Medical Consultation Systems: Some Issues Concerning User Interface

Dissertation submitted to the Jawaharlal Nehru University
in partial fulfilment of the requirements
for the award of the Degree of
MASTER OF PHILOSOPHY

HAMID TUAMA MUBARK

138p.

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI - 110067

1984

CERTIFICATE

The research work presented in this dissertation has been carried out under the supervision of Dr. R. Sadananda, Associate Professor, School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi.

The work is original and has not been submitted, in part or full, to any other University for the award of any other degree or diploma.

HAMID TUAMA MUBARK
Student

R. Sadananda

DR. R. SADANANDA
Supervisor

N. P. Mukherjee

PROF. N.P. MUKHERJEE
Dean,
School of Computer and
Systems Sciences,
Jawaharlal Nehru University,
New Delhi.

ACKNOWLEDGEMENTS

I am greatly indebted to Dr. R. Sadananda, my supervisor, for introducing me to the subject. Under his able guidance I had the full liberty to work uninterrupted to present this dissertation in time. His valuable suggestions, encouragement and timely help were a source of inspiration for me.

I am obliged to Prof. N.P. Mukherjee, Dean, School of Computer and Systems Sciences, for his kind help.

I must also acknowledge the valuable suggestions given to me by Dr. G.V. Singh.

A word of thanks is due to Mr. C.A. Thakur, the previous Section Officer, and Mr. Jagdish Lal, who now holds the same office at our School.

My thanks are due to my friends, Mr. R.N. Singh, without whose help, this dissertation would not have been ready by now, and Mr. Rajeshwar Basa, for his help in locating the right books and journals.

I express my sense of gratitude to Mr. Atul Narang, Indian Institute of Technology, Delhi, Mr. K.V.L. Nagendra, R&D, Hindustan Instruments Limt., Delhi, and Mr. Anil Narang, Regional Engineering College, Warangal,

for their advice and good ideas that helped in preparing this dissertation.

I also take this opportunity to thank the officials and staff of National Informatics Centre, New Delhi for allowing me to have access to the CDC CYBER 170/730 Computer facility, and also the staff at the libraries of Jawaharlal Nehru University (Science), and Indian Institute of Technology, Delhi.

Thanks are due to Mr. S.K. Sapra for his diligent and immaculate typing of this manuscript.

This acknowledgement would not be complete without expressing my deep sense of gratitude to the Ministry of Education (Cultural Department), Govt. of India, for the award of the General Cultural Scholarship to pursue my M.Phil/Ph.D program.

HAMID TUAMA MUBARK

CONTENTS

CHAPTER - 1

INTRODUCTION

A simple and general definition of a computer may be expressed in three words: information processing machine. Information is the data ("data is a term that refers to information, which may consist of numbers, words, symbols, or combination of these" [16]) that have been organized into a meaningful sequence. Computer takes information as input and gives information as output. It is much like a washing machine that takes dirty clothes as input and gives clean clothes as output.

During early 1950's, even as the world was just beginning to discover the existence of computers, "a few dreamers" (as B. Raphael [31] puts it) were thinking about the distant future of such devices. He quotes from a famous paper published in 1950 by A.M. Turing, that states :

> "We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best

> sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named,etc. Again I do not know what the right answer is but I think both approaches should be tried."

Now, we are living in the 34th year after A.M. Turing put his suggestions. Today, computers have started invading the lives of practically every one. They are used in medicine, education, control systems, fine art, airlines, law enforcement, printing and publishing, and in homes.

One would put questions as to : "Can machines think?" and "Is it possible to speak of machine understanding?" A field of science called "ARTIFICIAL INTELLIGENCE" may give an answer to these questions.

1.1 ARTIFICIAL INTELLIGENCE

Artificial Intelligence is the part of computer science that is concerned with the design of intelligent computer systems. Intelligent computer systems are those systems that exhibit the characteristics we associate with intelligence in human behaviour, characteristics like learning, reasoning, understanding language and solving problems.

More clearly, the practical character of artificial intelligence is nicely defined by M.L. Minsky: 'Artificial Intelligence is the science of making machines do things that would require intelligence if done by men'[30].

The first success in developing programs that exhibit intelligence was a program by A.L. Samuel of IBM. In 1961, he developed a program to make a computer play checkers. The program could even beat Samuel, its creator. Another program that was developed could play chess. These programs work by searching through a space of possible moves, that is, considering the alternative moves and their consequences several steps ahead in the game, just as a human players do. Thus, computers can be taught how to <u>think</u>.

The two main goals of artificial intelligence are 41 : (a) to make computers more useful; and (b) to understand the principles which make intelligence possible. Computer scientists and engineers concerned with the first goal, need to know how artificial intelligence can help them in the solution of difficult problems. Psychologists and philosophers concerned with the second goal, think that understanding

artificial intelligence is a way to study natural intelligence. Artificial intelligence researchers believe that intelligent machines should help men in all fields. Just as mechanical machines have been helping men in their physical activities, these intelligent machines would help men in their intellectual activities.

Many programs were designed, using artificial intelligence techniques, to perform the following tasks:

1) Proving theorems: These programs prove assertions by manipulating a database of facts.

2) Learning: The ability to learn is the most significant aspect of human intelligence. Programs that can learn from examples, from their own performance and from being told about the knowledge domain are attempted.

3) Automatic programming: In this field the attempt is to develop systems that can write computer programs.

4) Robotics and Vision: Programs that manipulate robots are developed. Most industrial robots are blind, but some see through a TV Camera. Processing visual information is another very active area of

artificial intelligence research that can make robots see their environment. Programs have been developed that can recognize objects and shadows in visual scenes, and can even identify small changes from one picture to the next, for example, for aerial reconnaissance.

5) Natural language understanding: Using artificial intelligence techniques, programs have been written that answer questions posed in English, that translate sentences from one language to another and that acquire knowledge by reading textual material and building an internal data base. As an example, LUNAR (Woods, 1973) was a natural language program designed as an interface to a database that described the moon rock samples brought back by the Apollo astronauts. LUNAR could answer questions posed in English.

6) Problem Solving: Humans always solve a problem by finding a way of thinking about it that makes the solution easy. Artificial intelligence programs must be told how to think about the problems they solve. Techniques like looking ahead several moves and dividing difficult problems into easier subproblems evolved into the fundamental artificial intelligence techniques of

search and problem reduction [10]. Using these techniques, programs that solve puzzles and play games like chess were developed. To solve a problem the choice of problem representation is most important. Consider the classical problem of artificial intelligence called "monkey and bananas" that involves a monkey, a box, and a bunch of bananas hanging from the ceiling in a room, the distances are such that the monkey is unable to get the bananas unless he is standing on the box. To solve this problem, knowledge about boxes, monkeys, bananas, and distances must be encoded in the program. For example, the program must be told that boxes may be pushed, that pushing has certain effects on a box and on the individual doing the pushing, that boxes may be climbed on, etc. An intelligent program then can deduce from this knowledge domain, that the best plan for the monkey is to push the box under the bananas, to climb on the box, and finally to grasp the bananas.

7) Expertise: The most useful applications of artificial intelligence techniques are found in a very recent area called <u>expert systems</u>. A user interacts with an expert system in a <u>consultation dialogue</u>, just

as he would interact with a human who had some type of expertise. Through consultation dialogue, the user explains his problem, gets responses and solutions, and can ask questions about the proposed solutions. Current experimental expert systems have achieved high levels of performance in consultation tasks, like chemical and geological data analysis, computer systems configuration and medical diagnosis.

1.2 EXPERT SYSTEMS

Among the most significant successes in the field of artificial intelligence has been the development of powerful new computer systems known as "expert" or "knowledge-based" systems [2]. Expert systems are programs that have been designed to represent and apply factual knowledge of specific areas of expertise to solve problems.

Expertise consists of knowledge about a particular domain, understanding of domain problems and skill for solving some of these problems. These systems provide expert level solutions to given problems within their domain of knowledge.

In what ways do these expert systems differ from conventional data processing systems? An answer to

this question will give us an idea as to how these systems could behave much like an expert in problem-solving environment.

Traditional programs (also called algorithm-based programs) tend to reflect the orderly domain for which they were written; they are deterministic and possess no redundancy. For any given input there is a single computational path that is always followed and there is a single mechanism capable of producing the correct output for that input. Also, a conventional program usually exhibits a sharp distinction between code and data, i.e. between the procedures for how to manipulate structures and the structures themselves. Since only the structures are accessible to the program, it thus lacks any ability to reason about or explain the techniques and mechanisms that it employs.

In contrast to these traditional data processing programs, knowledge-based systems are completely dependent on knowledge in solving problems. Knowledge involves facts about the task domain, and heuristics or rules of thumb that guide the use of knowledge to solve problems in the domain. Most current expert systems are divided, not into code and data, but into a corpus of

knowledge and a comparatively simple mechanism for applying the knowledge in an opportunistic way to solve problems. The power of the system does not come principally from this knowledge application mechanism, but from the richness, pertinence and redundancy of the knowledge itself.

The expert systems are also different from the broad class of artificial intelligence tasks in several respects. First, they perform difficult tasks at expert levels of performance. Second, they emphasize domain specific problem-solving strategies. Third, they employ self-knowledge to reason about their own inference processes and provide explanations or justifications for the conclusions reached.

Expert systems solve problems in the following categories [2]: interpretation, prediction, debugging, design, planning, monitoring, diagnosis, repair, instruction, and control.

Well known practical expert systems are: PROSPECTOR in geology, R1 in configuring customer requests for VAX computer systems at DEC, DENDRAL in chemical structure elucidation, and CADUCEUS and PUFF in disease diagnosis.

When evaluated, some expert systems were found to be performing better than experts themselves. For example the performance of MYCIN was judged as good as or superior to that of some medical experts.

1.3 MEDICAL CONSULTATION SYSTEMS

Computer-based medical consultation systems are the type of expert systems that deal with solving domain specific medical problems.

There are two reasons for having medical consultation systems. First, medical knowledge is rapidly growing and changing, and often, especially at the primary care level, only a subset of available medical knowledge is put to actual use. Computer-based medical consultation systems would make available the most advanced and complete clinical knowledge, especially in areas where experts are difficult to find. Second, in medical consultation systems the process of reasoning and use of medical knowledge is made explicit. Hence, these systems will aid in the training of clinicians.

Some of the experimental medical consultation systems are: MYCIN (Shortliffe, 1976) in the domain of infectious diseases, CASNET (Weiss, et.al., 1977) in

the domain of glaucoma, PIP (Szolorits and Panker, 1978) in renal diseases, and INTERNIST (Pople, 1975) in the area of internal medicine.

To be accepted by the clinicians, the medical consultation systems provide users with explanations about their conclusions. They answer questions and show the path of their reasoning process. As clinicians did not know computer languages and would not agree to learn these artificial languages, medical consultation systems were forced to accept natural language questions upto a limited extent.

Medical consultation systems may be made very powerful and perform accurate diagnosis, but for a user these systems are of no use if he cannot interact with them.

1.4 INTERFACING

In general, the most important factors affecting the computer usage are: reduced cost, increased functionality, improved availability and servicing, and perhaps the most important one is user interface. The first three factors alone are necessary, but not sufficient for widespread usage. For example, reduced cost will allow people to buy computers, but improved user

interface will allow people to effectively use computers.

If computer-based medical consultation systems are to be accepted and attract a wide range of users, their success will be largely dependent on user interfaces. The interface here is of paramount importance since the user is a person not quite acquainted with the computer, e.g., the user is a clinician or the patient himself.

It follows then, that it is important to study the issues concerning the interfacing between a user and a medical consultation system. Specifically, the problems that are encountered in the interaction (consultation) process are of our interest in this dissertation.

Let us first identify as to what we mean by interface and what exactly happens in an interaction session between the user and the consultation system.

Interface is a communication boundary between two systems [26]. In this definition, the user is representing one system, while the other is the medical consultation system, and communication refers to the exchange of information between the two interacting systems. During the process of a typical consultation

session (Appendix A) we may describe interaction between the system and a user as follows :

1. System asks for data - e.g. laboratory data and other personal information about the patient.

2. Answers for system's questions are to be entered by the user.

3. User may ask system for explanations as to why such questions are asked.

4. User may ask the system about the path of reasoning in diagnosing a disease or prescribing a therapy.

5. Entering (addition) of new knowledge to the knowledge base of the system or changing existing knowledge, and

6. User may ask general questions for the sake of learning about the domain of medical of the system, i.e. using the system for educational purposes.

For effective exchange of information in the above interaction both systems must (a) be able to understand each others messages, (b) be able to generate responses in reply, and (c) have a common language for communication. It is also required that there should be some means of ensuring that the message received by

any of the two systems is correct and means what the source system intended to mean.

In human communication, people adjust to each other, tolerate errors, communicate freely through unrestricted dialogue, pose questions to clarify ambiguities, give responses with almost no time, and people try to phrase their questions and responses in a very brief and clear form.

As an accepted dialogue partner, a user would want the medical consultation system to have all the above characteristics of human communications. In practice, the following <u>problems of interface are encountered in using medical consultation systems</u> : (1) The problem of language adaptation: for a user the most efficient way of expressing himself is through the natural language. Teaching clinicians and patients computer languages and training them to use these artificial languages effectively, while interacting with the systems, are very expensive jobs and most inefficient. A better approach would be the teaching of natural languages to systems instead of teaching users the computer languages. (2) response : short and well-formed responses are easy to understand. Systems must be able to frame their responses and

questions clearly and in short form. Response time is a very important factor in a dialogue. Consultation systems are to provide responses in very short time to avoid boredom or impatience of an on-line user who is waiting for an answer. (3) The behaviour of the system towards the user : medical consultation systems must provide a friendly atmosphere for a user. Users would not be inspired by partners that force them to act in a restricted manner. The system should provide help when asked or whenever it perceives that a user is in need for help. (4) Tolerance : people adjust to each other during a dialogue. They tolerate grammatical and spelling errors. Systems must be designed in such a way as to be able to tolerate grammatical and spelling errors.

For example, misspellings are a real problem of interface that cause confusion and frustrate a user interacting with the medical consultation system through a terminal. This reality can be seen in the following interaction between MYCIN and a clinician, figure (1.1). In figure (1.1), MYCIN'S requests are numbered, while user responses are in upper case and follow double asterisks [12].

```
------ PATIENT - 538 ------

1)  Patient's name :

**  PT 538

2)  Age :

**  34 YEARS

3)  Sex :

**  MAEL

= MALE
```

figure(1.1)

In the interaction above, fig (1.1), as a response to MYCIN'S third request, the user enters "MAEL" which is wrongly spelled, its correct spelling is "MALE". For a moment, let us think of MYCIN to have no spelling correction facility. Then, the MYCIN system would have simply rejected the physician's answer, as it does not understand "MAEL" because it (MALE) is not in the system's dictionary. But our clinician is very sure of his answer ! This will confuse the user. Thus, the user may think of changing his response or he may feel that MYCIN is not friendly or he may feel that MYCIN could be sick too ! (the user did not notice his mistake even after rejection). However if the user sees his mistake, he would hurriedly

retype his answer or if he was quite experienced in using the system, he would call the line editor and correct his response without retyping the entire sentence (response).

Fortunately, MYCIN has an ability to correct misspellings. MYCIN calls its spelling correction routines. It inspects if the word is a misspelling of another in its dictionary. MYCIN detects and corrects the error (transposition of adjacent letters, i.e. "L" and "E") and prints "=WALE" to bring the attention of its user to the error committed (for verification). MYCIN continues (as user agrees to the correction) the consultation as if there was no misspelling.

Thus, after understanding the problem of spelling correction and its importance in an interface with medical consultation systems, we chose to design an interactive spelling correction program as an implementation of one of the problems of interface.

Our interactive spelling correction program is a conventional program designed to correct only four classes of single error that account for 80 percent of the spelling errors [40].

(a) one letter wrong

(b) one letter missing

(c) one letter extra

(d) Transposition of two adjacent letters.

It is also assumed that there is exactly one error in the word to be corrected and that the error arises from one of the above-mentioned four causes.

Since response time is the most important factor in the user interface with interactive systems, we chose hash files for dictionary storage to provide direct look-up. The hash table entry is the head of a chain of entries (words) each with the same hash value.

To correct a word, that is suspected to be a misspelling of another in the dictionary, the program first searches the dictionary for the given word. If the given word is found in the dictionary, the program gives the message "word is correct" and it terminates. If no match for the given word is found in the dictionary, then the program inspects the given word for the assumed single error by using its four correction routines; EXTRA, TRANSPOSE, WRONG and MISSING, one at a time. The matching words, if any, are put on a MATCHLIST.

If inspection results in more than one matching, all the words in the MATCHLIST are displayed and the program asks the user to pick up the correct word that he was intending. If exactly one matching word is found on the MATCHLIST, the program displays that single word and asks for user's approval. If no matching was found, the given word is rejected and the user is informed that his word is not a valid one, and the program terminates.

CHAPTER - 2

REVIEW OF MEDICAL CONSULTATION SYSTEMS

Like other fields, the medical field had been greatly influenced by the computer revolution during the last two decades. Computers are used in: hospital accounting systems, medical records maintenance, Laboratory and Pharmacy data systems, hospital information systems, computer-aided instructions for medical education, biomedical engineering and clinical decision-making support.

The aim of this chapter is to review the work done in medical consultation programs, that is, decision-making support programs that can give explanations of their line of reasoning and can conduct a dialogue with physicians.

There are two factors that motivated the development of computer-based medical consultation systems; first, to provide the society with reliable and thorough diagnostic services, perhaps even at a reduced cost (in view of the rapid decrease in hardware cost and increasing speed of computation in computers, and the rising unavailability of medical experts). There are

some tasks that computers can perform more rapidly and accurately than the clinician can such as calculating doses of medicines. "It has been observed (Ledley and Lusted, 1959) that most of the errors made by clinicians are errors of omission" [11]. This is due to the fact that in deciding what the disease is, the physician does not go through all possibilities and this results in wrong diagnosis. On the other hand, a computer program provided with the needed data about the diseases and the patient's case can give a highly accurate diagnosis by searching all possibilities within its domain.

Second motivation is the study of the cognitive process in computer science. Cognition refers to the acquisition and use of knowledge. It includes thinking, memory, problem solving, decision making, intelligence, and epistemology [8]. Diagnosis as a cognitive process involves a well-organized knowledge base, a number of human experts in the specific medical domain, and a highly developed medical taxonomy.

2.1 EARLY USE OF COMPUTERS IN MEDICAL DECISION MAKING

Medical decision making process involves; data gathering - laboratory data and other personal infor-

mation about patient; diagnosis - using the gathered data to determine the illness; and treatment recommendations. In the early sixties, computer programs were designed to aid the medical decision making process, mentioned above, by providing physicians with the Laboratory data and collected personal information about the patient under treatment, performing the diagnostic process by selecting the disease and finally aiding in the selection of proper drugs and customization of drug doses.

These early decision making support programs (that is, before the evolution of expert Medical Consultation Systems using artificial intelligence techniques) are classified into three groups based on the method of computation used by these programs [6]:

1. Data Retrieval Systems :

Large amounts of data for several patients are stored in a computer-based information storage system. Data retrieval systems retrieve data from these information storages and provide the data to physicians to aid physicians in the decision-making process. The stored information in the storage system may include physical parameters of patients, diagnosis procedures,

treatment plans and responses to therapy. For example, physician is supplied information on how previous patients with a similar disease as that of the present patient (patient under treatment) have responded to therapy. This helps the physician select the best treatment plan for his patient.

2. Programs Using Numerical Computations :

Computer programs that perform calculations and their interpretations are very important for physicians in providing results of some mathematical formulae that are usually difficult to remember or perform by physicians.

For example, programs were written to assist physicians in the classification and management of electrolyte and acid-based disorders; here the relationship of blood pH to variables such as kidney function and electrolyte levels is well characterized by formulae that utilize the numerical values of blood gas and other laboratory tests.

Programs written by Bleich and Schwantz provided calculations of patient parameters and information regarding patient's status. They also provided physicians with a list of etiologies and gave literature

references to guide the physician, (Bleich, 1969, 1971, 1972) and (Schwantze, 1970) [6].

Numerical calculations using well defined formulae were used in the customization of drug doses once the agent to be used had been selected. Programs in this field are those that involve the selection of a digoxin regimen for a patient with heart disease (Sheiner, 1972; Jelliffe, 1972; Peck, 1973) and those that help physicians decide on insulin doses for diabetics (Bollinger, 1973) [6].

3. Programs Using Statistical Techniques :

Decision trees and Baye's Theorem are widely used statistical techniques in these programs. Decision trees cannot be adjusted in cases when unexpected findings or unavailable test results take place. This is due to the fact that trees are nondynamic. Moreover modification of trees is very difficult because of the subtle interrelationships within such reasoning networks, (Horner, 1972a; Sletten, 1973; Bodman, 1966; Button, 1973; Koss, 1971; Mayer, 1973) are examples of programs that are at least partially dependent upon tree structured decision pathways [6].

Bayesian probability theory is the most commonly used statistical technique in medical decision aid programs. According to the probability theory to every assertion A one can assign a probability value P(A). P measures the degree to which P is believed to be true, where P = 1 if A is known to be true, and P = 0 if A is known to be false. The degree of belief in A changes as new information is obtained. If, for example, the new information is $B \longrightarrow A$, then P(A) denotes the intial or prior belief in A and the conditional probability P(A/B) denotes the revised belief in A upon learning that B is true.

In diagnosis, if A is taken as the CAUSE and B as the EFFECT, it is viewed that the computation of P(A/B) is an inference that the CAUSE is present upon observation of the EFFECT. Another computation is that of P(B/A) which is the probability of observing the EFFECT B when the CAUSE A is active. The calculation P(B/A) is easier than that of P(A/B) due to the fact that; "The probability of a disease, given a symptom, may vary with time and place, while the probability of a symptom, given a disease, remains invariant (Lusted, 1968)" [12]. Thus Baye's rule is commonly employed to

compute P(A/B) from P(B/A). To employ the Bayesian Theory two important things are needed to be computed: the prior probability P(A) and the likelihood ratio (L) defined by :

$$L = \frac{P(B/A)}{P(B/\sim A)}$$

where $P(B/\sim A)$ is the probability of observing effect B when cause A is absent.

So far it is considered that every cause A has one effect B. But when cause A has n plausible effects B_1, B_2, B_3, ... B_n, then the likelihood ratio L is calculated as follows :

$$L = \frac{P(B_1 \& B_2 \& B_3 \& \ldots B_n/A)}{P(B_1 \& B_2 \& B_3 \& \ldots B_n/\sim A)}$$

*In 1964, Warner et. al. introduced a computer program that aided in the diagnosis of congenital heart diseases (Warner, 1964). Data had been gathered for several hundred patients with congenital cardiac malformations. As a result, all the conditional probabilities needed for the Bayesian Theorem could be computed. The program accordingly classified new patients with an

accuracy similar to that of cardiologists" [6].

The disadvantages of programs using statistical techniques like Bayesian Theorem are; firstly, due to the exhaustive patient data collection process needed for these programs to perform diagnosis, problems like errors in data and unavailability of data are very common, making the use of these programs a difficult tasks. Secondly, errors occur due to the change of symptoms of diseases from time to time.

Besides Baye's Theorem other statistical diagnostic programs used a technique called pattern recognition. Pattern recognition here, is the method that attempts to extract the most characteristic features of each diagnostic category, rather than trying to discriminate directly between categories. The patient then is said to belong to a specific category with which the patient's data shares the most features. This pattern recognition technique needs huge data about a large number of patients with various diseases.

2.2 WHY MEDICAL CONSULTATION PROGRAMS AS DECISION MAKING AIDS

As mentioned in the preceeding subsection, the early statistical and numerical computation programs had only provided the physicians with results such as data,

numerical computation results, disease selection by diagnosis, selection of drugs or treatment for particular disease or customization of drug doses, etc., but did not provide an explanation as to how such results were obtained or show their reasoning path or allow a dialogue to let physician ask questions and know about the domain of knowledge in the program. Therefore, this left physicians unsatisfied with such performance, because physicians were willing to accept the advice of the program only if they are able to understand the decision steps that the program has taken in reaching the particular result.

For systems to explain their reasoning and show their decision steps and carry a dialogue with physicians, they need to understand questions and generate answers and explanations. Questions and answers constitute a mechanism by which physicians and systems can communicate with each other.

These characteristics, explanation and dialogue, were not available in the early statistical and numerical programs like Bayesian and decision tree programs. Hence, the need for medical consultation systems as decision-making aids was very necessary for physicians.

"G.A. Gorry became aware that the purely statistical programs had three failings that are major impediments to physician acceptance of the systems" [6]. Firstly, the programs have no real "understanding" of their problem area. Gorry explains this point as follows (Gorry, 1973a) :

"There are several approaches to inferring renal function and assessing whether it is stable or changing. This determination is very important in diagonsis and in choosing management strategies. From the experts, it is possible to obtain the procedure by which they infer a value from renal function. Further, many statements about the interpretation of changes in renal function can be made. To capture the knowledge embodied in these statements, some computer realization of the concept of renal function must be developed."

This "concept formation" that Gorry feels is needed can be achieved by using knowledge representation techniques in artificial intelligence.

Secondly, the programs cannot conduct a dialogue with the user. Even if one gives the traditional programs an 'understanding' of their domain, they will still be unable to communicate with the user because they have

no mechanism for discussing their knowledge with the user. Physicians are unhappy with programs that give diagnosis and numerical values but are unable to answer questions about how diagnosis and results were arrived at. Gorry therefore calls for the development of Natural Language Interfaces to permit discourse between physicians and diagnostic programs. Once again artificial intelligence provides a natural environment for examining this requirement.

Thirdly, physicians need programs that give explanations for their advice. This is to justify their reasoning process. To achieve this, programs should understand their reasoning process and should be able to generate explanations in a language that is understood by the user. Here, a natural language will be most convenient for the user to understand explanations and communicate with the system. "Gorry's group has therefore worked on developing knowledge representations and language capabilities that will heighten the acceptability of a system such as their acute renal failure program (Gorry, 1973b)" [6].

2.3 MEDICAL CONSULTATION SYSTEMS : STATE OF THE ART [11]

The state of the art in medical consultation systems will be represented in the following paragraphs

which describe systems in a very brief way. Some of the programs will be discussed in the next subsections of this chapter from the point of view of their reasoning and knowledge representation techniques.

MYCIN (Shortliffe, 1976): This system was designed to provide consultative advice on diagnosis and therapy for infectious diseases. MYCIN'S knowledge base consists of 200 rules. To solve a problem MYCIN tests the conditions of a rule against available data or requests data from the physician. "When a panel of experts evaluated the performance of several different agents, including medical experts, interns and MYCIN, MYCIN's performance was judged as good as or superior to that of all others" [2].

CASNET (Weiss, Kulikowski, and Safir, 1977): It was developed at Rutgers university for performing medical diagnosis. Its application is in the domain of glaucoma. In this system the disease is represented as a dynamic process (not static state) which is modeled as a network of causally linked pathophysiological states. It determines the pattern of pathophysiological causal pathways present in the patient and it identifies this pattern with a disease category. After identifying the

category of the disease, it prescribes the most appropriate treatment.

INTERNIST (Pople, 1975): It is a consultation program in the domain of internal medicine, developed jointly by H. Pople, Computer Scientist, and J. Myers, a specialist in internal medicine. This program gives only diagnosis and no therapy. Manifestations of diseases in a patient (Laboratory data, physical signs, and symptoms) are presented to the system. The diagnosis involves a list of diseases that account for the manifestations. Using information during the course of the consultation, the program is able to discriminate between competing disease hypotheses.

PIP (Present illness program): It was developed at M.I.T. by Szolovits and Panker in 1978. This program can decide the present illness of patients with renal (Kidney) disease. A patient is asked about his main complaint. This main complaint then becomes the focus of the consultation. The program requires very low cost information about the patient, like patient history, physical examination and routine laboratory tests.

Digitalis Therapy Advisor (Silverman, 1975; Swartout, 1977): This system advises physicians on the

administration of the drug digitalis. It does not perform the diagnostic process. It determines an appropriate treatment regimen and the subsequent management of that treatment for patients known to require digitalis.

IRIS (Trigoboff and Kulikowski, 1977): It is a program that is used as a tool for building and experimenting the consultation systems. It is used by computer specialists in collaboration with medical domain experts in building and experimenting systems with different representations of general medical knowledge, clinical strategies, and modes of interaction. IRIS employs both semantic nets and production rules techniques in its knowledge representation.

EXPERT (Weiss and Kulikowski, 1979): This system was developed at Rutgers university to help investigators design and test consultation models in medicine.

PUFF (Freiherr, 1980): It is a pulmonary function program. It is an EMYCIN (system that contains all of MYCIN except its knowledge of infectious diseases).

There are several other experimental consultation programs under development, including :

1. HODGKINS, a system for performing diagnostic planning for Hodgkins disease (Safrans, Desforges, and Tsichlis, 1976) ;
2. HEADMED, a Psychopharmacology advisor (Heiser, 1978) ;
3. VM, an intensive-care Monitor (Fagan, 1979) ;
4. RX (Blum and Wiederhold, 1978) ; and
5. ONCOCIN, a program for Monitoring the treatment of oncology out-patients on experimental treatment regimens (shortliffe. et.al,1981).

2.4 KNOWLEDGE REPRESENTATION, REASONING, EXPLANATIONS AND KNOWLEDGE ACQUISITION

KNOWLEDGE REPRESENTATION :

"A representation is a set of conventions for describing the world. In the parlance of aritificial intelligence, the representation of knowledge is the commitment to a vocabulary, data structures, and programs that allow knowledge of a domain to be acquired and used," [12].

Among human beings, the ability of a person to behave with intelligence is always described in terms

of the knowledge the person has. Thus, the focus of artificial intelligence researchers in designing systems that can perform "intelligent" tasks was concentrated on developing data structures for storing information in computer programs and development of procedures that allow intelligent manipulation of these data structures to make inferences.

In medical consultation systems, the medical knowledge needed by the system to perform its tasks is of two types : a) knowledge about diseases which involves taxonomy of diseases, manifestations, causal mechanisms, and diagnostic procedures extracted from the medical experts, and b) knowledge about the patient which includes current medical history and therapies.

Artificial intelligence techniques like production rules, predicate calculus, frames and semantic nets are used in the representation of this medical knowledge. These representation formalisms could not be used directly because of the inexact nature of medical knowledge, i.e., incomplete certainty of medical facts. Thus, factors or weights are used to augment their representations to provide some way of expressing strength of belief or strength of association.

For example, the medical knowledge in MYCIN is represented as a set of production rules. These rules are augmented by certainty factors to express the strength of "belief" in the conclusion of a rule, assuming that all the premises are true. The following rule is from MYCIN knowledge base :

IF : 1) THE STAIN OF THE ORGANISM IS GRAM POSITIVE, AND
2) THE MORPHOLOGY OF THE ORGANISM IS COCCUS, AND
3) THE GROWTH CONFORMATION OF THE ORGANISM IS CHAINS

THEN: THERE IS SUGGESTIVE EVIDENCE (.7) THAT THE IDENTITY OF THE ORGANISM IS STREPTOCOCCUS

In this rule the medical expert indicates a 70% belief that the conclusion was valid.

A causal-net work representation is used by CASNET in which each CAUSES link is associated with a number representing the strength of causality.

In INTERNIST, knowledge (Taxonomy of diseases) is stored in a huge tree. Each node in the tree represents a disease. A list of manifestations is associated with each disease node. Manifestations are

augmented with numerical weights reflecting the strength of association between them and the diseases.

REASONING :

"Clinical reasoning involves weighing different pieces of evidence for particular hypothesis" [1]. The threshold technique is used by most of the systems in their reasoning process. If the value of present threshold (defined by the medical expert who builds the knowledge base) is exceeded for a hypothesis, then that hypothesis is believed to be true. A summary of some reasoning mechanisms and their use by the different systems is presented in the following paragraphs.

A rule-based mechanism is used in MYCIN system in its reasoning process to determine parameters like patient's infections and the causative organisms. The parameter is said to be known if the certainty factor of the most highly supported hypothesis exceeds the threshold. The premises of a rule are considered true if the combined value of the associated certainty factors exceeds a predefined threshold. If several rules contribute to a conclusion about a parameter, their certainty factors are functionally combined to form a composite certainty factor for this conclusion.

In CASNET, a status measure is associated with each state in the causal net work. In the semantic network in CASNET, weights are propagated in both forward and backward directions depending on disease causality. A state is considered conformed if its status exceeds a specified threshold.

In INTERNIST, figure (2a) shows the knowledge representation on which reasoning is done.

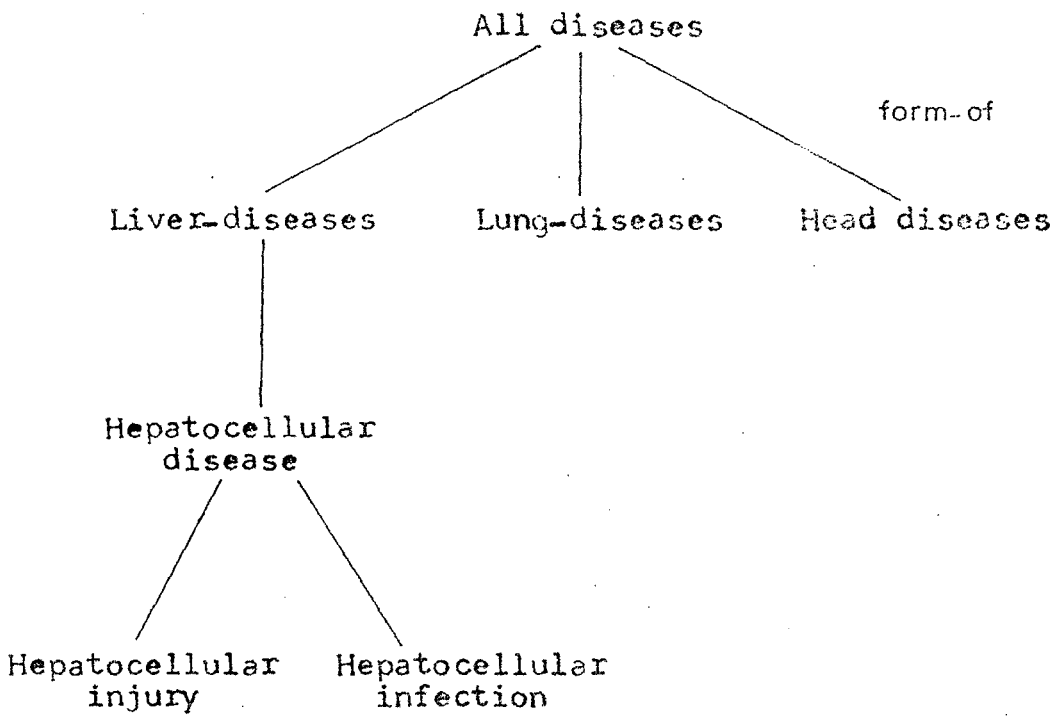


figure (2a). INTERNIST's Disease Tree [11].

A list of manifestations is entered at the begining of a consultation, then each manifestation will invoke one or more nodes of the disease tree. For each invoked disease node a model is created which consists of four lists :

1. Observed manifestations that this disease cannot explain,
2. Observed manifestations that are consistent with the disease,
3. Manifestations that should be present if this disease is the correct diagnosis but have not been observed in the patient, and
4. Manifestations consistent with this disease but that have not yet been observed in the patient.

At this stage, very few of the terminal nodes will be invoked, so the program must ask for further information. Program will focus on a disease area and formulate a problem and then start asking questions.

If a disease model explains a manifestation it gains positive score, and it gains a negative score for the manifestations that it can not explain. Also a disease model receives bonus if it is linked causally to a disease that has already been confirmed.

The models of diseases are partitioned into two sets: a) the top-ranked model and the diseases that are mutually exclusive to it (alternatives). b) the diseases that are complementary to the top-ranked model. From figure (2a) on last page: if the top-ranked node is hepatocellular injury, then other invoked liver diseases will be alternatives to it, while lung or heart diseases will be complementary. After this partitioning, depending on the number of candidate diseases in the problem set, the system will follow one of the following strategies.

If there are more than four alternative hypotheses, the system attempts to rule out as many as possible. This is done by selecting the manifestations that strongly indicate a disease and if these manifestations are not present, the disease is ruled out. If the number of hypotheses is between two and four, then manifestations strongly indicating one disease and manifestations weakly indicating another disease are selected, and the system asks questions to discriminate between these two diseases. When the difference between the scores of the top two disease hypotheses reaches a predetermined threshold level the top ranking disease is confirmed.

If there is only one hypothesis, then the questions that have a good chance of confirming the disease are asked. When a disease is confirmed, its manifestations are marked "accounted for", and bonus scores are given to previously manifested diseases that are causally linked to this disease.

EXPLANATIONS :

The most important factor for physicians acceptance of medical consultation systems is explanation. By explaining the steps of its reasoning process, the system makes it possible for physicians to understand and be able to detect and correct wrong steps in the reasoning process in the particular diagnosis.

So far, MYCIN and the Digitalis Therapy Advisor have offered good explanation capabilities. MYCIN gives English translation of the rules used in the diagnosis. The Digitalis Therapy Advisor gives the English Translation of its reasoning steps. While the other consultation systems could give their scores of hypotheses, they could not give explanations of the methods by which they arrived at these scores.

For example, in systems that ask user for information from which to draw conclusions, the user

might wish to check the reasoning of the system by asking "Why do you want to know X?" The response of the system would be an explanation or a translation of the rule or related rules to the current question (information) that was needed by the system. The answer of the system may be in the form "So that the value of Y can be determined."

The following is an example of a question and answer between MYCIN and the physician :

MYCIN : What is the age of the patient ?

PHYSICIAN : Why do you ask the age of the patient?

MYCIN : Relevant rules: Several rules including Rule 118 Rule 135 Rule 137 Rule 138 Rule 150.

Physician : Rule 150

MYCIN : Rule 150

IF : 1) THE THERAPY UNDER CONSIDERATION IS SULFISOXAZOLE, AND
2) THE AGE (IN YEARS) OF THE PATIENT IS LESS THAN 58, AND
3) THE PATIENT IS JAUNDICED

THEN : IT IS DEFINITE (1) THAT SULFISOXAZOLE IS NOT A POTENTIAL THERAPY FOR USE AGAINST THE ORGANISM.

KNOWLEDGE ACQUISITION :

Knowledge acquisition is the transfer of medical expertise from the medical expert to the consultation system. The medical expert is a physician specialized in a domain of medical knowledge on which the system would give consultative advice. The medical expertise is a collection of specialized facts, procedures, and judgmental rules about the specified domain in which the medical expert is specialized.

MYCIN consultation system has this ability of

THEN : 1

and 2

After this frame is filled by the expert physician, MYCIN translates the premises and actions into its LISP representation and requests an associated certainty factor if necessary. MYCIN translates the rule to English and displays its understanding of the rule, asking the expert if MYCIN'S understanding was correct,and if the rule means what it was meant to be. If there are any mistakes, the premises and the actions are corrected and the process continues till the expert approves MYCIN understanding of that rule. After approval by the expert, the rule is added to the corpus of rules that constitute the knowledge base of MYCIN.

CHAPTER - 3

PROBLEMS OF INTERFACE BETWEEN USER AND CONSULTATION SYSTEMS

As defined in our introductory chapter, interface is a communication boundary between two systems [26]. In this definition, a user is considered to be representing one system in the communication process, while the other is the medical consultation system. Here, communication refers to the exchange of information between the user and the medical consultation system during a typical consultation session. (Typical consultation session between MYCIN consultation system and a user-physician - is given in Appendix A). During the process of a typical consultation session we can describe interaction between the system and the user as follows :

1. System asks for data - e.g. Laboratory data and other personal information about the patient,
2. Answers for system's questions are to be entered by the user,
3. User may ask system for explanations as to why such questions are asked,

4. User may ask the system about the path of reasoning in diagnosing a disease or prescribing a therapy,
5. Entering of new knowledge to the knowledge base of the system or changing existing knowledge; and
6. User may ask general questions for the sake of learning about the domain of medical knowledge in the system, i.e. using the system for educational purpose.

For effective exchange of information both systems, user and consultation system, must : (a) be able to understand each others messages; (b) be able to generate responses; (c) have a common language for communication. And it is also required that there should be some means of ensuring that the message received by any of the two systems is correct and means what the source system intended to mean.

What are the problems encountered in the interaction process? For a user, the medical consultation system is replacing a human expert-physician in the interaction process. Thus, user's acceptance would be very much dependent on the performance of the medical system which, for the user, has to resemble that of the human expert. In this chapter, our aim is to study

the problems encountered in the communication (interaction) process between the user and medical consultation system. The interface problems range from "user's adaptation to systems programming language", "System's behaviour towards the user", "System's tolerance to errors introduced by the user", to "System's response". Solving these problems would humanize the consultation systems. The success of medical consultation systems will be largely dependent on user interface. The interface here is of paramount importance since the user is a person not quite acquainted with a computer, e.g. our user is a patient or a physician.

3.1. PROBLEMS OF LANGUAGE ADAPTATION

In the past use of computers was limited to highly trained computer specialists. Computers were used, in a small number of specialized fields like scientific computations, engineering and banking with the help of these computer specialists. In the near future computers will be almost as readily available as typewriters or telephones. Future computer users will be people from all walks of life - doctors, patients, lawyers, students, managers, library users

and even housewives. Do we have to teach every one computer languages to enable them to use computers? It would definitely be a waste. Instead of teaching computer languages to people, a preferable approach would be to teach natural languages to computers.

Talking of the importance of natural language in the user interface with fifth generation computers, T. Moto-Oka says, "the larger the number of computer users becomes, the more urgent will become the need for high level conversational language. Natural language is the highest level conversational language and is also one which non-expert users prefer to use" [5]. Thus, equipping computers with natural language would "democratize" computer use.

Health care is the field where every one of tomorrow's society will become a computer user. For a user of medical consultation systems language of interaction with the system will be the most important factor that influences user acceptance of the system. Natural language is the most efficient language for a user in expressing himself. In contrast, using artificial languages (languages that have been invented by people for particular kinds of communication [31]),

the user can express only limited set of ideas. For example, using computer language a user can express easily only those concepts that are important in programming - "IF THIS IS TRUE, THEN DO THAT", "SEE IF THIS IS EQUAL TO THAT". While inquires like - "IF THE GRAMSTAIN OF AN ORGANISM IS NEGATIVE AND IT IS A ROD, DO YOU CONCLUDE THAT IT MAY BE A PSEUDOMONAS?" are not possible to express in an artificial language with ease.

Artificial programming languages are not clear and generally they are confusing. The following example will illustrate the difficulty in using artificial language in communicating with the system [25]:

RS : / TOOTH / , / TRUTH /

This is a typical interactive text editor command in which: RS stands for "Replace String". The effect of this command is to replace the next occurrence of the character string "TOOTH" with the string "TRUTH". We notice that the meaning of this command is not self-evident. The syntax is arbitrary in that it does not follow the conventions of any widely used means of communication.

By contrast a command such as :

REPLACE "TOOTH" WITH "TRUTH"

is meaningful to any English Speaker. Its format is that of a legitimate English phrase and the words are both familiar and highly descriptive of the task to be performed by the command. It is clear, for example, that "TRUTH" will replace TOOTH, and not the other way around.

Before we proceed to the other problems of interface, we should introduce some definition of terms relating to natural language processing as they will be used in the coming subsections of this chapter.

Natural Language : A natural language is the language that is used as the principal means of communication in the day-to-day business of a human society [31]. English, Arabic, French, Hindi, etc., are all natural languages. The main property of these natural languages is that nearly any concept that comes to mind can be conveyed to another person through a common natural language. On the other hand, artificial languages are those languages that have been designed to be highly

expressive over a limited range of ideas [4].

Grammar Grammar of a language is a scheme for specifying the sentences allowed in the language, indicating the syntactic rules for combining words into well-formed phrases and clauses [10]. The grammar is used in parsing the language sentences.

Parsing : It is the "delinearization" of linguistic input, that is, the use of grammatical rules and other sources of knowledge to determine the functions of the words in the input sentence (a linear string of words) in order to create a more complicated data structure, for example, a derivation tree [10]. This structure depicts some of the relations between words in the sentence ("this adjective modifies that noun, which is the object of a prepositional phrase .....") and can be used to get at the meaning of the sentence. The structural description of the input sentence that results from parsing is called the "parse". For example, the parse of the English sentence "THE CAT DRANK THE MILK" is shown in the tree of figure (3.1).

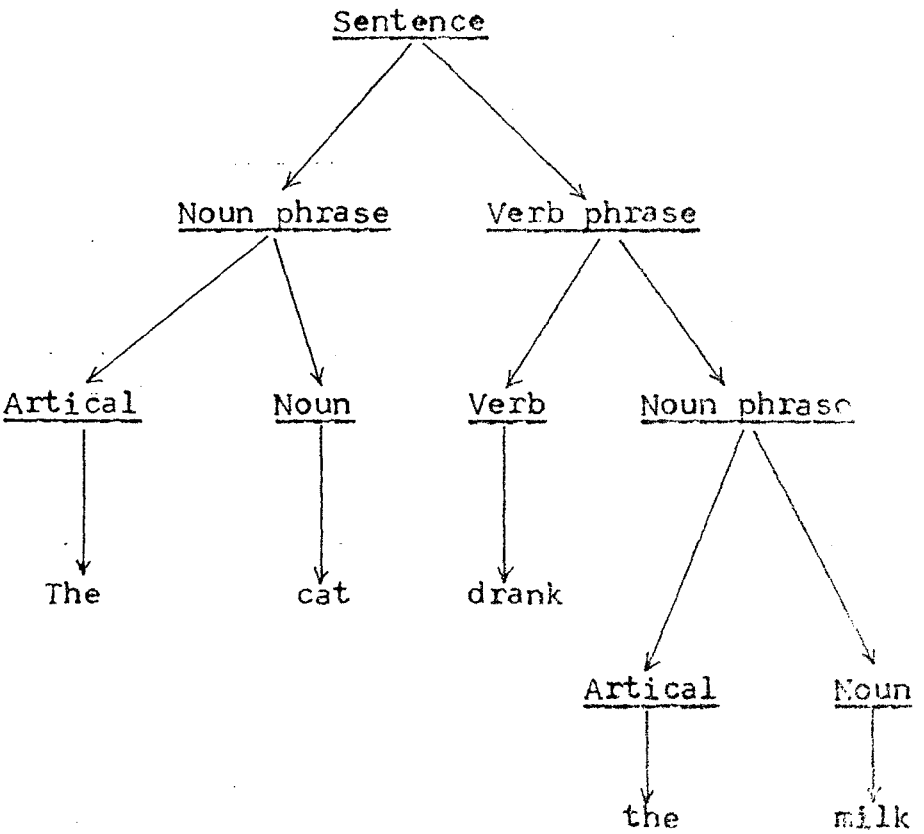


Figure (3.1) taken from [36] : Parse of the English sentence "THE CAT DRANK THE MILK".

Syntax It is the description of the ways in which words must be ordered to make structurally acceptable sentences in a language [4]. Every natural language has a syntax. It (syntax) may be defined as a system of word ordering. The basic task for Syntactic analysis is to tell which sentences are grammatical and which

are not. A sentence may be syntactically correct but have no meaning.

Semantics : Semantics refers to the meanings of words and sentences of a given language. There are two methods of processing natural language sentences by computers. a) Complete syntactic parsing method: In this method the parsing depends on the Syntax (grammar) of the sentence. If the sentence is syntactically wrong (ungrammatical), then it cannot be processed by the computer. b) Semantic processing method: Here, depending on semantics, the input sentence is processed. If not successful, the computer calls syntactic analyzer.

3.2 SYSTEM'S BEHAVIOUR

A "well-behaved" system is one that makes a user feel that the system is friendly. A friendly system is one that provides help to a user, does not constrain a user in interaction, and one that is easy to use.

A user of medical consultation systems would not be inspired by a system that asks him to act in a strictly prescribed way. User, as human, would need his interaction with the system to resemble that of

human to human communication in problem-solving environment. Hence, the system must be flexible towards a user. Free dialogue in a natural language would be an ideal flexibility. A user should be able to ask questions whenever he/she needs clarification. This can be achieved through a question-answering facility. A two-sided question-answering facility will humanize the communication process. Both, user and system, may interrogate each other in the consultation session.

A friendly atmosphere is to be provided by the medical consultation system, such as, giving help to user on request or helping user whenever system perceives that the user is in difficulty. For example, system may help a user in phrasing answers or may provide for the user a list of possible responses that he might be required to give.

An example of such help facilities is provided by the MYCIN consultation system. It prints a summary of user options as shown below :

UNKNOWN - (May be abbreviated U or UNK) used to indicate that the physician does not know the answer to the question, usually because the data are unavailable.

?	-	Used to request a list of sample recognized responses.
??	-	Used to request a list of all recognized responses.
RULE	-	Used to request that MYCIN display the translation of the current decision rule. This feature provides a simple capability for explaining why the program is asking the question.
QA	-	Used to digress temporarily in order to use the Explanation System.
WHY	-	Used to request a detailed explanation of the question being asked. This feature is much more conversational than the RULE option above and permits investigation of the current state of the entire reasoning chain.
CHANGEXXX	-	Used to change the answer to a previous question. Whenever MYCIN asks a question it prints a number in front of the prompt. This CHANGE 4 means "Go Back and Let me reanswer question 4".
STOP	-	Halts the program without completing the consultation.

HELP - Prints this list

For example, if a user wants to know why the system is asking such a question, he should be able to simply put WHY as he does in a conversation with another person. The following is an example from MYCIN [6]:

26) Has Jeremiah Sample (PATIENT-1) had any gu surgery or manipulation of the urinary tract? (e.g. indwelling catheter, cystoscopy, etc.)

**WHY This is the user question to which the system displays the following

I AM TRYING TO FIND OUT ABOUT THE IDENTITY OF THE ORGANISM.

I ALREADY KNOW THAT :

THE PORTAL OF ENTRY OF THE ORGANISM IS UPPER - GU

THE MORPHOLOGY OF THE ORGANISM IS ROD

THE STAIN OF THE ORGANISM IS GRAMNEG

THE SITE OF THE CULTURE IS BLOOD

THEREFORE, IF

1) THE PATIENT HAS NOT HAD A GENITO-URINARY MANIPULATIVE PROCEDURE, AND

2) URINARY-TRACT-INFECTION IS NOT A PROBLEM FOR WHICH THE PATIENT HAS BEEN TREATED

THEN :

THERE IS SUGGESTIVE EVIDENCE (.6)

THAT THE IDENTITY OF THE ORGANISM IS E. COLI

(RULE 156)

**YES

After the explanation from system the user enters his answer, that happens to be YES, and the system continues asking questions.

3.3 TOLERANCE

In human communication people adjust themselves to each other in tolerating errors, such as, grammatical and spelling errors introduced by any of the two persons engaged in a two-sided conversation. In interacting with medical consultation systems a user commits similar errors, i.e. grammatical and spelling errors. For effective interaction and for user's acceptance of medical consultation systems, these systems must be tolerant to grammatical and spelling errors and must be able to engage themselves in a conversation with the user when they do not understand user's request or answer. Such facilities, if provided by the system, will increase user's confidence in the consultation system and will humanize the interaction process.

Systems do not have emotions and do not understand human psychology ! How can such systems, then, have tolerance? The answer to this is: We have to equip systems with such abilities like tolerance to grammatical and spelling errors. The first we discuss the problem of grammaticality : "It is a problem because, despite researchers best efforts no complete grammar of English exists"[32]. If no complete computer grammar of English exists, allowing unconstrained English sentences is not possible if we insist that a syntactic 'parse' be constructed in order to derive the semantic component (meaning). That is, first checking if the sentence is grammatically correct; if correct, then its meaning is derived, otherwise, the sentence is rejected.

The solution to the problem of grammaticality is to design the system in such a way that it is primarily driven by semantic considerations (semantic processing). In this method no complete syntactic 'parse' is required prior to semantic processing. Rather, syntactic processing is done only when a semantic analyzer requests it. This technique will result in a system that tolerates grammatical errors

and it does not forces the user to follow rigid grammar. Instead a user has only to form meaningful sentences.

The other inferior solution to the problem of grammaticality is to process only a subset of English. This will follow "Syntax first" strategy. It is inferior because it puts, on the user, the constraint of expressing himself in that limited subset of English.

The Second is the problem of misspellings. Spelling errors are introduced by the user while he is typing his input through the keyboard or due to his ignorance of correct spelling. These spelling errors, if not detected automatically by medical consultation systems, cause confusion to the user. For example, consider the following interaction between MYCIN and physician user. MYCIN'S requests are numbered. User's responses are in upper case and follow double asterisks [12]. Here, the system guides the consultation.

——————PATIENT - 538——————

1) Patient's name :

** PT 538

2) Age :

** 34 YEARS

3) Sex :

** MAEL

= MALE

In the third response to MYCIN'S request, the user (physician) enters "MAEL" which is wrongly spelled. Its correct spelling is "MALE". For a moment, let us think of MYCIN as not having spelling correction facility. The MYCIN System would have simply rejected the physician's answer because it does not understand "MAEL", since it (MAEL) is not in the system's dictionary. But our physician is very sure of his answer ! This will confuse the physician. He may think of changing his response or he may feel that MYCIN is not friendly (the user did not notice his misteke even after rejection).

Fortunately, MYCIN has an ability to correct misspellings. MYCIN calls its spelling correction routines which inspect if the word is a misspelling of a one in its dictionary. MYCIN finds out the misspelling (transposition of adjacent letters "L" and "E") and prints " =MALE" to bring the attention of its user to the error (for verification). MYCIN continues (as user agrees to the correction) the consultation as if there was no misspelling.

It is ridiculous, for example, that a system does not understand question like : HOW DID YOU DECIDE THAT ORGANISM-2 WAS THE SAME AS ORGANSM-3, simply because ORGANISM-3 was misspelled ("I" is missing). More examples of misspellings corrected by MYCIN are shown in Appendix (A).

Types of misspellings and an automatic spelling correction program, as an implementation, are given in the next chapter.

3.4 SYSTEM'S RESPONSE

Another factor which is a problem in interface is the response time of the medical consultation system. Response time is the time taken by the system to generate a response for the user's request. It is a problem because our user is an on-line user who is waiting for a response. Response time must be very short to avoid boredom or impatience of an on-line user.

The output of the system should be given in its short form. This will help a user to quickly learn the shortened forms during an ongoing consultation session. For example, in reply to question the system must provide explicit answers [25]. It should not simply

dump out a file of information which some where contains the answer.

3.5 AN EXAMPLE OF MAN-COMPUTER COMMUNICATION PROBLEMS

As a summery to some of the problems of interface discussed in this chapter, we choose to give an example of a dialogue with RdMail system. RdMail is an interactive electronic mail system in daily use by several hundred researchers in the computer science department at Carnegie-Mellon University. RdMail has a conventional typescript-oriented command language, where most commands consist of a verb followed by a sequence of arguments. This example is intended to "demonstrate communication problems characteristics of all current interactive interfaces"[35]. Problems of system tolerance to grammatical and spelling errors, system behaviour towards user, and language adaptation are very clearly noticed in the example. The example was given by Hayes, et.al[35]. In this transcript, the user input follows the " ← " sign and the system's response follows the "?" sign. The annotations are added in square brackets.

[The user in this example is talking to the RdMail program. He wants to find out what messages he has received from someone called Robertson

since May 15. To do this the user selects the haeders command which prints out a line of summary information for each of a sequence (i.e. set) of messages it takes as argument. The user also knows that it is possible to specify subsets of all the messages in his message file by various selection keywords, including "from" which selects messages from a given source, and "since" which selects messages dated after a given date. With this in mind, the user gives the following command.]

← haeders from robertson since May 15

to which the system replies

? No such command as haeders. Type ? for help

[Unfortunately, the user has misspelt the name of the command. However, user sees his mistake immediately, decides he can proceed without the proffered help, and so retypes "headers" correctly, along with the remainder of the command line. In fact, the error could have been corrected by invoking a line editor built into RdMail; unfortunately, the user can only learn about this facility by reading the entire RdMail user's manual.]

← headers from robertson since May 15.

? Illegal Message sequence at "from" - missing quoted string

FROM ROBERTSON SINCE MAY 15
↑

[Now the user finds out about the second mistake he made. He failed

to put quotation signs around the string argument to "from", as RdMail grammar wants him to. The error message here is quite good, and in addition, RdMail shows him where it went wrong in Parsing the command. From this, it is quite easy for the user to guess that he should have enclosed the word, robertson, in quotes. What about the date? Should that be in quotes too? The user decides he had better put them in.]

← haeders from "robertson" since "May 15"

? No such command as haeders Type ? for help.

Darn it ! That same type again. No way but to type everything over again.

← headers from "robertson" since "May 15"

? Illegal Message sequence at "SINCE" - junk at end

FROM "ROBERTSON" SINCE "MAY 15"
↑

[What went wrong this time? The quotes around robertson seem to be acceptable to the system. This time it said the error had to do with "since", but "junk at end" is not very informative. Perhaps he should not have quoted the date? However, at this stage the user is so intimidated that he decides he really does need help, so he asks for it in the way some earlier error messages advised, i.e. type? for help.]

← ?

Type HELP for complete online assistance

Type DOCUMENTATION to enter the editor reading

the entire, cross-referenced RdMail documentation

[In addition to the above lines, the RdMail displays the following : all system commands, system attributes, and the message sequence keywords. The user finds that 'help' was not very helpful to the problem at hand. The user decides he needs the "complete" online assistance, he gives "help".]

← help

Type HELP <topic> for help on <topic> .

Type HELP TOPIC for assistance in using the TOPIC command to search the Help index.

? gives a list of commands

[Since he went wrong in using the "Since" keyword, the user assumes he needs to know more about it. He also implicitly assumes that "since" is a <topic> , and this turns out to be correct.]

← help since

[The user accepts all the help available about "since",by taking the default "yes" reply to each question, but he still does not feel very helped. The only information helpful to his program is that the date following "since" must indeed be quoted. At this point the user has had enough of on-line help. Poor user ! He goes off to ask someone better rehearsed in the folklore of the system. It turns out that this problem has a simple answer. In order to specify

two constraints on a RdMail message sequence, you must tell the system to take the intersection of the two sets generated by the constraints individually.]

← headers from "robertson" intersection since "May 15"

? Illegal message sequence at "ROBERTSON" - unknown symbol

FROM "ROBERTSON" INTERSECTION SINCE "MAY 15"

[It still does not work. Back to the Oracle ! Oh, it should be "intersect" not "intersection".]so finally ——

← headers from "robertson" intersect since "May 15"

52 16 May 80 George Robertson

67 21 Jul 80 Kamila Robertson

73 25 Jul 80 Kamila Robertson

[At last the user has the information he wanted, and that was so readily accessible with just a single RdMail command. But why should the user have had such difficulty in communicating that command to the system?]

CHAPTER - 4

SPELLING CORRECTION PROGRAM

In primary and secondary schools, students are given dictation classes, where the teacher reads a part of a text (or a list of words) and students are asked to write down. Student performance is then evaluated on the basis of the number of spelling errors he or she makes. Students are taught how to spell words correctly in order to minimize the spelling errors in their communication with the world, through writing, and also to add more power to their common sense to be able to recognize a misspelled message given to them. By this process people develop the ability to correct and understand messages that contain spelling errors upto some level.

People receive telegram messages with misspelled words that make the telegrams, sometimes, ununderstandable. Errors in telegrams may be due to typographical mistakes in the source office and receiving office or due to transmission errors by both transmitting and receiving mechanisms. Hence, the spelling errors may be classified into three types depending on the source that introduces the error :

1. Persons ignorance of correct spelling : These errors result from the difference between how a word sounds and the way it is spelled. For example, one may write TYPOGRAPHICAL as TYPOGRAFICAL.
2. Typographical errors : These errors are committed during typing due to wrong movements of fingers, resulting in pressing wrong keys or missing keys.
3. Transmission errors : This type of error occurs due to specific encoding and transmission mechanisms within the system.

Computers, like students, have to be taught how to detect and correct spelling errors in a given message to them (an input to the computer). No one has ever heard of a school that teaches computers the art of spelling correction through dictation classes ! For computers, there are computer specialists who design programs that enable computers detect and correct spelling errors in a given input data.

Research work on automatic detection and correction of spelling errors by computers was started as early as 1957, but the first application program was "SPELL" written by Ralph Govin at Stanford in 1971.

SPELL was used in the DEC-10 machine for checking arbitrary text files. The following paragraphs will narate the history of research in spelling correction programs as given by Peterson [7].

Early work on spelling correction programs was carried out by Davidson (1962); he was concerned with the retrieval of misspelled names in airlines passenger record system. Similarly, Carlson (1966) was concerned with names and places in a geneological data base. McElwain and Evans (1962) worked for improving the output of a system to recognize Morse Code.

Theoritical work was done on the problem of string matching and correcting algorithms. For example, Alberga (1967), states the problem of string similarity and misspelling as follows; "Given a list of words, perhaps with information, instructions or some such data attached, and a string of characters which does not occur in that list, can one determine with some fair degree of certainty whether that string is a misspelling of one of the words in the list. Restriction of the determination to a fair degree of certainty is necessitated because we are here attempting to divine the intent of the person who originated the

message"[9]. On the same problem (string matching) work was done by Damerau 1964; Riseman and Henson 1974; Lowrance and Wagner 1975.

SPELL, a spelling checker program was developed in 1971 and used by DEC-10 and DEC-20 machines. SPELL was used for correcting spelling in text files. Since 1971 when SPELL was developed as an experimental program, work on spelling correction programs was always in the context of correcting misspellings in a general manuscript, and input data to system programs. A recent approach by Durham.et.al 19 takes up the problem of spelling correction in user interface. They developed spelling correction program for the RdMail System.

This chapter will discuss the development of an interactive spelling correction program in a user interface application. Different factors are to be considered in the design of a spelling correction program to be incorporated in the user interface with interactive systems like - medical consultation systems. Human factors, storage, types of spelling mistakes to be corrected and other factors will be discussed in the coming subsections as we proceed.

4.1 SPELLING CORRECTION AND USER INTERFACING WITH MEDICAL CONSULTATION SYSTEMS

In a typical consultation session(Appendix-A), while using the medical consultation system, two types of interaction processes take place between the system and the user; a) dialogue guided by the system; the system asks questions to the user and answers are entered by the user in response, and b) dialogue guided by the user, the user asks questions where the user's questions are entered through a terminal and the system generates answers in response.

During the interaction process between the user and the system, spelling errors are introduced by the user while he is entering his answers and questions. A question or an answer may be a single word or a sequence of words (sentence). Spelling errors in the entered word(s) are introduced by the user due to ; ignorance of correct spelling or due to typographical errors. These types of errors will be discussed in section (4.4) as we proceed.

It is very important to incorporate an interactive spelling corrector in the user interface with medical consultation systems. This will help in

humanizing the interaction process between the user and the consultation system (In human written communication people use common sense in understanding misspelled words in a given message of communication). Hence, it would be an unfriendly act by the system to reject an answer or a question of user simply because it contains one or two misspelled words. For example, consider the following question by MYCIN consultation system and a misspelled answer of physician.

MYCIN : Enter the identity (gems) of ORGANISM-3
Physician : KLEBSIELA

In this example, "KLEBSIELA" is wrongly spelled by physician (user). Its correct spelling in the system's dictionary is "KLEBSIELLA" (one letter, i.e. "L", was missing). The systems rejection by the system of this answer will cause much of confusion and diversion of attention of physician who is very sure of his correct answer (misspelling still not observed by physician). Physiciens are not inspired by such an unfriendly partner in the dialogue.

Instead, MYCIN calls its spelling correction routines, corrects the misspelled input, and continues asking questions, as if no spelling error was there.

(MYCIN displays the correct spelling, i.e. = KLEBSIELLA). Hence, spelling correction ficility in user interface will enhance the acceptability of consultation systems by physicians, in making the interaction (consultation) more like a human-to-human communication process.

A misspelling of a word may, sometimes, match more than one word in the dictionary of the system. In this situation, the interactive spelling corrector takes note of all the words that match and asks the user as to which word of those matchings does he mean. This is another advantage of interactive spelling correctors in preventing wrong diagnosis due to misspelling.

For example, we consider the two words "XXYXY" and "XXYYX" which are two symptoms of two different diseases. The two words are stored in our dictionary (correct words). Let the symptom of patient be "XXYYX". Let the systems question be : what are the symptoms of the patient? The physician reply will be "XXYYX". But the physician wrongly enters "XXYXY" (by transposing the last two letters). If this error is not corrected the system would assume the symptom is XXYXY, as entered, and this will result in a wrong diagnosis.

To avoid this disaster, our interactive spelling corrector searches for all possible matchings and asks the user as to which word out of many (if any) does the user mean. This is another advantage of the interactive spelling corrector in making the consultation system more reliable.

4.2 THE DICTIONARY

The most important factors in the design of the dictionary are; 1) the dictionary size and 2) the search time, i.e. time taken by the system to find a particular word in the dictionary. The smaller the dictionary size, the cheaper the storage cost and the shorter the search time. A small dictionary may be stored in the main memory. In medical consultation systems, the vocabulary that constitutes the dictionary is very small, this is due to the fact that medical consultation systems are specialized in a very narrow domain of medical knowledge. For example, the dictionary of MYCIN system is about 800 words.

One method of reducing the size of the dictionary is to remove affixes. Only roots of the words are stored.

Prefixes :

Prefixes of the words are detached and the roots are looked up in the dictionary. Pre-, re-, anti-, trans-, dis-, un-, etc. are examples of prefixes. Words like, "recompute" and "reapply" are removed from the dictionary. For words like "remainder", "retain", "requirements", etc. since their meaning is not readily deducible when the prefix "re-" is suppressed, first we should look up the word in the dictionary. If not found, delete the prefix and search for the word. By this method (prefix removal) the word "redecorate", for example, is removed from the dictionary. There is no point in storing redecorate and decorate both.

Suffixes :

There are many endings, suffixes, that can be added to a verb stem to make a new word. Examples of suffixes are : -ing, -er, -ion, -ions, -ional, -ionally, -able, -ably, -es, -s, etc. For example, the plural is obtained simply by adding "-S" to many words. The storing of a noun in singular and plural forms is a waste of storage such as "DRUG" and "DRUGS". The suffix of any given word is removed and the word is

looked up in the dictionary. "If prefixes and suffixes are correctly handled, our dictionary may even be able to deduce the meaning of "antidisestablishmentarianism" given only the very stem "establish" [13].

In an interactive user interface, the response time is of great concern. The user is waiting for response. Thus, the interactive spelling corrector must be sufficiently fast to avoid frustrating the user. Hence, in the design of our dictionary, we concentrate on the search time factor.

For efficient dictionary storage and look-up, many methods were developed that take advantage of properties found in English text. For instance, the ten most common words (the, of, and, to, a, in, that, is, was, he) account for 24% of all the words. 128 of the most frequent words account for nearly 50% of all words [34]. Taking these properties of English text into consideration, one may design a dictionary of two levels for fast searching [7]. A two-level dictionary consists of a small (128 words) dictionary containing the most common (frequently referenced) words. This small dictionary is searched, for a given word, before

the larger dictionary that contains the rest of the words. One may, further, go for a 3-level dictionary, in which the 3rd dictionary is a subject-specific dictionary. This multi-level dictionary is used in full text processing where the dictionary size goes to more than 20,000 words.

In the design of the dictionary, for our program, we assume that all words are equally likely to be referenced. Thus a one level dictionary is chosen. How to store the one level dictionary such that it provides fast searching, is our next step.

There are two methods of storing a dictionary. Firstly, a dictionary may be stored as a tree structure and secondly, it may be stored in a form of tables.

The tree structures are convenient for storing sets of lexicographically ordered objects for purposes of alphabetically oriented information retrieval [36]. An example will illustrate how tree structures are used for words storage. Consider the five words "dog", "cat", "lion","fox", "tiger" to represent a dictionary. Figure (4.1) shows the tree structure.

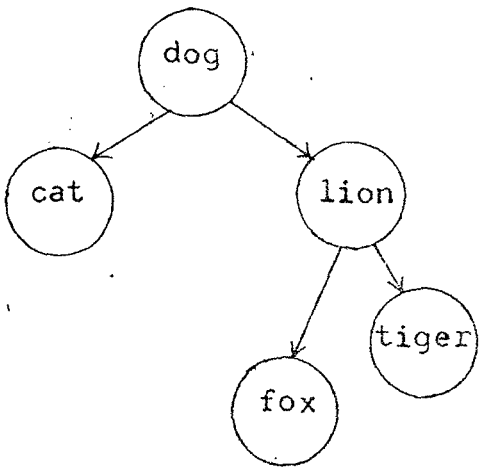


Fig. (4.1) taken from [36]. Tree used for information retrieval.

To search a word in the tree of fig. (4.1), the given word is compared with the root node first, If not matched the left successor of the root node is considered next for comparison if the given word occurs earlier in the dictionary, otherwise, the right successor of the root node is considered for comparison, if the word occurs later in the dictionary. For example, to determine if the word "fox" is in the tree, "fox" is compared with the word "dog" and since it occurs later in the alphabet, the right branch is taken. Then "fox" is compared with "lion" and since it occurs earlier in the alphabet, the left branch is taken. Then "fox" is compared with the left successor of "lion" which results in a match.

From the example above, we observe that for a dictionary of five words, fig. (4.1), three comparison operations were required. For a dictionary (tree) of more than thousand words, how many comparison operations are required? It is clear, from the above example, that the tree structure for a large dictionary is very time consuming, while searching for a given word. Hence, we must look for an efficient method of structuring the dictionary that allows very fast searching.

Hash table method for dictionary structure provides fast searching. A hash table consists of a number of buckets. Each bucket consists of one or more equal blocks of storage, and each bucket has a unique key. The hash table entry is the head of a chain of entries (words) each with the same hash value. The chain is searched for the input word. Hashing function is used to compute an address (head of a chain of entries) into the hash table.

4.3 HASHING

We have chosen the structure of the dictionary to be of tabular form. A table is a collection of data records (words). The words have no relationship other

than they are in the same table. A table is a small file, and a large group of files constitute the dictionary. Every data record (word) in the table has a key. A key is a particular field or combination of fields in a data record upon which some "Look up" or ordering process is performed [36]. A rationale for our choice of hash files for the storage of the dictionary, may be provided by giving a brief touch to other techniques of files of storage and their search time.

Sequential Storage Files

Records of a sequential file are physically stored in ascending or descending order. The records can be retrieved only in sequential order. Thus if the file had 20,000 records, retrieving the 10,000th record would require the leading of 10,000 records. This method is far too slow to be used for a dictionary search where search time is most important factor.

Index Sequential Storage Files

If the file is in key sequence, the usual method of addressing it is by means of a table called index. An index may be defined as a table which operates with a procedure that accepts information

about certain attribute values as input and gives information as output which assists in quickly locating the record or records that have those attribute values [23]. The input to the index table is the key of the record (word) sought and the result of the table look-up operation is the relative address or actual address of the record on the file unit. The disadvantages of index-sequential files are : (a) the time required to access and search the index table for a given key, and (b) the memory size required for index table storage. It is much faster than the sequential search, and inferior to hashing files.

Binary Searching of Index Table :

Binary searching requires that a table be ordered on the table keys. The principal feature of this provides that each "look" into the table either finds the key in question or eliminates half of the table positions from further consideration. The key of the given record is compared with the key of the midpoint of the area to be searched. If the given record-key is greater, the lower half of the area to be searched is ignored. The next look is taken at the

midposition of the remaining upper half. This process continues until a key match is found or the area "shrinks" to nothing. This searching method is time consuming because it uses the trial-and-error method in locating the key.

"Hashing is an ingenious and useful form of address calculation"[23]. Hashing or key transformation is a technique in which direct table look-up can be made rather than trial-and-error search. It is the fastest method of table search. In hashing technique, some routine operation(hashing function) is done on the original key to transform it into a new key. The transformed keys are called the hash addresses. These hash addresses are then used as direct-entry to table positions.

Ideally, the hashing scheme would convert the original keys to transformed keys with no duplicates. The situation where two keys have the same hash value (duplicate keys) is called "hash clash" or "collision". Collision cannot be eliminated completely, but a hashing function may be chosen in such a way as to minimize the number of collisions.

For a given number of entries, if the hashed addresses could be spread over a large table space, the number of collisions would be reduced, but the number of empty table buckets would increase. A good hashing technique should balance the opposing strains of empty (wasted) table buckets and decrease in the number of collisions. A bucket is an address space which can hold one or more records (words). The size of the bucket is chosen by the programmer.

The collisions can be handled by an overflow mechanism. In this mechanism, if the transformed key hashes to a storage space (bucket) that is already full, the record is stored in an overflow bucket. The address of the overflow bucket is recorded in the home bucket (the bucket to which the transformed key hashed first). If the first overflow bucket is full and another transformed key is assigned to the same full home bucket, then the record is stored in a second overflow bucket. The hash address of the second overflow bucket.is recorded in the first overflow bucket and so on (More than one overflow bucket is seldom needed). Thus, the overflow area (overflow buckets) will be searched sequentially. This will

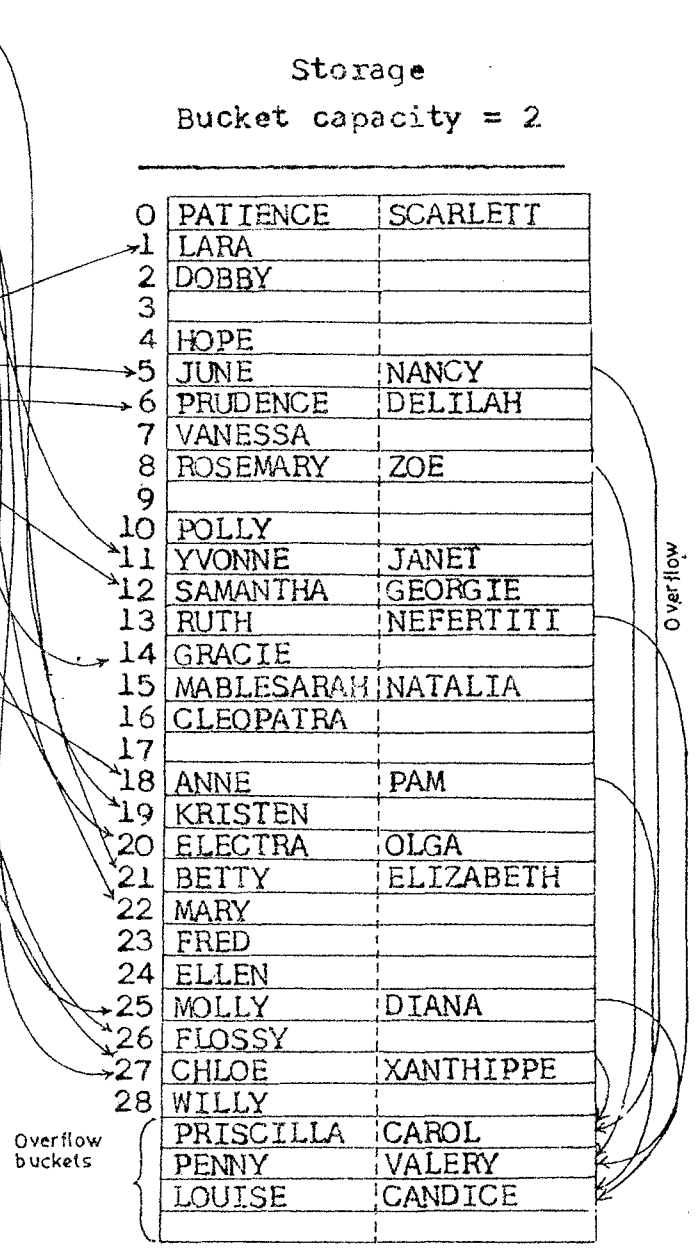
result in longer search time for locating a record that happens to be in the overflow area. Thus, the bucket size is to be carefully decided such that the number of overflow buckets required is minimum. In practice, the bucket capacity is often tailored to the hardware characteristics. The bucket capacity is the number of records (words) that can be accommodated in one bucket. Figure (4.2), is an illustration of hashing to a storage of 29 buckets, each of capacity 2.

For a given word, how to determine whether the word is in the hash table (dictionary) or not? Every word may be represented by a numeric key. The word's key is converted into a near-random number (transformed key). This near-random number refers to the address where the word is stored. Words having the same near-random numbers are stored in one bucket. The bucket is referred to with this near-random number. If the number of words having the same near-random number is greater than the bucket capacity, then an overflow mechanism is used. Thus, searching for a word in our dictionary is done by applying the hashing function to the word and producing the transformed key.

Figure (4.2) Simple hashing process with a bucket capacity of 2. From : MARTIN, J. [23]

A. Hashing to a storage with 29 prime buckets, each of capacity of 2.

	Key	Key converted to digits	Remainder after dividing by 29
1	BUTTY	25338	21
2	JUNE	1455	5
3	CHLOE	38366	27
4	KRISTEN	2992355	19
5	YVONNE	856555	11
6	MOLLY	46338	25
7	DIANA	49151	25
8	ELECTRA	5353391	20
9	OLGA	6371	20
10	GRACIE	791395	14
11	LARA	3191	1
12	NANCY	51538	5
13	PRUDENCE	79445535	6
14	SAMANTHA	21415381	12
15	ANNE	1555	18
16	FRED	6954	23
17	MARBLESARAH	41235021918	15
18	MARY	4198	22
19	FLOSSY	636228	26
20	JANET	11553	11
21	PAM	714	18
22	XANTHIPPE	715389775	27
23	PRISCILLA	799239331	27
24	CAROL	31963	5
25	ROSEMARY	96254198	8
26	RUTH	9438	13
27	ELIZABETH	539912538	21
28	NEFERTITI	556593939	13
29	ELLEN	53355	24
30	ZOE	965	8
31	PATIENCE	71395535	0
32	PENNY	75558	13
33	VANESSA	5155221	7
34	WILLY	69338	28
35	VALERY	513598	8
36	LOUISE	364925	18
37	SCARLETT	23193533	0
38	CLEOPATRA	335671391	16
39	GEORGIE	7569815	12
40	CANDI CE	3154935	25
41	NATALIA	5131391	15
42	POLLY	76338	10
43	HOPE	8675	4
44	DELILAH	4539318	6
45	GERT	7593	24
46	DOBBY	46228	2

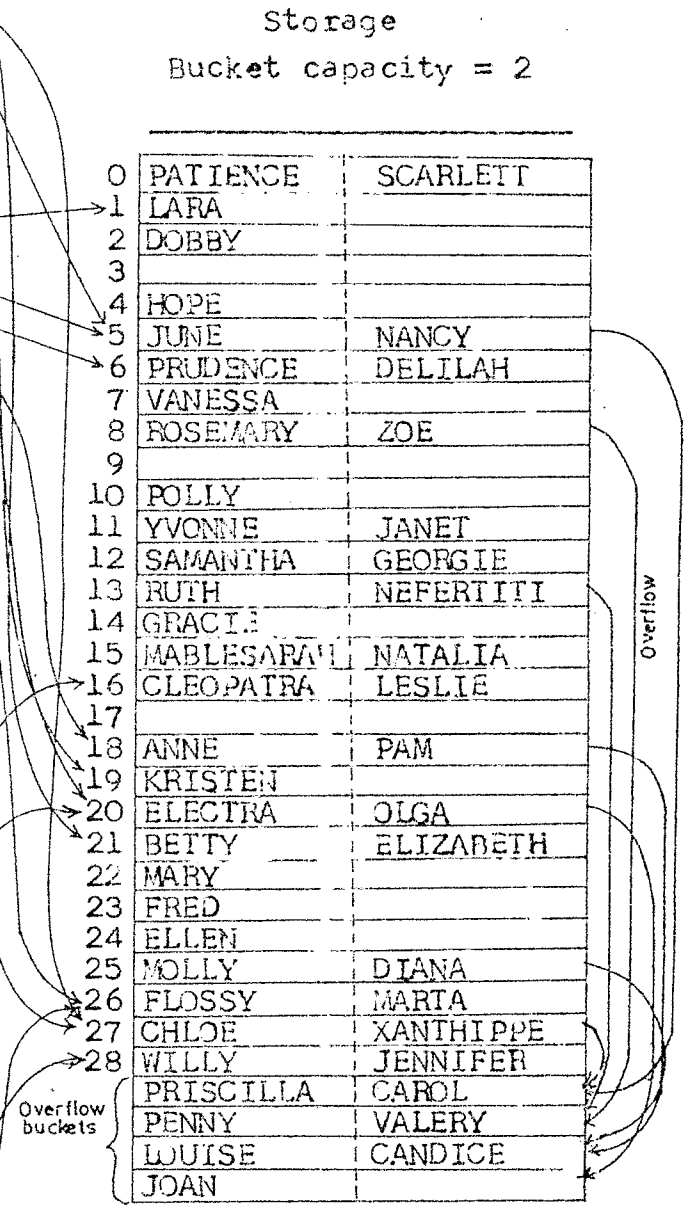


↑ Sequence of file loading

↑ The key is converted into digits by retaining only the four bits which represent numbers in BCD code. This method is used here only for illustration.

B. Four new records (the words - LESLIE, JOAN, JENNIFER, MARTA) are added to fig. 4.2(A).

	Key	Key converted to digits	Remainder after dividing by 29
1	BETTY	25338	21
2	JUNE	1455	5
3	CHLOE	38365	27
4	KRISTEN	2992355	19
5	VYONNE	856555	11
6	MOLLY	46338	25
7	DIANA	49151	25
8	ELECTRA	5353391	20
9	OLGA	6371	20
10	GRACIE	791395	14
11	LARA	3191	1
12	NANCY	51538	5
13	PRUDENCE	79445535	6
14	SAMANTHA	21415381	12
15	ANNE	1555	18
16	FRED	6954	23
17	MABLESARAH	41235021918	15
18	MARY	4198	22
19	FLOSSY	636228	26
20	JANET	11553	11
21	PAM	714	18
22	XANTHIPPE	715389775	27
23	PRISCILLA	799239331	27
24	CAROL	31963	5
25	ROSEMARY	96254198	8
26	RUTH	9438	13
27	ELIZABETH	539912538	21
28	NEFERTITI	556593939	13
29	ELLEN	53355	24
30	ZOE	965	8
31	PATIENCE	71395535	0
32	PENNY	75558	13
33	VANESSA	5155221	7
34	WILLY	69338	28
35	VALERY	513598	8
36	LOUISE	364925	18
37	SCARLETT	23193533	0
38	CLEOPATRA	335671391	16
39	GEORGIE	7569795	12
40	CANDICE	3154935	25
41	NATALIA	5131391	15
42	POLLY	76338	10
43	HOPE	8675	4
44	DELILAH	4539318	6
45	GERT	7593	24
46	DOBBY	46228	2
47	LESLIE	352395	16
48	JOAN	1615	20
49	JENNIFER	15559659	28
50	MARTA	41931	26



i.e. hash address, which will hash to the bucket address where the word is stored. The bucket is then, searched for the given word. If the word is stored in an overflow bucket, the overflow bucket is located and searched for the given word. Address of the overflow bucket is given in the home bucket. If the word is not found in the dictionary then it is either misspelled or it is not in the dictionary at all. The spelling correction routines (section (4.5)) may be called and the word may be checked for spelling correction.

KEY-TO-ADDRESS CONVERSION ALGORITHMS :

The key of any word is converted to an address where that word (or a group of words having the same address) is stored. The algorithm to convert a key to an address is of three steps [23]:

1. The key is converted into a numeric form ready for manipulation.
2. The keys are operated on by an algorithm which converts them into a spread of numbers of the order of magnitude of the address numbers required. The key set should be distributed as evenly as possible across this range of addresses.

3. The resulting numbers are multimplied by a constant which compresses them to the precise range of addresses. If the second step, for example, gives four digits when 7000 buckets are to be used, the four-digit number is multiplied by 0.7 to put it in the range 0000 to 6999. This relative bucket-number is then converted into a machine address of the bucket.

For the second step many transforms have been proposed and tested. It is desirable that the transform distribute the keys as evenly as possible between the available buckets. Below some of these good transforms. Martin [23] gives several transform methods.

1. Mid-square Method :

In this method, the key is multiplied by itself and the central digits of the square are taken and adjusted to fit the range of addresses. For words having 6 digit keys that are to be stored in 7000 buckets, the key may be squared to form a 12-digit field of which, digits 5 to 8 are used. For example, if the key is 172148, the square is 029634933904 and the central four digits are 3493. The four digits 3493 are multiplied by 0.7 :

3493 X 0.7 = 2445

The number 2445 is used as the bucket address.

2. Dividing :

The key is divided by a number approximately equal to the number of available addresses, and the remainder is taken as the relative bucket address. Consider the same key 172148 again and assume there are 7000 buckets. Here, 172148 may be divided by 6997. The remainder is 4220, and this is taken as the relative bucket address. This method distributes the keys much more evenly than mid-square method.

3. Radix conversion :

In this algorithm, the radix of a number may be converted, for example, to radix 11. The excess high-order digits may then be truncated. The key 172148 is converted to :

$$1 \times 11^5 + 7 \times 11^4 + 2 \times 11^3 + 1 \times 11^2 + 4 \times 11^1 + 8 \times 11^0 = 266373$$

and the digits 6373 are multiplied by 0.7 to give the relative bucket address 4461.

4.4 MISSPELLINGS :

In this section we will consider the possible misspellings that occur during an interaction process

between a user and the system. Methods of how to correct these misspellings are discussed too. In the next section, design considerations and flow-chart of routines of an implementation program are presented. The program corrects some types of spelling errors.

In communicating with an interactive system, like medical consultation system, the misspellings mostly occur because of user ignorance of correct spelling and due to typographical errors. Irrespective of the source that caused the error, all are called spelling errors.

Four classes of single spelling error were discovered by F.J. Dameran [40]. In the experiment, his program rejected all the words that had no matching in the systems master list of acceptable words. An inspection of those words rejected because of spelling errors showed that over 80 percent fell into one of the four classes of single error - one letter was wrong, or one letter was missing, or an extra letter had been inserted, or two adjacent characters had been transposed. These errors may occur due to hitting a key twice, or misreading, or ignorance of spelling, or hitting a wrong key.

For example, consider the word INFECTION to be as input to the system. The four possibilities of having a single error to occur in entering this word are :

INFACTION INFECTION	One letter wrong at "A"
INFECTON INFECTION	One letter missing, that is "I"
INFEECTION INFECTION	One letter extra, that is "E"
INFCETION INFECTION	Transposition of two adjacent letters, "C" and "E"

From table (4.1), one may notice that the single error mostly occurs after the second or third character in the misspelled word. This observation may help in constructing the correction routines.

Table (4.1)

The incorrect spelling is due to one of the four classes of single error - one letter missing, one extra, one wrong, transposition of two adjacent letters. Words are taken from [40].

Correct Spelling	Incorrect Spelling
ABSORBENT	ABSORBANT
ABSORPTION	ABSORBTION
ACCOMODATE	ACCOMDATE

Correct Spelling	Incorrect Spelling
COMMITTEE	COMMITEE
CONCEDE	CONSEDE
CONSENSUS	CONCENSUS
CYNICAL	SYNICAL
DISAPPOINT	DISAPOINT
ECSTASY	ECSTACY
FEBRUARY	FEBUARY
GRAMMAR	GRAMMER
HYGIENE	HYGEINE
LIQUEFY	LIQUIFY
WEIRD	WIERD

Misspellings based on how the words sound are very often. A user may wrongly write the sound "ph" as "F" or "qu" as "K" or "ou" as "O" or even "ough" as "O". Thus, the word "physics", for example, is misspelled as "fysics", and the words "thought" and "brought" are spelled as "thot" and "brot" respectively. One suggested solution to correct such spelling errors is that the words (that are suspected to be misspelled) are converted to a standard phonetic spelling which would be used to find similarly sounding words in the dictionary.

There are some common misspellings in English, such as, words,like "gratefully", which is always misspelled as "greatfully". If a proper study was done and these common misspellings are identified, the solution

would be to include the misspelled words in our dictionary along with an information indicating that this word is a misspelling of another correct word. Thus "greatfully" would be tagged as a misspelling of "gratefully".

Another spelling error source is the misuse of affixes. As was discussed in section (4.2), careful handling of affixes would reduce the dictionary size. One method is used to solve affixes handling. In this method, each word is examined for a list of common affixes. These affixes are removed if found in the word. The resulting root word is then looked up in the dictionary. If found, the word is considered correct. This method does not catch misspelled words which are the result of correct affixes incorrectly applied to correctly spelled words, such as "implyed". Here, the suffix "-ed" is incorrectly applied to "imply". The correct form is "implied". This allows a large number of misspelled words to escape detection.

The solution to this problem is to flag each word in the dictionary with its legal affixes. In this approach, the dictionary is first searched for the word. If not found, the word is examined for

possible affixes. These are removed and the dictionary searched for the root. If root is found, its flags are examined to see that the particular affix removed is legal for this root.

4.5 DESIGN CONSIDERATIONS AND PROGRAM FLOW CHARTS

Before discussing the design considerations of the interactive spelling corrector, it would be worthwhile pointing out the difference between correcting spelling in general manuscripts and correcting spelling in the user interface applications. In a general manuscript, the correction program reads the whole text which is to be corrected and the program searches the dictionary for every word that is read. Then the program puts in a list, all the words that are not found in the dictionary. For each given word that is not found in the dictionary, the program constructs a list of all words which could produce the given word by using its correction routines. These words in the list are called candidate spellings. The list of candidate spellings (of each word) is displayed to the user for the selection of the correct word. From this, we conclude that if a given a text has 5 misspellings, the user is asked 5 times to select a correct word each time.

On the other hand, in the user interface applications the spelling correction program is called only when the interactive system does not understand a particular word of the input sentence.

Our spelling correction program is an interactive spelling corrector that could be incorporated in the user interface to interactive systems like - medical consultation systems.

Although our program is for correcting spellings in the user interface, we borrowed a few ideas from Peterson [7], whose program was for correcting spelling in general manuscripts.

The design considerations for our interactive spelling correction program are of two categories : (a) considerations that effect the design of the algorithm, and (b) considerations that effect the user of the interactive system.

a) Algorithm Design Considerations

We assume that there is exactly one error in the given word which is to corrected and that the error is caused by one of the four classes of single error that account for over 80 percent of the spelling errors ;

(1) transposition of two adjacent letters

(2) one letter extra

(3) one letter wrong

(4) One letter missing

These four classes of single error that account for 80 percent of spelling errors were discovered by F.J. Dameran [40]. In the same test that was done by him, (Appendix-B), we notice that 91 percent of the four classes of single error were committed in the third or subsequent letters of the misspelled word, and only 9 percent of these errors occured in the first two letters of the word.

For the hashing function, we choose the first two letters of the word and the word length to generate the numeric key of a given word. This choice has two advantages: (1) Since the frequency of errors that occur in the first two letters in the given word is very low, this will decrease the possibility of having words that are wrong but have matching words in the dictionary and (2) It will help in detecting the errors of one letter missing and one letter wrong. This point will be clearer as we proceed.

How does the program work? For a given word, the program searches the dictionary and if the word is found in the dictionary it is assumed to be correctly spelled. Thus, as stated by L. Press [18]: "Don't expect too much from a spelling proofreader. It will

sea nothing wrong with this sentence, because the word sea is in its dictionary", this program cannot detect the error in the word "sea" which is a misspelling of "see" because both words are in the dictionary.

If the word is not found in the dictionary, it is suspected to be a misspelling of another word in the dictionary. Then the program calls its four correction routines - TRANSPOSE, EXTRA, MISSING, and WRONG, one at a time, and it inspects the given word for one of the four errors: the transposition of two adjacent letters, one letter extra, one letter missing and one letter wrong.

To correct the error that results from the transposition of two adjacent letters, the program transposes each pair of adjacent letters of the given word, one pair at a time, and it searches for the resulting word in the dictionary. As an example, let the given word be represented by the string "ABCDE", then the results of transposing each pair of adjacent letters are the following strings :

BACDE (transposition of "A" and "B")

ACBDE (transposition of "B" and "C")

ABDCE (transposition of "C" and "D")

ABCED (transposition of "D" and "E")

Thus for a word of length "L", the program searches the dictionary "L-1" times to detect all possible matchings. The matching words, if any, are put on a list called MATCHLIST.

Next, the word is inspected for the error of one letter extra. This is done by deleting each letter in the word, one at a time, and the program searches the dictionary for the resulting words. If we consider the string "ABCDE" as the given word, then the results of the deletion process are the following words :

BCDE ("A" is deleted)

ACDE ("B" is deleted)

ABDE ("C" is deleted)

ABCE ("D" is deleted)

ABCD ("E" is deleted)

If any of the above strings is found to be matching a word in the dictionary, the string is added to the MATCHLIST, and the program proceeds to inspect

the given word for the next possible errors, i.e. one letter wrong and one letter missing.

To correct one letter wrong in the given word, our program assumes that the error (wrong letter) is in the third or subsequent letters. Applying our hashing function to the given word will result in a hash table entry where all the expected matching words are stored. Each expected word is compared with the given word to determine the words that differ from the given word by exactly one letter. Our program tries varying that single miss-match letter of the given word through all the 26 alphabets till an exact match is reached. The dictionary word is put on the MATCHLIST.

For errors in the first and second letters, the program tries varying the second letter through all the 26 alphabets, each time searching the dictionary for a matching and adding the matching words, if any, to the MATCHLIST . Then all 26 possible values of the first letter are tried, after setting the second letter to its original value. After making the above 26 searchings in the dictionary and after adding the matched words (if any) to the MATCHLIST, the ~~problem~~ program proceeds to do the next test, i.e. one letter missing.

For the one letter missing in the given word of length "L", the program inserts a blank in every possible position in the given word. This will result in "L+1" words each with one letter wrong (the blank character). If the given word is represented by the string ABCDE, then the resulting words due to the insertion of the blank are :

ABCDE

ABCDE

ABCDE

ABCDE

ABCDE

ABCDE

To correct a word with one letter wrong (blank), the same procedure as that of one letter ~~missing~~ wrong is followed and the matching words (if any) are added to the MATCHLIST.

After inspecting the given word for all the four possible errors, if exactly one word is present in the MATCHLIST then the program guesses that word as the correct one and it displays the word for the user approval. If more than one word is in the MATCHLIST, the program displays the list and asks the user to pickup the correct word.

If no words are there in the MATCHLIST, then the given word is rejected and the user is asked to change his input, then the program terminates.

b) User Interaction Considerations :

Having a spelling corrector in the user interface to interactive systems gives rise to many considerations that are to be decided. These considerations concern the interactions with the user when spelling correction is attempted. These considerations are important because they affect the user's ability to work with the interface.

One of the considerations that effects the user's interaction with the system is the number of times the system interacts with the user when spelling correction of a given word is attempted. For example, if for a given misspelled word, each of the four tests for the single error (transposition, extra-letter, wrong-letter and missing-letter) results in one matching word or more, should the system then interact with the user four times to correct the given word? This option, if used, will cause the diversion of attention of the user whose main aim is the consultation (the medical advice) and also will cause the loss of computer time (in the

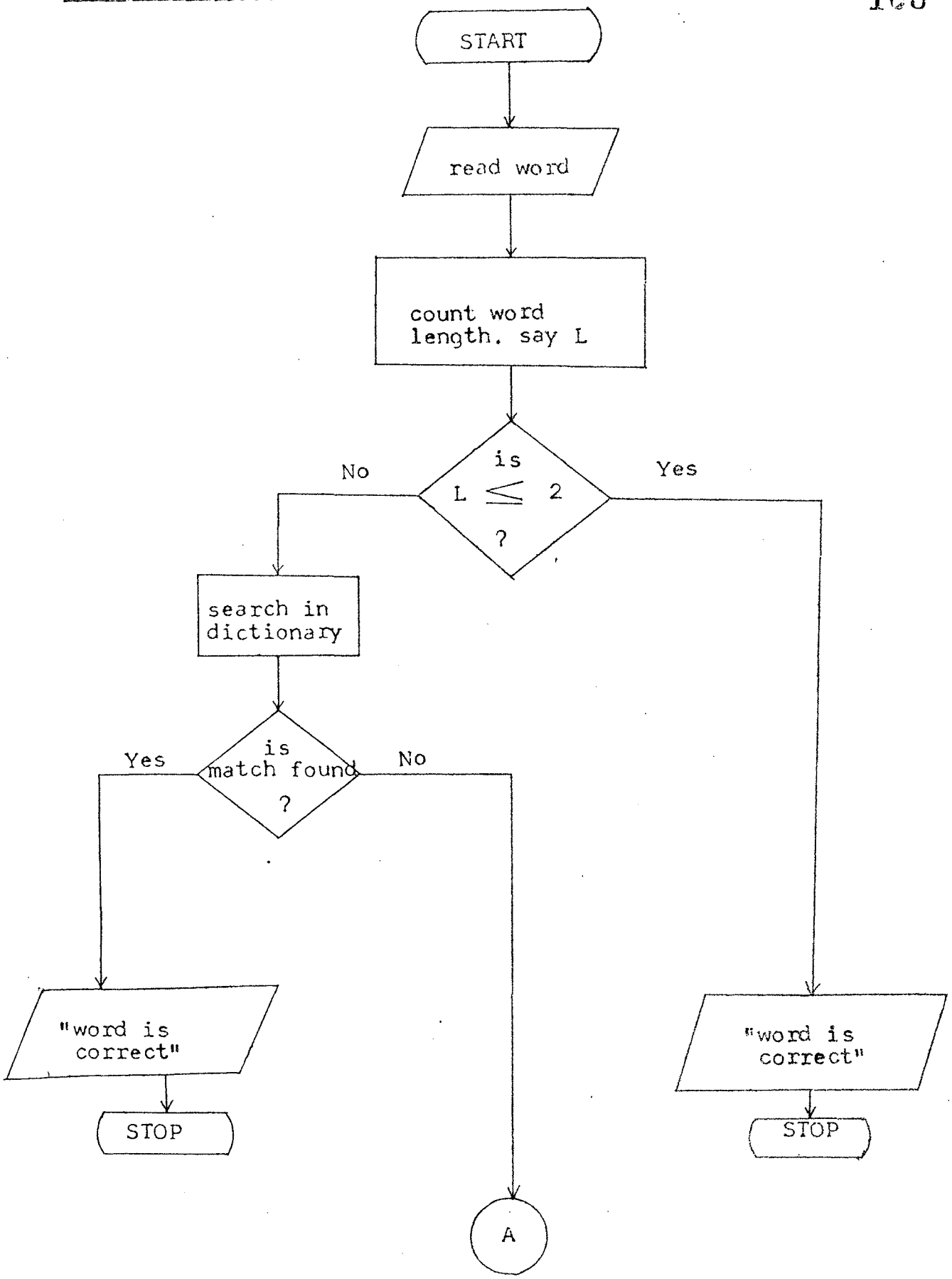
generation of messages). Hence, we chose that the spelling corrector should put on a list all the matching words that result from the four (transposition, extra-letter, wrong-letter, and missing-letter). The list of matching words (if any) is then displayed to the user for the choice of one correct word out of many.

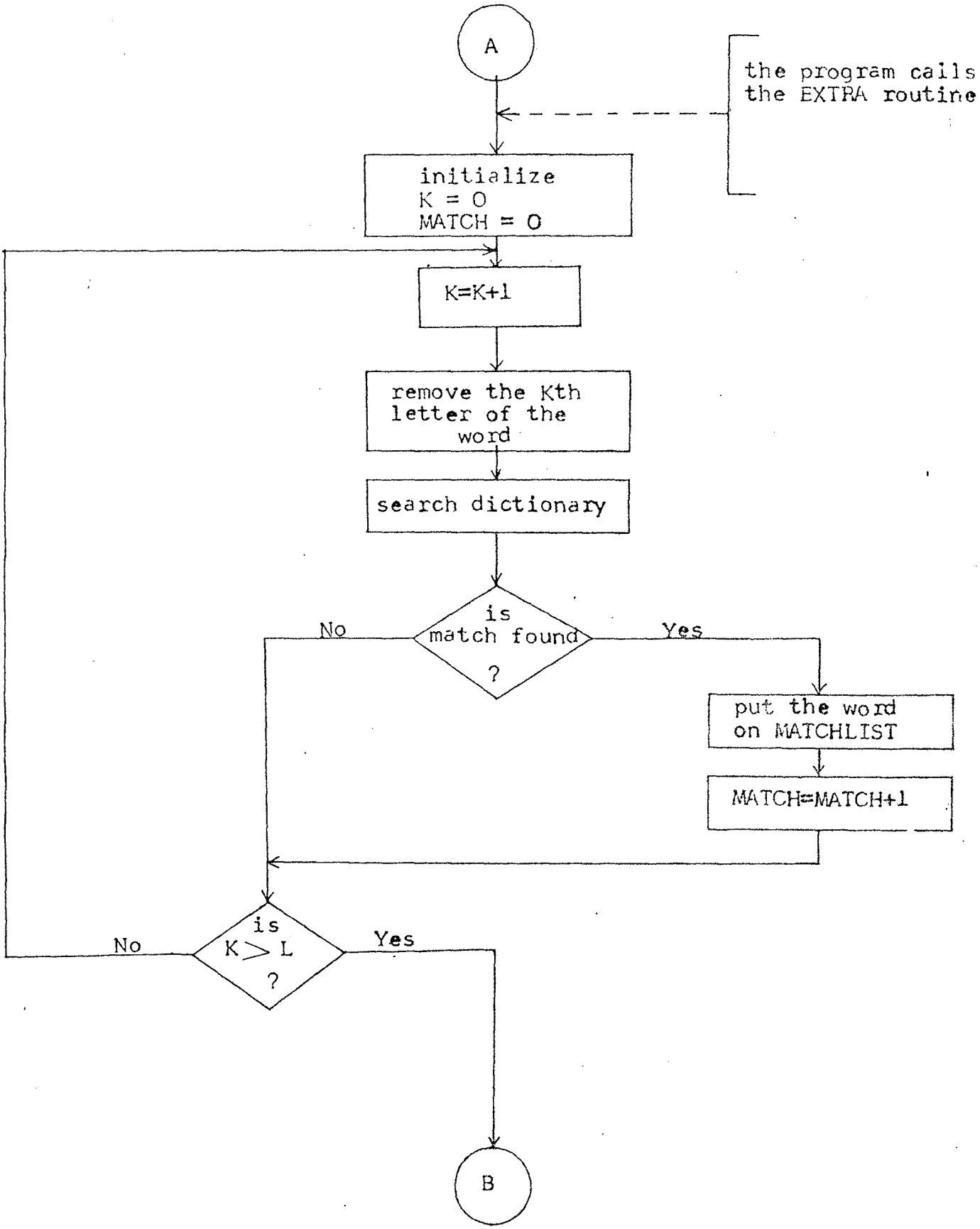
Another consideration is the length of the given word. If the given misspelled word contains only two letters, the various tests for correcting errors may still produce a substantial number of possible matches. Here, the "extra-letter" and wrong-letter tests will produce a large number of possible matches. As an example, let "XY" be the given misspelled word which is to be corrected. Applying the "extra-letter" test would match all the words beginning with either X or Y. While the "wrong-letter" test would match all words beginning with X? and ?Y, where ? matches any letter. Then the application of the two tests (extra and wrong-letter) to the two letter word would result in a large number of matching words. The displaying of such a long MATCHLIST would puzzle the user and would make it difficult for him to choose the correct word. On the other hand the "transposition" test and "missing-letter"

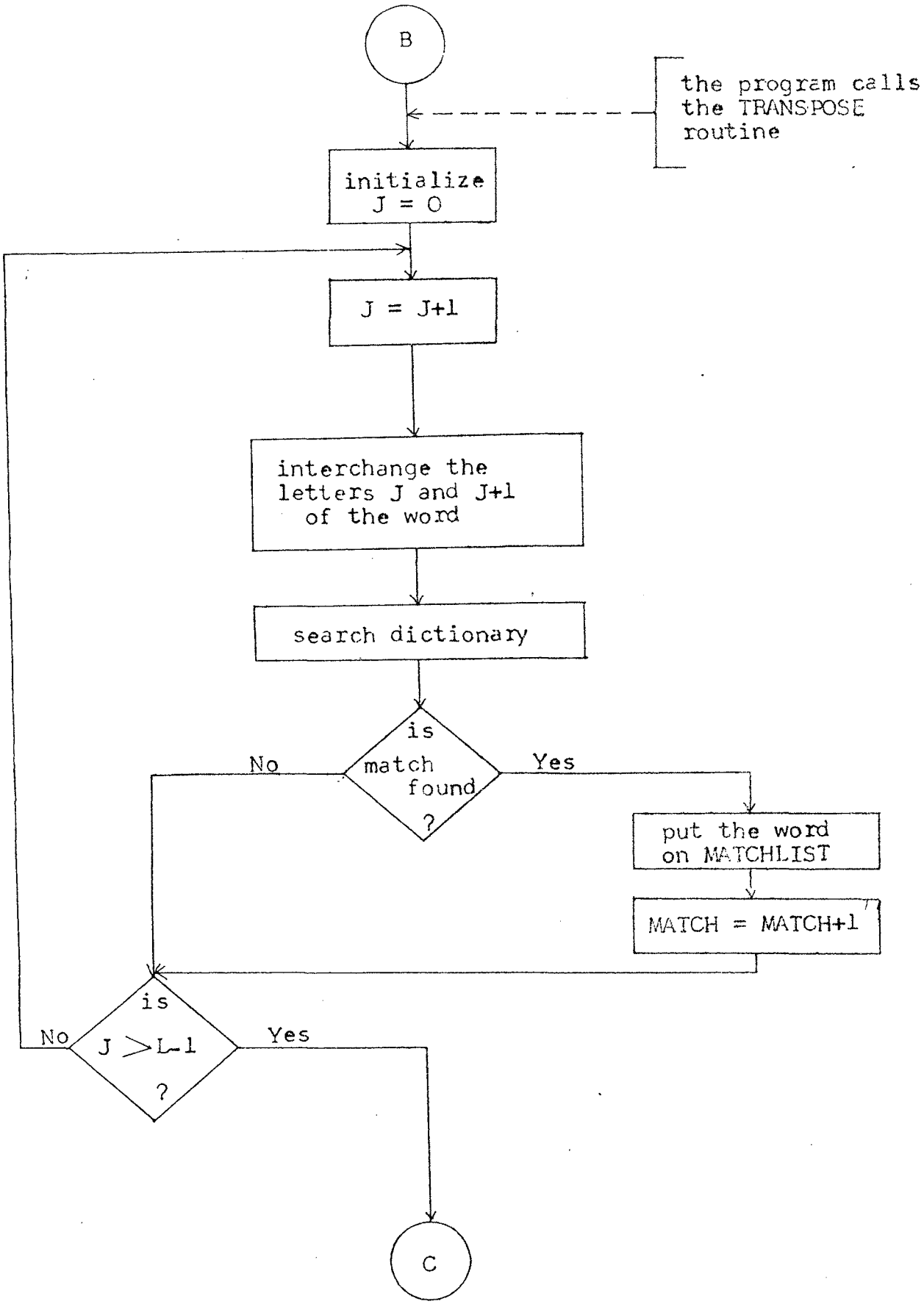
tests are reasonable and behave in the same way for both two-letter words and longer words. To avoid the above problem, our program assumes all words of length "2" are correctly spelled.

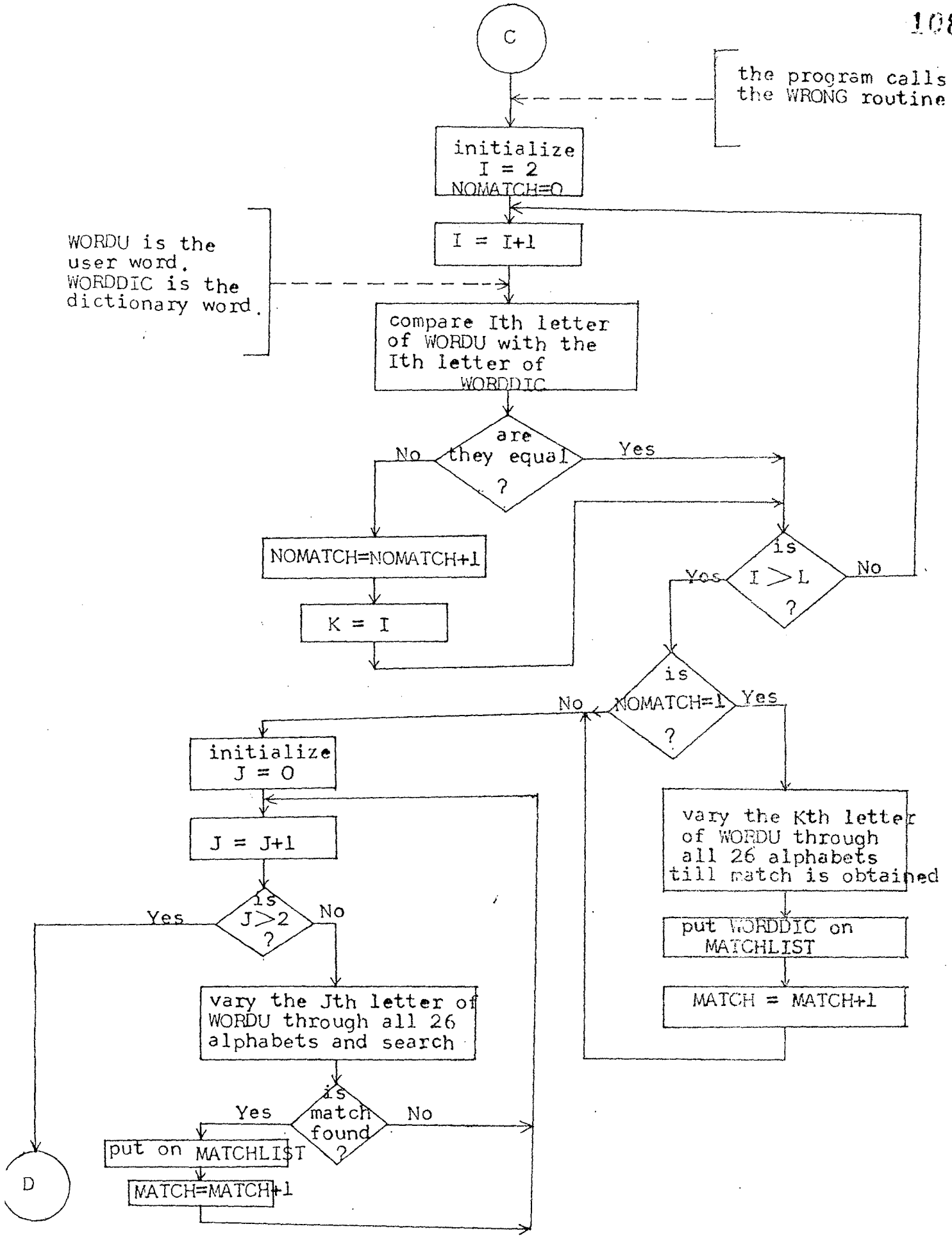
The other consideration concerns the words of length "1". What should we do with words that have one letter? Let us, for a moment, forget our proposed hashing function. If the given word is of length "1", the"extra-letter" test would omit this word of single letter from the dictionary search and would therefore match all the words in the dictionary (or none of them, depending on the semantics of an empty string). This does not help the user. On the other hand the "wrong-letter" test would match the given one letter with all the other letters in the alphabet. Thus, our program assumes all the words of length "1" are correctly spelled.

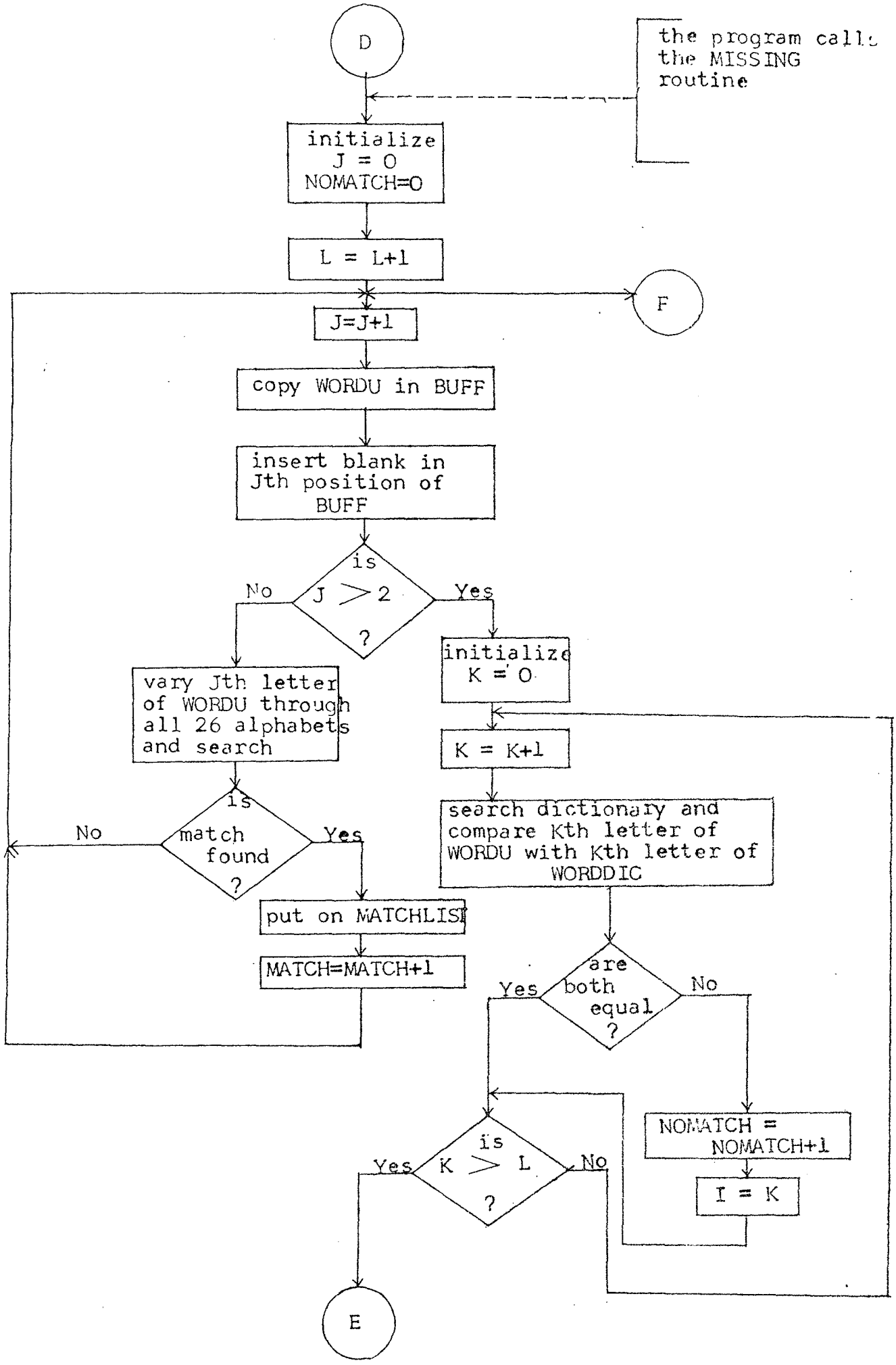
PROGRAM FLOWCHART :

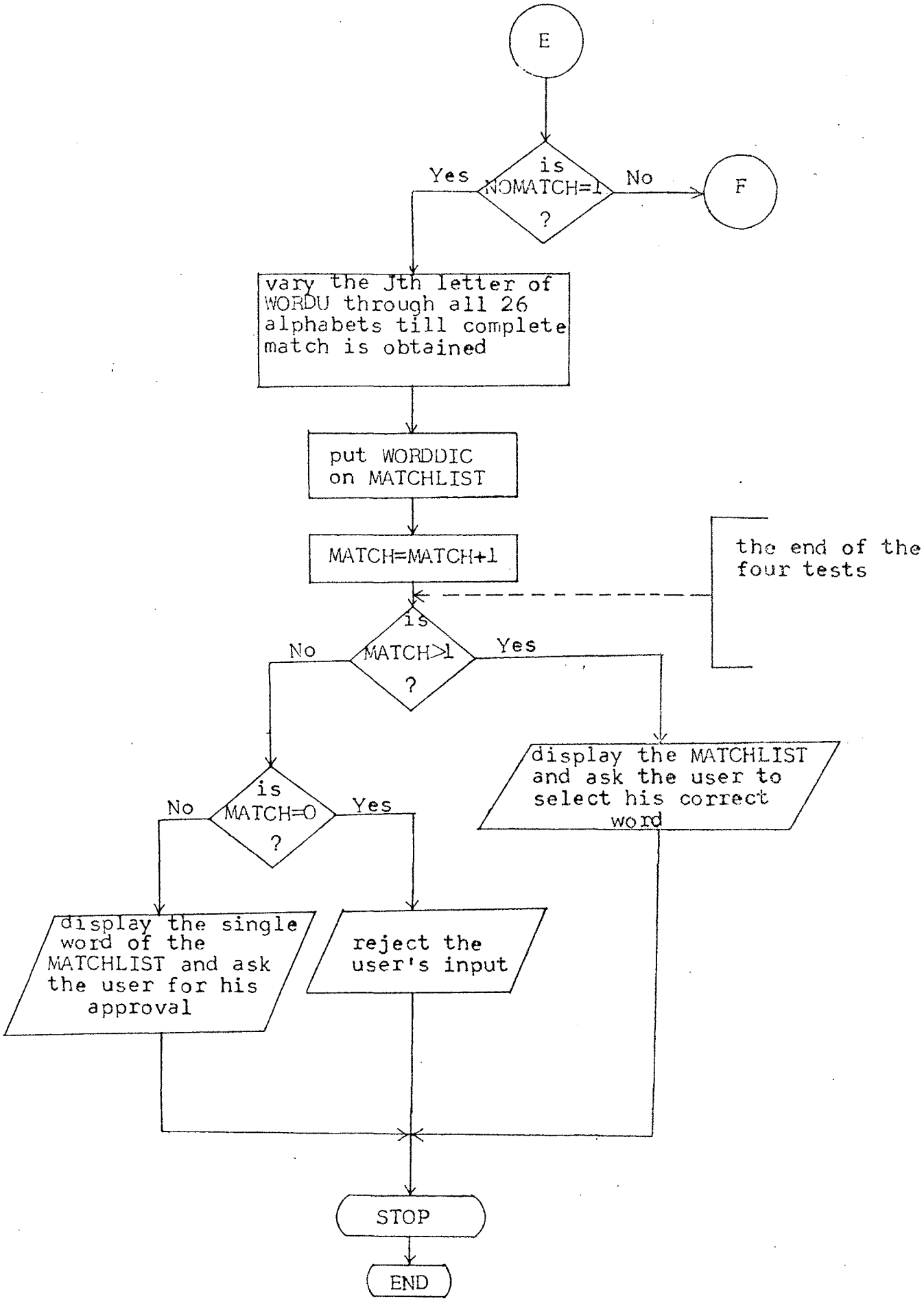



A
the program calls the EXTRA routine
initialize
K = 0
MATCH = 0
K=K+1
remove the Kth letter of the word
search dictionary
is match found ?
No
Yes
put the word on MATCHLIST
MATCH=MATCH+1
is K > L ?
No
Yes
B


B
the program calls the TRANSPOSE routine
initialize J = 0
J = J+1
interchange the letters J and J+1 of the word
search dictionary
is match found ?
No
Yes
put the word on MATCHLIST
MATCH = MATCH+1
is J > L-1 ?
No
Yes
C


C
the program calls the WRONG routine
initialize I = 2 NOMATCH=0
I = I+1
WORDU is the user word. WORDDIC is the dictionary word.
compare Ith letter of WORDU with the Ith letter of WORDDIC
are they equal ?
No
Yes
NOMATCH=NOMATCH+1
K = I
is I > L ?
Yes
No
is NOMATCH=1 ?
No
Yes
initialize J = 0
J = J+1
is J>2 ?
Yes
No
vary the Jth letter of WORDU through all 26 alphabets and search
is match found ?
Yes
No
put on MATCHLIST
MATCH=MATCH+1
vary the Kth letter of WORDU through all 26 alphabets till match is obtained
put WORDDIC on MATCHLIST
MATCH = MATCH+1
D


D
the program calls the MISSING routine
initialize J = 0 NOMATCH=0
L = L+1
F
J=J+1
copy WORDU in BUFF
insert blank in Jth position of BUFF
is J > 2 ?
No
Yes
vary Jth letter of WORDU through all 26 alphabets and search
initialize K = 0
K = K+1
search dictionary and compare Kth letter of WORDU with Kth letter of WORDDIC
is match found ?
No
Yes
put on MATCHLIST
MATCH=MATCH+1
are both equal ?
Yes
No
is K > L ?
Yes
No
NOMATCH = NOMATCH+1
I = K
E


E
is NOMATCH=1 ?
Yes
No
F
vary the Jth letter of WORDU through all 26 alphabets till complete match is obtained
put WORDDIC on MATCHLIST
MATCH=MATCH+1
the end of the four tests
is MATCH>1 ?
No
Yes
display the MATCHLIST and ask the user to select his correct word
is MATCH=0 ?
No
Yes
display the single word of the MATCHLIST and ask the user for his approval
reject the user's input
STOP
END

CONCLUSION

After studying the problems of interface with interactive computer systems, it is evident that for mass usage of the interactive systems, the user interface plays the major role.

Natural language is an essential factor for mass usage of interactive systems in an area like the medical care service, where the user (physician or patient) is a person not quite acquainted with the computer. Here, the aim of natural language interfaces is to make the interactive systems flexibly and effectively accessed by users with no training in the communication language, and perhaps with no computer training at all.

The analysis of a natural language sentence consists of three major steps : syntactic analysis, semantic analysis, and semantic interpretation with respect to a world model. One problem that effects the user of natural language interactive system is the tolerance of the system to grammatical errors. Thus a careful study must be made before deciding as to how the system should process the given sentence, how far should the system depend on syntactic processing and how to deal with ungrammatical input.

The other problem of interface is spelling errors. In our implementation program, we have designed an interactive spelling correction program that corrects only one type of errors namely the four classes of single error - one letter extra, or one letter wrong, or one letter missing, or two letters transposed. These errors account for 80 percent of spelling errors [40]. Also our program assumes that the given misspelled word contains only one of those four errors.

However, in practice, a user might commit more than one error in one word. This is very probable due to the fact that some words are relatively long (for example, the medical terms). Thus, a proper study must be done on the characteristics of the words in the language and work to develop a method that corrects more than one error, in a given word, would be of great advantage to spelling correction in user interface.

114

APPENDIX - A

A typical consultation session between MYCIN system and physician. This part of this consultation is reproduced from : Computer-Based Medical Consultation: MYCIN. E.H. SHORTLIFFE [6].

The following comments will help the reader follow the sample interactive session :

(1) MYCIN asks questions and makes comments in mixed upper and lower case letters;

(2) MYCIN answers questions and displays decision rules using upper-case letters only;

(3) The physician types words in upper-case letters and these phrases always follow MYCIN's prompt character (**), the double asterisk (note the asterisks distinguish words typed by the physician from upper-case phrases printed by MYCIN);

(4) Explanatory comments have occasionally been inserted in square brackets at the extreme right margin of the page.

*****Welcome from MYCIN*****

Instructions?: (Yes or No)

**YES

I am a computer program named MYCIN that has been designed to advise you regarding an appropriate choice of infectious disease therapy. I understand that you have a patient from whom a possibly positive culture (called CULTURE.1) has been obtained. Please answer the following questions, terminating each response with RETURN.

If you are not certain of your answer, you may modify the response by inserting a certainty factor (a number from 1 to 10) in parentheses after your response. Absolute certainty (10) is assumed for every unmodified answer.

If you wish to change an answer to a previous question, enter CHANGE followed by the number(s) of the question(s) to be altered. Try to avoid going back, however, because the process requires reconsidering the patient from the beginning and it may therefore be slow.

Note that you may also enter UNK (for UNKnown) if you do not know the answer to a question,? if you would like to see some examples of recognized responses,?? If you want to see all recognized responses, the word RULE if you would like to see the decision rule which has generated the question being asked, the word WHY if you would like to see a more detailed explanation of the question, or the letters OA if you would like to interrupt the consultation in order to ask questions regarding the current status of the program's reasoning. If you are ever puzzled about what options are available to you during a consultation, enter the word HELP and a list of options will be listed for you.

Sample Response

Did ORGANISM-1 grow in clumps, chains, or pairs?
**CHAINS (7) PAIRS (3)

SUMMARY OF USER OPTIONS:

UNK	-answer not known
?	-request for examples of recognized responses
??	-request for list of all recognized responses
RULE	-request for current decision rule
OA	-request for enter question-answering mode
WHY	-request for detailed explanation of the question being asked

CHANGE XXX -go back and re-request answer to question number XXX

STOP -Halts the program without completing the consultation

HELP -Prints this list

[At this point the Consultation System (Subprogram 1) begins.]

7-JUN-74 16:27:27

------PATIENT-1------

1) Patient's name: (first-last)

**JEREMIAH SAMPLE

2) SEX:

**MALE

3) Age:

**60

I will refer to the most recent positive culture for which you desire therapeutic advice as:

------CULTURE-1------

4) From what site was the specimen for CULTURE-1 taken?

**BLODD [MYCIN corrects one letter wrong error]

=BLOOD

5) How many days ago was this blood culture (CULTURE-1) obtained?
**2

I will refer to the first offending organism from this blood culture (CULTURE-1) as:

-------ORGANISM-1-------

6) Enter the identity (genus) of ORGANISM-1:

**UNK

7) Staining characteristics of ORGANISM-1(gram):

**?

WHAT IS THE STAIN OF ORGANISM-1?

Expected responses are: ACIDFAST GRAMPOS GRAMNEG
Enter HELP for list of user options.

**GRAMNEG

8) Is ORGANISM-1 a rod or coccus (etc.):

**ROD

9) Were any organisms that were significant (but no longer require therapeutic attention) isolated within the last approximately 30 days?

**YES

-------CULTURE-2-------

10) From what site was the specimen for CULTURE-2 taken?

**SPUTUM

11) How many days ago was this upper-resp culture (CULTURE-2) obtained?

**4½ DAYS

PLEASE USE DECIMAL NUMBERS FOR FRACTIONS
4½ DAYS NOT A RECOGNIZED RESPONSE
WHAT IS THE NUMBER OF DAYS SINCE THE SPECIMEN FOR THIS UPPER-RESP CULTURE WAS ACQUIRED?

Expected responses are: NUMBER OF DAYS
Enter HELP for list of user options.

**4.5

I will refer to the prior organism from this upper-resp culture (CULTURE-2) as:

------ORGANISM-2------

12) Enter the identity (genus) of ORGANISM-2:

**PSEUDOMONAS

13) Any other significant organisms isolated from this upper-resp culture (CULTURE-2)?

**YES

------ORGANISM-3------

14) Enter the identity (genus) of ORGANISM-3:

**KLEBSIELA [System corrects one letter missing]

=KLEBSIELLA

15) Any other significant organisms isolated from this upper-resp culture (CULTURE-2)?

**YES

------ORGANISM-4------

16) Enter the identity (genus) of ORGANISM-4:

**ENTEROB [System corrects suffix error]

=ENTEROBACTER

17) Any other significant organisms isolated from this upper-resp culture (CULTURE-2)?

**NO

18) Any other significant earlier cultures from which pathogens were isolated?

**NO

19) What is the suspected non-sterile portal of entry of ORGANISM-1 into this sterile site?

**RULE

RULE 121

IF: 1) THE SITE OF THE CULTURE IS ONE OF: THOSE SITES THAT ARE NORMALLY STERILE, AND

2) THE PORTAL OF ENTRY OF THE ORGANISM IS THE SAME AS THE SITE OF A PRIOR ORGANISM WITH THE SAME IDENTITY AS THIS CURRENT ORGANISM

THEN: THERE IS FURTHER SUGGESTIVE EVIDENCE (.3) THAT EACH OF THEM IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS THIS CURRENT ORGANISM

**GU

20) Did ORGANISM-1 grow aerobically?

**YES

21) Did ORGANISM-1 grow anaerobically?

**NO

22) Was any chemotherapy recently begun specifically to combat ORGANISM-1?

**QA

[The physician indicates that he wants to enter the Explanation System (Subprogram 2) for a question-answering (QA) session.]

Type HELP for the list of question options

**IQ WHAT DO YOU THINK THE IDENTITY OF ORGANISM-1 IS AT **THIS POINT?

THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS (.333) KLEBSIELLA (.333) ENTEROBACTER (.333)

**RETURN

[Having asked the question of interest, the physician returns to the Consultation System (Subprogram 1).]

22) Was any chemotherapy recently begun specifically to combat ORGANISM-1?

**NO

23) Is Jeremiah Sample (PATIENT-1) a burn patient?

**RULE

RULE047

IF: 1) THE SITE OF THE CULTURE IS BLOOD, AND
2) THE IDENTITY OF THE ORGANISM IS NOT KNOWN WITH CERTAINTY, AND
3) THE STAIN OF THE ORGANISM IS GRAMNEG, AND
4) THE MORPHOLOGY OF THE ORGANISM IS ROD, AND
5) THE PATIENT HAS BEEN SERIOUSLY BURNED

THEN: THERE IS WEAKLY SUGGESTIVE EVIDENCE (.4) THAT THE IDENTITY OF THE ORGANISM IS PSEUDOMONAS

**NO

24) Is the patient's illness with ORGANISM-1 a hospital-acquired infection?

**YES

25) Do you consider Jeremiah Sample (PATIENT-1) to be a compromised host? (alcoholic, immunosuppressed, etc.):

**NO

26) Has Jeremiah Sample (PATIENT-1) had any gu surgery or manipulation of the urinary tract? (e.g., indwelling catheter, cystoscopy, etc.)

**WHY

I AM TRYING TO FIND OUT ABOUT THE IDENTITY OF THE ORGANISM.
I ALREADY KNOW THAT:
THE PORTAL OF ENTRY OF THE ORGANISM IS UPPER-GU
THE MORPHOLOGY OF THE ORGANISM IS ROD
THE STAIN OF THE ORGANISM IS GRAMNEG
THE SITE OF THE CULTURE IS BLOOD
THEREFORE, IF:

1) THE PATIENT HAS NOT HAD A GENITO-URINARY MANIPULATIVE PROCEDURE, AND

2) URINARY-TRACT-INFECTION IS NOT A PROBLEM FOR WHICH THE PATIENT HAS BEEN TREATED

THEN:

THERE IS SUGGESTIVE EVIDENCE (.6) THAT THE IDENTITY OF THE ORGANISM IS E.COLI

(RULE 156)

**YES

27) Does the patient have 'pseudomonas-type' hemorrhagic skin lesions (ecthyma gangrenosum)?

**YES

28) Was the patient treated for the injection with the enterobacter (ORGANISM-4)?

**NO

29) Do you feel that the finding of the enterobacter (ORGANISM-4) at this normally non-sterile site is suggestive of disease in this patient?

**NO

30) Do you believe that the enterobacter (ORGANISM-4) is a contaminant (or normal flora in the case of non-sterile sites)?

**NO

31) Was the patient treated for the infection with the klebsiella (ORGANISM-3)?

**NO

32) Do you feel that the finding of the klebsiella (ORGANISM-3) at this normally non-sterile site is suggestive of disease in this patient?

**NO

33) Do you believe that the klebsiella (ORGANISM-3) is a contaminant (for normal flora in the case of non-sterile sites)?

**UNK

34) Was the patient treated for the infection with the pseudomonas (ORGANISM-2)?

**NO

35) Do you feel that the finding of the pseudomonas (ORGANISM-2) at this normally non-sterile site is suggestive of disease in this patient?

**YES(8)

36) Has the lab reported the in-vitro antibiotic sensitivities of the pseudomonas (ORGANISM-2)?

**NO

37) Has Jeremiah Sample (PATIENT-1) had any operations that you feel may be relevant to the current infection?

**YES

-------OPERATION-1-------

38) How long ago was OPERATION-1 performed?

**11DAYS

39) Were any antibiotics given during OPERATION-1?

**YES

-------DRUG-1-------

40) What drug?

**CEFALOTHIN

=CEPHALOTHIN [MYCIN corrects phonetic spelling error]

41) Were any other antibiotics given at the time of OPERATION-1?

**NO

42) Any other significant operative procedures?

**NO

43) Including this blood culture (CULTURE-1), how many cultures were obtained from this normally sterile site on the same day?

**RULE

RULE 044

IF: 1) THE SITE OF THE CULTURE IS ONE OF: THOSE SITES THAT ARE NORMALLY STERILE, AND

2) THE NUMBER OF CULTURES IN THE SERIES INCLUDING THE CULTURE IS GREATER THAN 1, AND

3) THE NUMBER OF CULTURES CONTAINING THE ORGANISM IS GREATER THAN 1

THEN: THERE IS EVIDENCE (CF = THE QUOTIENT OF THE NUMBER OF CULTURES CONTAINING THE ORGANISM DIVIDED BY THE NUMBER OF CULTURES IN THE SERIES INCLUDING THE CULTURE) THAT THERE IS SIGNIFICANT DISEASE ASSOCIATED WITH THIS OCCURRENCE OF THE ORGANISM

**4

44) How many of the cultures were positive for ORGANISM-1?

**2

[Note that the certainty factor (CF) for RULE 044 is calculated from the answers to questions 43 and 44.]

45) Any other organisms isolated from this blood culture (CULTURE-1) for which you would like a therapeutic recommendation?

**NO

46) Does Jeremiah Sample (PATIENT-1) show any evidence of infection at a site inaccessible to culture (e.g., brain, bone, liver, etc.)?

**NO

47) Were any additional positive specimens recently drawn from Jeremiah sample (PATIENT-1) which revealed one or more organisms for which you wish therapeutic advice?

**NO

[MYCIN has now asked all questions necessary in order to decide (a) whether ORGANISM-1 is significant, and (b) the likely identities of ORGANISM-1.]

My therapy recommendation will be based on the following possible identities of the organism(s) that seem to be significant:

Item 1 The identity of ORGANISM-1 may be PSEUDOMONAS
Item 2 The identity of ORGANISM-1 may be KLEBSIELLA
Item 3 The identity of ORGANISM-1 may be ENTEROBACTER
Item 4 The identity of ORGANISM-1 may be E.COL
Item 5 The identity of ORGANISM-1 may be PROTEUS (Subtype unknown)

[MYCIN now begins asking questions to assist with drug selection.]

48) On a scale of 0 to 4, where higher numbers indicate increasing severity, how would you rate Jeremiah Sample (PATIENT-1)'s degree of sickness?

**4

49) Does Jeremiah Sample (PATIENT-1) have a clinically significant allergic reaction to any antimicrobial agent?

**NO

Considering GENTAMICIN for use against ORGANISM-1...

Rec 1 My preferred therapy recommendation is as follows:

In order to cover for Items 1 2 3 4 5 :

Give the following in combination:

1 GENTAMICIN
Dose: 1.7 MG/KG Q8H - IV OR IM
Comments: MODIFY DOSE IN RENAL FAILURE

2 CARBENICILLIN
Dose: 25 MG/KG Q2H - IV
Comments: MODIFY DOSE IN RENAL FAILURE

Do you wish to see the next choice?

**YES

Considering COLISTIN for use against ORGANISM-1 ...
Rejecting SULFISOXAZOLE for use against ORGANISM-1 ...

Rec 2 Next best therapy recommendation:

In order to cover for Items 1 2 3 4 :

Give: COLISTIN

Dose: 1.25 MG/KG Q6H - IM
Comments: MODIFY DOSE IN RENAL FAILURE

Do you wish to see the next choice?

**NO

[The Consultation System (Subprogram 1) is now finished. Control therefore automatically passes to the Explanation System (Subprogram 2).]

Please ask any simple-English questions regarding the above consultation session or the overall knowledge of the consultation system. When discussing the consultation above, please refer to organisms, cultures, etc. by their numeric names (e.g., ORGANISM-1). The names of rules need not be printed in full when using the PR option ('PR37' is okay), but use the whole name in other contexts (e.g., RULE037). Questions may be specified by the numbers that preceded them above. Enter the word STOP when you have no further questions. Certain standard question types may also be preceded by special codes that will speed up the answers to your questions.

Type HELP for the list of question options.

**HELP

QUESTION-ANSWERING (QA) OPTIONS:

HELP	- PRINTS THIS LIST
EQ	- REQUESTS AN EXPLANATION OF THE SPECIFIED QUESTION(S) FROM THE CONSULTATION
IQ	- PREFIX TO A QUESTION WHICH ASKS ABOUT INFORMATION ACQUIRED BY THE PROGRAM DURING THE CONSULTATION
NOPREFIX	- THIS QUESTION QUERIES CONTENTS OF DECISION RULES IN THE SYSTEM
PR	- REQUESTS THAT SPECIFIED RULE(S) BE PRINTED
STOP	- ESCAPE FROM EXPLANATION SYSTEM
RA	- ENTRY TO RULE-ACQUISITION MODULE FOR RECOGNIZED EXPERTS

**IQ WHAT IS THE FINAL DECISION REGARDING THE IDENTITY OF
**ORGANISM-1?

THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS (.357)
KLEBSIELLA (.177) ENTEROBACTER (.166) E.COLI (.137)
PROTEUS (.118) SERRATIA (.042)

**WHAT MADE YOU THINK ORGANISM-1 MIGHT BE A PROTEUS?

I USED RULE 163 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PROTEUS. THIS GAVE A CUMULATIVE C.F. OF .277. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 26.

I USED RULE 084 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PROTEUS. THIS GAVE A CUMULATIVE C.F. OF .119. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 24.

**PR 84 163

RULE084

IF: 1) THE SITE OF THE CULTURE IS ONE OF: THOSE SITES THAT ARE NORMALLY STERILE, AND
2) THE LIST OF LIKELY PATHOGENS ASSOCIATED WITH THE PORTAL OF ENTRY OF THE ORGANISM IS KNOWN, AND
3) THIS ORGANISM AND THE MEMBERS THAT YOU ARE CONSIDERING AGREE WITH RESPECT TO THE FOLLOWING PROPERTIES: GRAM MORPH AIR

THEN: THERE IS STRONGLY SUGGESTIVE EVIDENCE (.8) THAT EACH OF THEM IS THE IDENTITY OF THE ORGANISM

RULE163

IF: 1) THE MORPHOLOGY OF THE ORGANISM IS ROD, AND
2) THE STAIN OF THE ORGANISM IS GRAMNEG, AND
3) THE IDENTITY OF THE ORGANISM IS NOT KNOWN WITH CERTAINTY, AND
4) THE PATIENT HAS HAD A GENITO-URINARY MANIPULATIVE PROCEDURE

THEN: 1) THERE IS WEAKLY SUGGESTIVE EVIDENCE (.36) THAT THE IDENTITY OF THE ORGANISM IS PSEUDOMONAS, AND
2) THERE IS WEAKLY SUGGESTIVE EVIDENCE (.18) THAT THE IDENTITY OF THE ORGANISM IS PROTEUS, AND

3) THERE IS WEAKLY SUGGESTIVE EVIDENCE (.1) THAT THE IDENTITY OF THE ORGANISM IS SERRATIA, AND
4) THERE IS WEAKLY SUGGESTIVE EVIDENCE (.18) THAT THE IDENTITY OF THE ORGANISM IS ENTEROBACTER, AND
5) THERE IS WEAKLY SUGGESTIVE EVIDENCE (.18) THAT THE IDENTITY OF THE ORGANISM IS KLEBSIELLA

**HOW DID YOU DECIDE THAT PSEUDOMONAS WAS THE MOST
**LIKELY IDENTITY OF ORGANISM-1?

I USED RULE163 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS. THIS GAVE A CUMULATIVE C.F. OF .651. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 26.

I USED RULE 084 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS. THIS GAVE A CUMULATIVE C.F. OF .455. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 24.

I USED RULE050 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS. THIS GAVE A CUMULATIVE C.F. OF .352. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 24.

I USED RULE040 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS. THIS GAVE A CUMULATIVE C.F. OF .833. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 27.

I USED RULE021 TO CONCLUDE THAT THE IDENTITY OF ORGANISM-1 IS PSEUDOMONAS. THIS GAVE A CUMULATIVE C.F. OF .19. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 21.

**PR21

RULE021

IF: THERE IS A PRIOR ORGANISM WHICH MAY BE THE SAME AS THE CURRENT ORGANISM

THEN: IT IS DEFINITE (1) THAT THESE PROPERTIES - IDENTITY - SHOULD BE TRANSFERRED FROM A PRIOR ORGANISM WHICH MAY BE THE SAME AS THE ORGANISM TO THIS CURRENT ORGANISM

**IQ DID YOU DECIDE ORGANISM-2 WAS THE SAME AS
**ORGANISM-1?

ORGANISM-4 (.19) ORGANISM-3 (.19) ORGANISM-2 (.19) IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS ORGANISM-1

**HOW DID YOU DECIDE THAT ORGANISM-2 WAS THE SAME AS
**ORGANISM-1?

I USED RULE003 TO CONCLUDE THAT THE PSEUDOMONAS IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS ORGANISM-1. THIS GAVE A CUMULATIVE C.F. OF .19. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 21.

I USED RULE004 TO CONCLUDE THAT THE PSEUDOMONAS IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS ORGANISM-1. THIS GAVE A CUMULATIVE C.F. OF .133. THE LAST QUESTION ASKED BEFORE THE CONCLUSION WAS MADE WAS 18.

**PR 45

RULE004

IF: 1) THE IDENTITY OF THE ORGANISM IS NOT KNOWN WITH CERTAINTY, AND
2) THIS CURRENT ORGANISM AND PRIOR ORGANISMS OF THE PATIENT AGREE WITH RESPECT TO THE FOLLOWING PROPERTIES: MORPH GRAM

THEN: THERE IS WEAKLY SUGGESTIVE EVIDENCE (.4) THAT EACH OF THEM IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS THE ORGANISM

RULE005

IF: THIS CURRENT ORGANISM AND A PRIOR ORGANISM WITH THE SAME IDENTITY AS THE ORGANISM AGREE WITH RESPECT TO THE FOLLOWING PROPERTIES:AIR

THEN: THERE IS WEAKLY SUGGESTIVE EVIDENCE (.2) THAT EACH OF THEM IS A PRIOR ORGANISM WITH THE SAME IDENTITY AS THE ORGANISM

**EQ 48

48 QUESTION 48 WAS ASKED IN ORDER TO FIND OUT JEREMIAH SAMPLE'S DEGREE OF SICKNESS (ON A SCALE OF 4) IN AN EFFORT TO EXECUTE RULE068.

**PR 68

RULE068

IF: 1) THE THERAPY UNDER CONSIDERATION IS GENTAMICIN, AND
2) THE IDENTITY OF THE ORGANISM MAY BE PSEUDOMONAS, AND
3) THE PATIENT'S DEGREE OF SICKNESS (ON A SCALE OF 4) IS GREATER THAN 3, AND
4) A - THE PATIENT IS NOT ALLERGIC TO ANTIBIOTICS, OR B-CARBENICILLIN IS NOT A DRUG TO WHICH THE PATIENT IS ALLERGIC

THEN: RECORD THE FOLLOWING AS A MORE APPROPRIATE THERAPY THAN GENTAMICIN: THE COMBINATION OF GENTAMICIN WITH CARBENICILLIN

**STOP

SO) Do you wish advice on another patient?

**NO

THANK-YOU

*****

APPENDIX - B

A table of the misspelled words that were rejected by DAMERAU'S program. 80 percent of the misspelled words that were rejected fell into one of the four classes of single error-one letter was wrong, or one letter was missing, or an extra letter had been inserted, or two adjacent letters had been transposed. We also notice that 91 percent of the four classes of single error occured in the third or subsequent letters of the misspelled word. From: DAMERAU, F.J.[40].

Correct Spelling	Incorrect Spelling	Single Error
ABSORBENT	ABSORBANT	X
ABSORPTION	ABSORBTION	X
ACCOMODATE	ACCOMODATE	X
ACQUIESCE	AQUIESE	
ANALYZE	ANALIZE	X
ANTARCTIC	ANTARTIC	X
ASININE	ASSININE	X
ASSISTANCE	ASSISTENCE	X
AUXILIARY	AUXILLARY	X
BANANA	BANANNA	X
BANKRUPTCY	BANKRUPCY	X
BRETHREN	BRETHEREN	X
BRITAIN	BRITIAN	X
BUOYANCY	BOUYANCY	X
CATEGORY	CATEGOREY	
CHAUFFEUR	CHAUFFUER	X
CHIMNEYS	CHIMNIES	
COLISEUM	COLOSIUM	
COLOSSAL	COLLOSAL	
COMMITMENT	COMMITMENT	X

Correct Spelling	Incorrect Spelling	Single Error
COMMITTEE	COMMITEE	X
CONCEDE	CONSEDE	X
CONSCIENTIOUS	CONSCIENTIOUS	X
CONSENSUS	CONCENSUS	X
CONTROVERSY	CONTROVERCY	X
CORRUGATED	CORRIGATED	X
CYNICAL	SYNICAL	X
DEUCE	DUECE	X
DEVELOP	DEVELLOPE	
DIGNITARY	DIGNATARY	X
DISAPPOINT	DISAPOINT	X
DRASTICALLY	DRASTICLY	
ECSTASY	ECTACY	X
EMBARRASS	EMBARASS	X
EXAGGERATE	EXAGERATE	X
EXISTENCE	EXISTANCE	X
EXTENSION	EXTENSION	X
FEBRUARY	FEBUARY	X
FIERY	FIREY	X
FILIPINOS	PHILIPINOES	
FLAMMABLE	FLAMABLE	X
FORTHRIGHT	FORTRIGHT	X
FORTY	FOURTY	X
FULFILL	FULLFIL	
GNAWING	KNAWING	X
GOVERNMENT	GOVERMENT	X
GRAMMAR	GRAMMER	X
HEARTRENDING	HEARTRENDERING	
HEMORRHAGE	HEMORRAGE	X
HINDRANCE	HINDERENCE	
HYGIENE	HYGEINE	X
IDIOSYNCRASY	IDIOCYNCRACY	
INCENSE	INSENSE	X
INCIDENTALLY	INCIDENTLY	
INFALLIBLE	INFALABLE	
INOCULATE	INNOCULATE	X
INSISTENCE	INSISTANCE	X
INTERCEDE	INTERSEDE	X
INTERFERED	INTERFERRED	X
JEOPARDIZE	JEPRODISE	
KIMONO	KIMONA	X

Correct Spelling	Incorrect Spelling	Single Error
LICENSE	LISENCE	X
LIQUEFY	LIQUIFY	X
MAINTENANCE	MAINTAINANCE	
MANAGEMENT	MANAGMENT	X
MANEUVER	MANUVEUR	
MORTGAGED	MORTGAUGED	X
NICKEL	NICKLE	X
NINETYNINTH	NINTYNINETH	
NOWADAYS	NOWDAYS	X
OCCASIONALLY	OCASSIONALY	X
OCCURRENCE	OCCURENCE	X
PAMPHLET	PHAMPLET	
PERMISSIBLE	PERMISSABLE	X
PERSEVERANCE	PERSEVERENCE	X
PERSUADE	PURSUADE	X
PHILIPPINES	PHILLIPINES	
PITTSBURGH	PITTSBURG	X
PLAGIARISM	PLAIGARISM	X
PLAYWRIGHT	PLAYWRITE	
PRAIRIE	PRARIE	X
PRECEDING	PRECEEDING	X
PRECIPICE	PRESIPICE	X
PREFERABLE	PREFERRABLE	X
PRESUMPTUOUS	PRESUMPTOUS	X
PRIVILEGE	PRIVELEGE	X
PROPELLER	PROPELLOR	X
PSYCHOLOGICAL	PSYCOLOGICAL	X
PUBLICLY	PUBLICALLY	
PURSUER	PERSUER	X
QUESTIONNAIRE	QUESTIONAIRE	X
RECIPIENT	RESIPIENT	X
RELEVANT	REVELENT	
RENOWN	RENOUN	X
REPEL	REPELL	X
RHAPSODY	RAPHSODY	
RHODODENDRON	RHODODRENDON	
RHUBARB	RUBHARB	X
RHYTHM	RYTHM	X
SACRILEGIOUS	SACRELIGIOUS	
SAFETY	SAFTY	X
SCISSORS	SISSERS	
SEIZE	SIEZE	X
SEPARATE	SEPERATE	X

Correct Spelling	Incorrect Spelling	Single Error
SHEPHERD	SHEPERD	X
SIMILAR	SIMILIAR	X
SINCERITY	SINCERETY	X
SOUVENIR	SOUVINER	
SPECIMEN	SPECIMENT	X
SUING	SUEING	X
SURREPTITIOUS	SUREPTITOUS	
TRANSFERABLE	TRANSFERRABLE	X
UNPARALLELED	UNPARALELLED	
USAGE	USEAGE	X
VEGETABLE	VEGATABLE	X
WEDNESDAY	WEDENSDAY	X
WEIRD	WIERD	X

REFERENCES

[1] Sager, N. (1981) : "Natural Language Information Processing." A Computer Grammar of English and Its Applications. ADDISON-WESLEY PUBLISHING COMPANY.

[2] Hayes-Roth, F.K., Waterman, D.A., and Lenat, D.B. eds. (1983) : "BUILDING EXPERT SYSTEMS." ADDISON-WESLEY PUBLISHING COMPANY.

[3] Shortliffe, E.H. (1977) : "A Rule-Based Approach to the Generation of Advice and Explanations in Clinical Medicine." In W. Schneider and a-l. Sagrall hein (eds.) Computational Linguistics in Medicine. Proceedings, IFIP Working Conference, Uppsala, Sweden.

[4] Tennant, H. (1981) : "Natural Language Processing." Petrocelli Books, Inc. New York.

[5] Moto-Oka, T. ed. (1981): "Fifth Generation Computer Systems." Proceedings of the International Conference on FGCS, TOKYO, JAPAN. North-Holland Publishing Company.

[6] Shortliffe, E.H. (1976) : "Computer-Based Medical Consultation : MYCIN." American Elsevier Publishing Company, New York.

[7] Peterson, J.L. (1980) : "Computer Programs for Spelling Correction." In G. Goos and J. Hartmanis (eds.) Lecture Notes in Computer Science, 96, Springer-Verlag, New York.

[8] Belzer, J., Holzman, A.G., and Kent, A. eds. (1976) : "Encyclopedia of Computer Science and Technology." Volume 5. MARCEL DEKKER, Inc. New York.

[9] Alberga, C.N. (1967) : "String Similarity and Misspellings." Communications of the ACM, Volume 10, Number 5, pages 302-313.

[10] Barr, A., and Feigenbaum, E.A. eds. (1981) : "The Hand Book of Artificial Intelligence." Volume 1. PITMAN BOOKS LIMITED, U.K.

[11] Barr, A., and Feigenbaum, E.A. eds. (1982) : "The Hand Book of Artificial Intelligence." Volume 2. Heuris Tech. Press.

[12] Buchanan, B.G., and Duda, R.O. (1983) : "Principles of Rule-Based Expert Systems." In C.M. Yovits (ed), Advances in Computers. Volume 22. Academic Press, New York.

[13] Knuth, D.E. (1973) : "The Art of Computer Programming." Volume 3 : Sorting and Searching. ADDISON-WESTLEY PUBLISHING COMPANY.

[14] Peterson, J.L. (1980) : "Computer Programs for Detecting and Correcting Spelling Errors." Communications of the ACM, Volume 23, Number 12, Pages 676-687.

[15] Bundy, A., Burstall, R.M., Weir, S., and Young, R.M. (1978) : "Artificial Intelligence : An Introductory Course." EDINBURGH UNIVERSITY PRESS, U.K.

[16] Spencer, D.D. (1978) : "Data Processing : An Introduction." Charles E. Merrill Publishing Company, Columbus, Ohio.

[17] Shortliffe, E.H. (1974b) : "MYCIN : A Rule-Based Computer Program For Advising Physicians Regarding Antimicrobial Therapy Selection." Doctoral Dissertation, Stanford University, Stanford Artificial Intelligence Laboratory.

[18] Press, L. (1983) : "Low-Cost Word Processing." Addison-Wesley Publishing Company.

[19] Durham, I., Lamb, D.A., and Saxe, J.B. (1983) : "Spelling Correction in User Interfaces" Communications of the ACM, Volume 26, Number 10, Pages 764-773.

[20] Dehning, W., Essing, H., and Maass, S. (1980) : "The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs." In G. Goos and J. Hartmanis (eds.), Lecture Notes in Computers, 110, Springer-Verlag, New York.

[21] Nau, D.S. (1983) : "Expert Computer Systems." IEEE Computer, Volume 16, Number 2, Pages 63-85.

[22] McCalla, G., and Cercone, N. (1983) : "Approaches to Knowledge Representation." IEEE Computer, Volume 16, Number 10 (A Special Issue on Knowledge Representation).

[23] MARTIN, J. (1977) : "Computer Data-Base Organization." Second Edition, PRINTICE-HALL, NEW JERSEY.

[24] Ullman, J.D. (1980) : "Principles of Database Systems." STANFORD UNIVERSITY, COMPUTER SCIENCE PRESS.

[25] Waltz, D.L. (1978) : "An English Language Question Answering System for a Large Relational Database." Communications of the ACM, Volume 21, Number 7, Pages 526-539.

[26] Hill, D.R. (1971) : "Man-Machine Interaction Using Speech." In Franz L. ALT and MORRIS RUBINOFF (eds.) Advances in Computers, Volume 11, Academic Press, New York.

[27] Clancey, W.J. (1983) : "The Epistemology of a Rule-Based Expert System - A Framework for Explanation." Artificial Intelligence Journal, Volume 20, Pages 215-251.

[28] Kaplan, S.J. (1982) : "Cooperative Responses from a Portable Natural Language Query System." Artificial Intelligence Journal, Volume 19, Pages 165-187.

[29] Thompson, F.B., and Thompson, B.H. (1975) : "Practical Natural Language Processing : The REL System as Prototype." In Morris Rubinoff and Marshall. c. Yovits (eds.), Advances in Computers, Volume 13, Academic Press, New York.

[30] Simons, G.L. (1983) : "Towards Fifth Generation Computers." National Computing Centre, England.

[31] Raphael, B. (1976) : "The Thinking Computers : Mind Inside Matter." W.H. Freeman and Company, SanFrancisco.

[32] De Jong, G. (1983) : "Artificial Intelligence Implications for information Retrieval." PROCEEDINGS OF THE SIXTH ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL. Special Interest Group on Information Retrieval, Volume 17, Number 4, Summer 1983, Pages 10-17.

[33] Ledgard, H., Whiteside, J.A., Singer, A., and Seymour, W. (1980) : "The Natural Language of Interactive Systems." Communications of the ACM, Volume 23, Number 10, Pages 556-563.

[34] Radue, J. (1983) : "On the Design of an Interactive Spelling Dictionary for Personal Computers." ACM CONFERENCE ON PERSONAL AND SMALL COMPUTERS. SIGPC Notes, Volume 6, Number 2, SAN DIEGO, CA. (December 7-8), Pages 197-199.

[35] Hayes, P., Ball, E., and Reddy, R. (1980) : "Computers With Natural Communication Skills." Computer Science Research Review (1979-1980) Carnegie-Mellon University.

[36] Ralston, A., and Meek, C.L. eds. (1976) : "Encyclopedia of Computer Science." PETROCELLI, New York.

[37] Hunt, E.B. (1975) : "Artificial Intelligence Academic Press, New York.

[38] Quinlan, J.R., (1980) : "An Introduction to Knowledge-Based Expert Systems." The Australian Computer Journal, Volume 12, Number 2, Pages 56-62.

[39] Bradley, J. (1983) : "Introduction to Data Base Management in Business." Holt-Saunders International Editions, Japan.

[40] Dameran, F.J. (1964) : "A Technique for Computer Detection and Correction of Spelling Errors." Communications of the ACM, Volume 7, Number 3, Pages 171-176.

[41] Basa, R. (1981) : "A Study on Representation of a Class of Problems of Pictorial Inference in Terms of Predicate Calculus." M.Phil Dissertation School of Computer Science, Jawaharlal Nehru University, New Delhi.

[42] INFOTECH STATE OF THE ART REPORT (1979) : "THE COGNITIVE INTERFACE - DIALOGUE AND LANGUAGE." MAN/COMPUTER COMMUNICATION Volume 1, ANALYSIS AND BIBLIOGRAPHY. Infotech International, England.