of symptoms and will choose the diagnosis based on a likelihood approach.

In contrast, a procedural knowledge base is composed of a set of programs which are organized to test for a cluster of symptoms, a syndrome or a disease. Medical knowledge of what symptoms should be considered after a particular set are found is organized as a body of executable code -- a procedure for systematically reasoning about the sets of symptoms attributed to the patient being diagnosed. In this paper we advocate a variety of this approach.

The proposed approach is going to have the following advantages:

a) diagnostic information is represented procedurally;

b) using procedures to explicitly deal with statistically dependent symptoms through use of boolean combinations, as

(AND (S_1 (OR S_2 S_3))) where $S_{i = 1, 2, 3}$ stands for symptom i;

c) adding new information via change (or extension) of procedures rather than through building a large data base to improve the statistical decision rules;

d) allowing Inexact Concepts (Multi-Valued) to deal with degrees of a symptom;

e) allowing the interpretation of inexactness to vary with context.

These last two advantages are the unique features of the diagnostic system proposed here.

At this point it is appropriate to state how this approach is related to the one developed by Shortliffe et al, given that his system, the MYCIN system, is the closest one to our proposed system. In MYCIN, the procedural knowledge is represented as production rules (P,A) where the action A is taken if the premises P are fulfilled. No provisions are made for a full goal-oriented approach (this is a variety of heuristic search) [3], where one decision-program for a specific disease can contain as one of its steps (statements) a call for another decision-program. Shortliffe's paper states the following:

> "Just as Bayesians who use their theory wisely must insist that events be chosen so that they are independent (unless the requisite conditional probabilities are known), we must insist that dependent pieces of evidence be grouped into single rather than multiple rules. The system therefore becomes unworkable for applications in which large numbers of observations must be grouped in the premise of a single rule in order to insure independence of the decision criterion. In addition we must recognize logical subsumptions, when examining or acquiring rules and thus avoid counting evidence more than once (For ex: if $S_1 \rightarrow S_2$, then the confirmation of the hypothesis H_1 with respect to the symptoms S_1 & S_2 is equal to the confirmation of H_1 with respect to S_1, regardless of the value/confirmation of H_1 with respect to S_2). MYCIN does not know this."

Therefore, MYCIN system represents only a first step toward a general goal-oriented approach. The way new information is added in MYCIN is the one to be also used in our approach. The inexact reasoning used by MYCIN is a measure of belief/disbelief rather than the use of fuzzy concepts. Each production rule (P,A) has an associated certainty factor that reflects the measure of belief/disbelief of the expert who suggested the rule. In our approach we allow various predicates e.g. "blood pressure" to take on a range of values. Thus, symptoms sets can be expressed in a fuzzy logic. In the following paragraphs we are going to present the use of fuzzy concepts, first informally, and eventually to axiomatize the mechanics of fuzzy computation. Then we show how a fuzzy diagnostic system should look like.

2. MECHANICS OF FUZZY COMPUTATION

Computers perform their tasks following the way they are instructed and they will continue to do so. The purpose of this paragraph is to show how we expect to get better performance from computers and a behavior more similar to that exhibited by people. This behavior is going to be a fuzzy behavior and by it we mean that the instructions to be executed deal with 'fuzzy concepts.' This approach, hopefully, will diminish the gap between computers and people and what is more important it will improve their performance, leading eventually to build systems that can reason effectively with 'inexact' concepts. We hope that the way we can get all the above at the present time is by incorporating in one of the new programming languages, e.g., QA4[3], fuzzy concepts of behavior. The first step in this direction has been taken by Kling [7].

The prevailing attitude in contemporary scientific research is to get exactness as much as possible. But, as Zadeh writes in his report [6],"a wise step can be today, less preoccupation with exact quantitative analyses and more acceptance of the pervasiveness of imprecision in much of human thinking and perception." He believes that by accepting this reality rather than assuming that the opposite is the case, we are likely to make more real progress in the understanding of the behavior of humanistic systems than it is possible within the confines of traditional methods. The effectiveness of a problem-solving system depends on the ability to provide it with a symbolic representation of the environment in which it is called to operate, and with heuristics which control the action of the problem solving system depending on such a knowledge representation [2].

One of the most exciting problem solving system is the Question Answering System (QA) [2]. Such a system must store a lot of data which can be relevant in different ways to different queries. Keeping in mind the finiteness of our resources and being willing to get efficiency as much as possible, we would like to express the way the data is relevant with respect to different criteria, as concisely as possible. Before we present our ideas about designing good QA systems according to these concepts, we think it worthwhile to elaborate our description of imprecise concepts.

The concept of information is based on the concept of probability. The concept of probability is defined on a σ-algebra or a distributive lattice. All the laws of Boolean logic can be derived from the characteristics function f(A/a) which is 1 if object 'a' belongs to class A and it is 0 if 'a' does not belong to class A. A class is understood as the extension of a predicate. The basic postulate that 'a' and A determine the value of f(A/a) which is either 0 or 1 is called "the postulate of fixed truth set." The breakdown of this postulate was already noticed when philosophers made distinction between the primary quality and secondary quality. The characteristic function of a secondary quality depends not only on 'a' and A but also on a third argument x, which is the observer [4]. At the same time it must be clear to the reader, that in the majority of cases, we cannot reduce the continuous value of f(A/a) to a binary case.

A fuzzy subset A of a universe of discourse U is characterized by a membership function $m_A : U \rightarrow [0, 1]$ which associates with each element 'u' of U a number $m_A(u)$ in the interval [0,1], with $m_A(u)$ representing the grade of membership of u in A [1]. Kling in his report [7], uses these concepts relating them to PLANNER, a procedural language developed at MIT [20].

Now, let us ask some very simple questions. Are we really interested in the grade of membership of each of A's members? What will happen if some day the definition of f(A/a) is going to be changed? Are we going to check in our storage all previous members of A, change their membership grade, and maybe exclude some of them? And this is not all that must be done. What about possible new members of A,(i.e., data for which f(A/a) > 0) and what about different membership grade with respect to different observers x? Are we going to search all the data? From all these difficulties we believe that using the membership grade is impractical for an efficient QA system.

In the following we will try to take advantage of the facilities provided by QA4 [3][1] and to show how it is possible to build a QA system which will incorporate in itself the imprecision-fuzziness concepts discussed above. Rather than using membership grade we are going to define classes of generative functions, whose task will be to define implicitly fuzzy sets.

A class is related to a concept and it contains one or more functions f_i^P (P stands for the property to be checked), each of them related to some context (i.e., to the observer x_i). The context facility is provided by QA4, so no special problems will arise from the above definitions. Now, if we change our mind about one concept, all that must be done is to change the definition for some function f_i^P. If we think about properties we can distinguish two categories:

a) The first category contains those properties which can be measured (example: height, weight, age). These will be called primitive properties.

b) The second category contains those properties which can not be measured and can be usually expressed only as functions of primitive properties (example: ripeness, freshness)[2]. The extension of this definition to a recursive one is obvious, i.e., if

Property b_1 = f (properties $\vec{a}$)

Property b_2 = $f(\vec{a}, b_1)$ is well defined.

---

[1] The QA4 language has features that are recognized as useful for problem-solving programs; these features include built-in backtracking, parallel processing, pattern matching, and set manipulation. Expressions are put into a canonical form and stored uniquely, so that they can have property lists. A context mechanism is provided, so that the same expression can be given different properties in different contexts. The QA4 interpreter is implemented in LISP and can interface with LISP programs. QA4 can store information in an imperative form, as a program. This makes it possible to store advice locally rather than globally; in giving information to the system we can tell it how that information is to be used [3].

[2] This classification is not at all arbitrary. As Boole wrote: "All logical propositions may be considered as belonging to one or the other of two great classes, to which the respective names of 'PRIMARY' or 'CONCRETE PROPOSITIONS' and 'SECONDARY' or 'ABSTRACT PROPOSITIONS,' may be given. In other words, every assertion either expresses a relation among things and therefore it is CATEGORICAL or it expresses a relation among propositions and therefore it is HYPOTHETICAL in its nature.

How are we going to deal with properties P which belong to the category (a)? In other words, how $F^P = \{f_i^P\}$ are defined?

$$f_i \triangleq f_i : H^P \times A^P \rightarrow \tilde{u} \in U$$

where H^P and A^P stand for the hedge[3] and adjective domain respectively and $\tilde{u}$ is some subset of the universe of discourse. The result of applying f_i^P is to get in fact $f_i^P(\tilde{u}/a)$, i.e., the predicate $\tilde{u}$ associated with class u. By using this kind of definition we allow, in fact, that the classes defined by $\{f(\tilde{u}/a)\}$ be overlapping, and this is consistent with the imprecision concept.

Example: P = age

H = {null[4], very, little, ...}

A = {old, young}

$F^P = \{f_i^P\}$

For some i : f_i^P (null young) = f_i^P (young) = (age 20, age 30).

If A = {baby, child, teenage, young, mature, old} for some other i:

f_i^P (baby) = (age 0, age 3)

f_i^P (child) = (age 2, age 14)

f_i^P (teenage) = (age 12, age 18)

f_i^P (young) = (age 17, age 30)

f_i^P (mature) = (age 26, age 60)

f_i^P (old) = (age 55, $\longrightarrow$ [5])

The overlapping of the above sets express better the concept of fuzziness than the membership function. We delay to the sequel the way we can cope with the properties belonging to the category (b).

Another important question is: being given with two items, which of them fits better the predicate $\{f(\tilde{u}/a)\}$ assuming that both of them belong to the class.

---

[3]By HEDGES we mean words of the following type: very, more or less, quite, fairly and so on.

[4]The hedge "null" means that no hedge at all is going to be used!

[5]Any positive integer greater than 55.

(In Zadeh's terminology, given a set A and a_1, $a_2 \in A$ decide either $m_A(a_1) \geq m_A(a_2)$ or $m_A(a_2) \geq m_A(a_1)$). We think that for the above question, the following two kinds of order-decision are suitable and sufficient. The first one tells us the closer you are to the middle of the interval defined by $f(\tilde{u}/a)$ the better you fit the property.

Example: f_i^{AGE} (young) = (age 17, age 30)

Now, if we look for someone young, we will prefer someone whose age is 25 to someone whose age is 20. Let us call this order the middle order and associate with f, M such that the definition of the class and its order are represented together by $< f(\tilde{u}/a); M >$.

The second criterion tells us simply that the first member of the interval is the best, i.e., we will get the interval ordered from left to right.

Example F_i^{GRADE} (good) = (A B). Let us call this order the LB order and then, as above, we will get $< f(\tilde{u}/a); LB >$, where LB has the property that the more left you are in the interval, the better off you are.

How can we conclude the discussion of the properties belonging to the category (a)?

First of all, let us remark that the definition is built based only upon the tasks we can be posed with. There is no real meaning to the membership grade. We can be asked (related to a QA system) only two kinds of questions related to some primitive property:

(i) Does someone have the property (i.e., is someone young)?

(ii) If both a_1 and a_2 have the property, which of them fits it better?

As we have seen before, by using our definition, we are able to answer both questions. This is not the only advantage of using our definitions. By using this approach all the operations involving predicates can still be done using the set-theoretical operations, which is of course, much more naturally than deriving new definitions, as it is done in [1].

What about implementation? As in QA4 the ASSERT and DENY statements take care about the existence of an item, these constructs are the ones to be used for the existence of a particular item at a given time. (For the existence, we assume that the postulate of fixed truth set is valid.) What about properties? This is done in QA4 by using PUT statements of the following format:

(PUT syntactic-form indicator property)

Now, the PUT statement will look like the previous or the following:

(PUT syntactic-form indicator value-field function)

where we understand the value field as being non-empty, i.e. containing either a real value or an adjective preceded perhaps by a hedge; and by function we mean the function stored in the system which takes care of the mapping (and also implicity

takes care of the context). The EXISTS statement is going to be changed in the same way. The PATTERN MATCHER must be built following the same concepts. Let us develop the idea and try to match the value field set up by the PUT statement with the value field the EXISTS statement is looking for.

(i) Both value fields are real-valued. Therefore, we can use the previous mechanism of the PATTERN MATCHER for comparing them.

(ii) Both value fields are expressed by [hedge] adjective. Use the set inclusion concept. Example:

(PUT Dick age very old f_1) (EXISTS (←X man) age old)

where f_1 is some context used for the age-property and ← means the variable X is to receive a value. If we assume that before the PUT statement we have executed

(ASSERT (Dick man))

we will succeed in executing the EXISTS statement because hedge (a) $\subseteq$ a, i.e., very old $\subseteq$ old. At the same time we should report a failure for the attempt to fit old $\subset$ very old.

(iii) Suppose now, that the value field of PUT is real-valued and that of EXISTS is [hedge] adjective. Use f ([hedge] adjective) for generating the class and check if the real-value belongs to it or not.

(iv) If the value field of PUT is [hedge] adjective and that of EXISTS real-valued, we will report a failure.

Now that we have described the category (a) of properties and the way we deal with them, we will exhibit the way we can cope with the category (b) of properties, i.e., non-primitive properties. For the sake of simplicity, we are going to employ a simple example not drawn from the field of medical diagnosis. Suppose we want to find in our refrigerator a ripe fruit. We need a goal program of type CHOOSE and we know that ripeness is a function of freshness, color, and size.

Size and color are properties of category (a) type. Freshness is a property of category (b) type and it is a function of the numbers of days the fruit was kept in the refrigerator (this is a property of category (a) type).

So, f (very fresh) → (0 days, 2 days)
f (fresh) → (0 days, 4 days)
f (unlikely fresh) → (3 days, 7 days)
f (not fresh) → (6 days, → [6])

Also, as we said previously

RIPENESS = g (FRESHNESS ∩ COLOR ∩ SIZE)

What will the program look like? First of all, at some stage we have the statements:

(ASSERT ((SET FRUIT1 FRUIT2. . .FRUIT1ØØ) FRUIT))

and for each fruit j

[6] Any positive integer greater than 6.

```
(ASSERT (FRUITJ FRUIT))
(PUT (FRUITJ FRUIT) KIND APPLE)
```

The kind can be apple as above or anything else. Later during the execution:

```
(GOAL $CHOOSE (←M RIPE))
```

For performing this goal of class CHOOSE the Pattern Matcher will evluate the following λ-expression:

```
(LAMBDA (←N RIPE)
    (PROGRAM (DECLARE L1 L2)
    (EXISTS (SET ←L1 ←←L2) FRUITS)
    (EXISTS ($L1  FRUIT) SIZE NOT TOO BIG f1size)
    (EXISTS ($L1  FRUIT) FRESHNESS FRESH f1fresh)
    (EXISTS ($L1  FRUIT) COLOR RED f1color)
    (DENY ((SET FRUIT1. . .FRUIT1ØØ) FRUITS))
    (ASSERT (SET $$L2 FRUITS))
    (RETURN (TUPLE ($L1 = (GET ($L1 KIND)))))
    (DENY (($L1 FRUIT)))))
```

In the above example we used the DENY and ASSERT statements in order to modify the state of the world after we choose a ripe fruit because we look for a ripe fruit in order to eat it, and therefore it doesn't exist any more after it is found. Usually λ-expressions have variables with prefix ← in the bound variable part; if we want to refer to the bindings of those variables in the body of the lambda expressions, we use the same variables with the prefix $. The rules for matching a variable prefixed by ←← or $$ are analogous to those for ← and $ prefixes. However, variables with these prefixes will be bound to a fragment of a set [22].

3. MEDICAL DIAGNOSIS USING PROCEDURAL AND FUZZY CONCEPTS

In many ways the consulting part of a medical-diagnosis system is like making available a computer-stored textbook of medicine [13]. Let us reproduce from [12] the symptoms of ulcers:

> "In the classic syndrome the pain is felt at or slightly below the tip of the xiphoid process, spreading more to the right of the midline than to the left, and rarely rising above the xiphisternal junction. It occurs 1 to 3 hours after eating, sometimes awakens the patient at night, and is least commonly present in the morning before breakfast. It is relieved by milk and other protein foods, even though cold, by antacids,... . In gastric ulcer, pain often occurs less than an hour after eating, is less reliably relieved by food, is accompanied by nausea, and only very rarely occurs at night."

As it is easy to see from the above description, rather than dealing with probabilities of different kinds of ulcers with respect to the syndromes, we look for the presence or the absence of syndromes, which are almost all expressed in fuzzy concepts.

All of the following examples are expressed in the terminology provided by QA4 [3] and the meaning of the symbols used was described in the previous paragraph.

Looking for a possible ulcer is equivalent to the following statement in a goal-oriented environment

(GOAL $FIND (←X ULCERS))

A program of type FIND is going to search for possible candidates belonging to the ulcers set and check if the syndromes are either present or not. The ulcers' decision-making knowledge is represented by context-dependent functions as $f^{property}_{ulcers}$, whereas the gastric ulcer which is a subset of ulcers can be referred by either f^{P}_{u} or $f^{property}_{gastric\ ulcer} = f^{P}_{gu}$. A typical program is a sequence of statements which includes, among the others, jump, conditional statements and call to any other program. Based upon the description provided by Mellinkoff, we know that the location of the pain should be almost the same for different kinds of ulcers, but if a gastric ulcer is present, it should be accompanied by nausea and only very rarely occurs at night.

Therefore, the procedural-knowledge should first of all check for common characteristics belonging to all kinds of ulcers and only finally to check for a specific kind of ulcer by using a multiple-conditional statement. Under the general context of the patient being checked for ulcers, a possible program should look like:

(LAMBDA (←D ULCERS)
(PROGRAM ( )
(EXISTS Patient pain-location slightly below the top of the xiphoid process f^{P1}_{u})
(EXISTS Patient time of pain breakfast f^{P2}_{u})))

If all the above tests are positive, we know that an ulcer has been detected and we shall want to find out which one.

(EXISTS (SET ←L1 ←←L2) ULCERS)

The first choice may be gastric ulcers and if the answer to the following tests are true, we will conclude that a gastric ulcer is present.

(COND((AND (EXISTS Patient nausea $f^{nausea}_{L_1}$)
(EXISTS Patient time of pain $f^{time}_{L_1}$))
(RETURN (TUPLE (=Patient = (GET ($L1 KIND)))))))

If one of the precedent EXISTS statement fails, i.e., it returns NIL, the AND is false and a next set of conditions is checked for finding what kind of ulcer we did detect.

From this brief example it is clear that there is no way to avoid the use of fuzzy concepts. In fact, the description reproduced contains almost only fuzzy concepts!

4. A PROPOSED MEDICAL DIAGNOSIS SYSTEM

Medical diagnosis is equivalent to pattern recognition and an excellent review of the field is provided by Patrick's paper [13].

Our intention is to present the difficulties encountered by previous research and to show how they can be solved using the fuzzy concepts discussed elsewhere in this paper. Patrick reviews some of the problems such systems failed to cope with. These include:

a) the measurements $x_1, x_2 \ldots x_L$ are considered to be statistically independent features when in fact they are measurements which are statistically dependent;

b) there are failures to recognize significant measurement (features) for a particular class (failure to incorporate a priori dimensionality reduction);

c) inability to introduce a priori knowledge about correlation among measurements or features.

Lusted in [15] had noted that even those researchers using Bayesian statistical inference models as an approach to medical decision-making have begun to group or cluster symptoms as a way to improve the efficiency of their mathematical search strategy.

Wortman in [17] feels that "hierarchical structures" are needed, because they will allow exponential increases in alternatives with only linear increases in time and that this will fit with Mandler's theory of memory organization [19].

Ledley [18] suggests introducing a priori medical knowledge as a class-feature relationship defined in terms of decision tables. He realizes that the decision table approach may not be appropriate when class conditional probability densities "overlap" and suggests using the Bayes theorem with multivariate class conditional probability densities, not specifying how to estimate them. As a feasible alternative he suggests storing in memory many sets of measurement values and recognizing which ones are closest to a patient's measurement vector (nearest neighbor rule). The system we are going to propose is aware of the way knowledge can be stored, retrieved and inferred, and the implications with regard to efficient use of the memory and fast-decision making. Our model is supposed to be more reliable in that it is closer to the way in which real-world medical diagnosis is performed and therefore it is more likely that the physicians will accept it, by cooperating with and being willing to improve it.

Taking in account the deficiencies remarked earlier we decided to discard the Bayes criterion as our tool. We do not choose anymore a possible disease because it is the most likely to be with respect to some cost function, but rather we are looking to see if a given disease is either present or not.

The knowledge of our system should be built around two different memory-structures corresponding to the first and second category attributes, respectively. The first storage unit is intended to contain data relevant to first category attributes and their valued-intervals within different contexts. Example: blood pressure and

fatness are such attributes. Within some defined context as heart diseases, we can define high blood pressure as being the valued-interval (17, → [7]) while for an intern physician high blood-pressure is (14, → [8]).

The decision-making knowledge is going to be stored in the second storage unit in a procedural-form (goal-oriented programs, i.e., disease-oriented programs).

For each patient we are going to keep a data-base unique to him (the ongoing record) which is used by the decision-making unit.

A top-level diagram of the proposed system should look like the one shown in Figure 1.

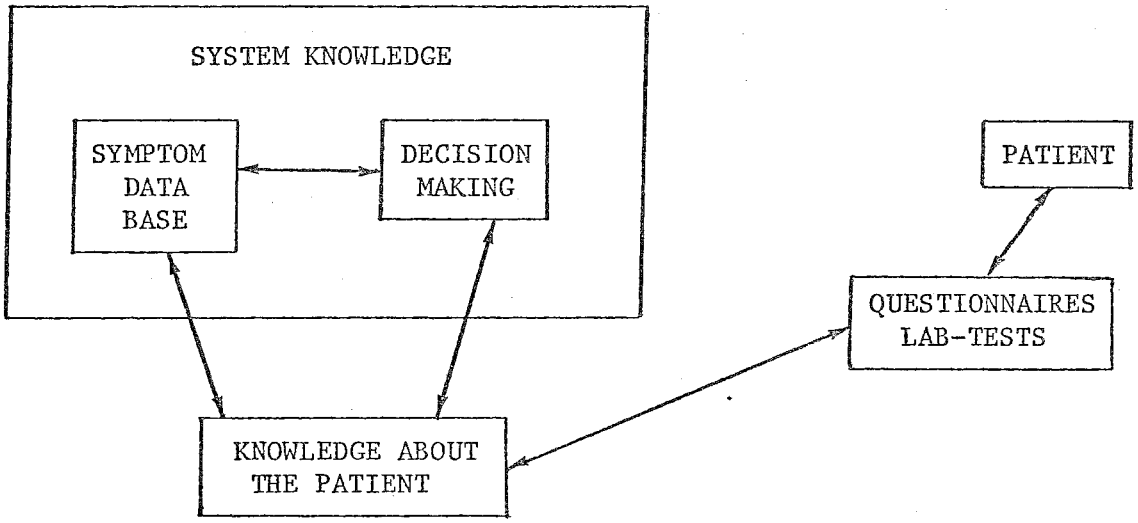

Figure 1.

The Symptom Data Base is built around the fuzzy concepts and all pertinent data of use for the decision-making is present here. More than one context is allowed to exist at a given time and the medical-diagnosis is done with respect to a given context. Contexts are created by the "decision-making" programs during the search for a solution.

The decision-making unit is organized around clusters of programs (i.e., specialists) which should be activated by demons[9]. Backtracking, a facility existing in QA4, is also used because the demons can point to different clusters of programs according to how their preconditions are fulfilled. Based upon the

---

[7]Any positive integer higher than 17.

[8]Any positive integer higher than 14.

[9]A demon is the software equivalent for hardware triggers that are fired out when some preconditions are fulfilled and force specific paths through the decision-making tree to be taken.

procedural form the knowledge is stored, sharing of procedures is made available and therefore the size of this memory unit can be kept reduced. Two of Patrick's problems were the inability to recognize significant features for a particular class (cluster) and the inability to introduce a priori knowledge about correlation among features. The first of the above problems can be solved by using a natural sequence-order of statements within the programs which are supposed to recognize some diseases. The ordering should force us to fail as soon as possible if we follow a misleading path and to backtrack to the previous point of choice.

The correlation among features can be introduced by using the IF-THAN-ELSE conditional construct. The data-base containing knowledge about the patient is a dynamic unit rather than a static one. It should contain positive evidence as well as negative with respect to the presence or the absence of symptoms. This unit should simulate a short-term memory and be used previously to any attempt to retrieve or infer something from the system knowledge.

5. CONCLUDING REMARKS

A proposed medical diagnostic system has been proposed in which the fuzzy concepts play a major role. It improves upon some of the most obvious weaknesses of the statistical decision-theoretic approaches most commonly used in automated diagnostic systems. Some of these defects can be alleviated by the use of procedural representations of the organization of symptoms into disease related syndromes. Furthermore, the use of fuzzy concepts allows additional and substantive flexibility and expressive power.

6. ACKNOWLEDGEMENT

Special thanks are given to Rob Kling for his suggestions and encouragement in writing this paper.

REFERENCES

[1] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," Memorandum No. ERL-M411, October 1973, Berkeley.

[2] F. Sirovitch, "Some ideas on semantic memory in automatic learning of heuristics," Carnegie Mellon University, Technical Report, 1972.

[3] J. Derkensen, J. F. Rulifson and R. A. Waldinger, "The QA4 language applied to robot planning," Proc. Fall Joint Computer Conference 1972, New York. Spartan Books, 1972.

[4] S. Watanabe, "Modified Concepts of logic, probability and information based on generalized continuous characteristic function," Information and Control 15, 1-21 (1969).

[5] A. Turing, "Computing machinery and intelligence," from "Computer and Thought," McGraw Hill, 1963.

[6] L. Zadeh, "Outline of a new approach to the analysis of complex systems and decision procsses," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-3, No. 1, January 1973, pp. 28-44.

[7] R. Kling, "Fuzzy Planner: Reasoning with inexact concepts in a procedural problem solving language," Journal of Cybernetics (in press).

[8] E. Shortfiffe et al, "An artificial intelligence program to advise physicians regarding antimicrobial therapy," Computer and Biomedical Reserach 6, 544-560 (1973).

[9] N. J. Nilsson, "Learning machines," McGraw Hill 1965.

[10] D. Bobrow, B. Raphael, "New programming languages for A.I. research," Computer Surveys, Vol. 6, No. 3, September 1974.

[11] G. Boole, "An investigation of the laws of thought," Dover 1958.

[12] S. Mellinkoff, "The differential diagnosis of abdominal pain," McGraw Hill, 1959.

[13] E. Patrick et al, "Review of pattern recognition in medical diagnosis and consulting relative to a new-system model," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-4, No. 1, January 1974.

[14] D. Croft, "Is computerized diagnosis possible?" Computer and Biomedical Research 5, 351-367 (1972).

[15] L. Lusted, "Introduction to medical decision making," Charles C. Thomas, Springfield, Ill. 1968.

[16] G. Gorry, G. Barnett, "Experience with a model of sequential diagnosis," Computer Biomedical Research, Vol. 1, pp 490-507, May 1968.

[17] P. Wortman, "Medical diagnosis: An Information Processing Approach," Computer Biomedical Research, Vol. 5. pp. 315-328, August 1972.

[18] R. Ledley, "Practical problems in the use of computers in medical diagnosis," Proc. IEEE, vol. 57, November 1969.

[19] G. Mandler, "Organization and memory," in "The Psychology of learning and motivation," vol. 1, pp. 307-372, Academic Press, New York 1967.

[20] C. Hewitt, "Procedural embedding of knowledge in PLANNER," Proc. 2nd International Joint Conference on Artificial Intelligence, pp. 167-184, 1971.

[21] R. Simmons, "Natural language question answering systems," CACM, Vol. 13, No. 1, pp. 15-30, 1970.

[22] J. Rulifson, J. Derksen, R. Waldinger, "QA4: A procedural calculus for intuitive reasoning," Technical Note 73, SRI, November 1972.