

ROTE LEARNING BY VERIFICATION

*Dissertation submitted to
Jawaharlal Nehru University
in partial fulfilment of the requirements
for the award of the degree of*
MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND TECHNOLOGY

by

SHIN IN OK

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


JANUARY 1993

C E R T I F I C A T E

This work , embodied in the dissertation titled

" ROTE LEARNING BY VERIFICATION "

has been done by Shin In Ok , a bona fide student of the School of Computer and Systems Sciences , Jawaharlal Nehru University, New Delhi. This work is original and has not been submitted for any degree or diploma in any other university or institute.



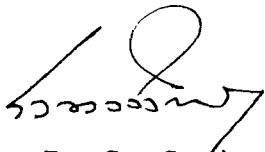
Jan 5, 1993

Professor K. K. Bharadwaj

School of Computer and Systems Sciences

Jawaharlal Nehru University

New Delhi .



Professor R. G. Gupta, Dean

School of Computer and Systems Sciences

Jawaharlal Nehru University

New Delhi .

ACKNOWLEDGEMENT

I wish to express my deep gratitude to Professor K. K. Bharadwaj who got me interested in the field of artificial intelligence and machine learning. His keen interest, inspiring guidance and constant encouragement throughout the course of this work will be unforgettable for me.

I thank Professor R. G. Gupta for providing me the facilities of the school to help me in carrying out my project.

I thank my friends in the M.Tech. class, whose stimulating company provided a good environment and encouragement for this work.

Thanks are also due to all the people who directly or indirectly helped me in the course of my work.



SHIN IN OK

ABSTRACT

In this work methodology and implementation of a learning scheme, **"Rote Learning by Verification"** is discussed. This learning scheme has been implemented for a Learning Tutoring System (**LEARNUTOR**). The first phase of the system is Game Playing & Learning. This is a step towards making an expert system capable of automatically acquiring knowledge, according to its needs. The expert system resembles a student, who is gradually acquiring knowledge in a particular domain of discourse; and eventually qualifying as an expert. The second phase of the system is Tutoring System and it is shown how the proposed system, LEARNUTOR, can be viewed as a tool in education.

C O N T E N T S

| | | |
|------------|---|----|
| CHAPTER 1 | INTRODUCTION | |
| 1.1 | LEARNING ----- | 1 |
| 1.1.1 | General Learning Model ----- | 1 |
| 1.1.2 | Symbolic Learning Techniques ----- | 3 |
| 1.1.3 | Neural Network Based Learning ----- | 9 |
| 1.1.4 | Genetic Algorithm Based Learning ----- | 11 |
| 1.2 | INTELLIGENT TUTORING SYSTEM (ITS) ----- | 13 |
| CHAPTER 2 | ROTE LEARNING BY VERIFICATION | |
| 2.1 | INTRODUCTION AND MOTIVATION ----- | 16 |
| 2.2 | THE LEARNING TUTORING SYSTEM : LEARNUTOR ----- | 19 |
| 2.2.1 | Architecture ----- | 19 |
| 2.2.2 | What the LEARNUTOR does ? ----- | 20 |
| 2.2.3 | Where is the learning element introduced ? ----- | 20 |
| 2.2.4 | Parallelism with a student learning model ----- | 21 |
| 2.3 | LEARNUTOR : Organization ----- | 23 |
| 2.3.1 | Game Playing & Learning System : Organization --- | 24 |
| 2.3.2 | Game Playing & Learning System : Flow Pattern --- | 25 |
| 2.3.3 | Tutoring System : Flow Pattern ----- | 27 |
| CHAPTER 3 | IMPLEMENTATION | |
| 3.1 | QUERY SESSION MODULE (QSM): THE INFERENCE ENGINE -- | 29 |
| 3.2 | THE KNOWLEDGE BASE ----- | 30 |
| 3.2.1 | The Primary Knowledge Base (PKB) ----- | 31 |
| 3.2.2 | The Secondary Knowledge Base (SKB) ----- | 31 |
| 3.2.3 | Learning Module (LM) ----- | 32 |
| 3.3 | HOW IT ALL WORKS : IMPLEMENTATION DETAILS ----- | 33 |
| 3.3.1 | Structuring the Knowledge Base ----- | 33 |
| 3.3.2 | Constructing the QSM : Inference Engine ----- | 34 |
| 3.3.3 | Building the Learning Module (LM) ----- | 36 |
| 3.3.3.1 | ETM Construct ----- | 36 |
| 3.3.3.2 | SKAM Construct ----- | 36 |
| 3.3.3.3 | CM Construct ----- | 37 |
| 3.4 | MENUS ----- | 37 |
| 3.5 | GAME PLAYING & LEARNING SESSION ----- | 39 |
| 3.6 | TUTORING SESSION ----- | 43 |
| CHAPTER 4 | CONCLUSION ----- | 44 |
| REFERENCES | ----- | 45 |

CHAPTER 1.

INTRODUCTION

1.1 LEARNING

Machine Learning is certainly one of the ultimate research goals in the field of **Artificial Intelligence**. Its application is very evident as it is seen for automatic knowledge for expert systems, which circumvents the problem of building the knowledge base of an expert system.

Learning is an inherent feature of intelligence. The process of learning may be defined as a directed change in the knowledge structure that improves the future performance of the system.

Learning can be accomplished using a number of different methods. For examples, we can learn by memorizing facts, by being told, or by studying examples like problem solutions. Learning requires that new knowledge structures be created from some form of input stimulus. This new knowledge must then be assimilated into a knowledge base and be tested in some way for its utility. Testing means that the knowledge should be used in the performance of some task from which meaningful feedback provides some measure of the accuracy and usefulness of the newly acquired knowledge.

1-1-1. General Learning Model :

A general learning model [11] is depicted in Figure 1 where the environment has been included as part of the overall learner system. The environment may be regarded as either a form of nature which

produces random stimuli or as a more organized training source such as a teacher which provides carefully selected training examples for the learner component. The actual form of environment used will depend on the particular learning program. In any case some representation language must be assumed for communication between the environment and the learner. The language may be the same representation scheme as that used in the knowledge base (such as a form of predicate calculus). When they are chosen to be the same, we say the single representation trick is being used. This usually results in a simple implementation since it is not necessary to transform between two or more different representations.

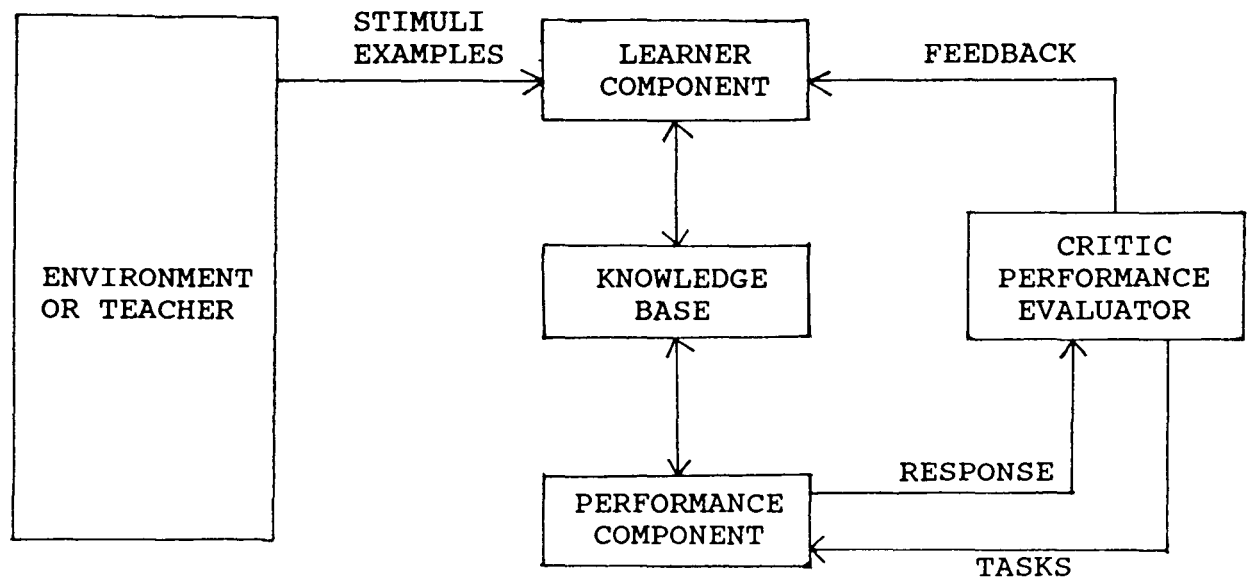


Figure 1 : General Learning Model

For some systems the environment may be a user working at a keyboard. Other systems will use program modules to simulate a particular environment. In even more realistic cases, the system will have real physical sensors which interface with some world

environment.

Inputs to the learner component may be physical stimuli of some type of descriptive, symbolic training examples. The information conveyed to the learner component is used to create and modify knowledge structures in the knowledge base. This same knowledge is used by the performance component to carry out some tasks, such as solving a problem, playing a game, or classifying instances of some concept.

When given a task, the performance component produces a response describing its actions in performing the task. The critic module then evaluates this response relative to an optimal response.

Feedback, indicating whether or not the performance was acceptable, is then sent by the critic module to the learner component for its subsequent use in modifying the structures in the knowledge base. If proper learning was accomplished, the systems performance will have improved with the changes made to the knowledge base.

The cycle described above may be repeated a number of times until the performance of the system has reached some acceptable level, until a known learning goal has been reached, or until changes cease to occur in the knowledge base after some chosen number of training examples have been observed.

1.1.2 Symbolic Learning Techniques

In what follows, it will be helpful to adopt a classification or taxonomy of learning types to serve as a guide. In studying or comparing differences among them. One can develop learning taxonomies based on the type of knowledge representation used (predicate calculus, rules, frames), the type of knowledge learned (concepts,

game playing problem solving), or by the area of application (medical diagnosis, scheduling, prediction, and so on). The classification is independent of the knowledge domain and the representation scheme used. It is based on the type of inference strategy employed on the methods used in the learning process.

The five different learning methods under this taxonomy are:

- Rote Learning
- Learning by Instruction
- Learning by Deduction
- Learning by Analogy
- Inductive Learning.

A Rote Learning:

Learning by memorization is the simplest form of learning. It requires the least amount of inference and is accomplished by simply copying the knowledge in the same form that it will be used directly into the knowledge base. We use this type of learning when we memorize multiplication tables, for example.

Rote memorization can be seen as an elementary learning process, not powerful enough to accomplish intelligent learning on its own (because not everything that needs to be known in any nontrivial domain can be memorized). but an inherent and important part of a learning system. All learning systems must remember the knowledge that they have acquired so that it can be applied in the future. In a rote learning system, the knowledge has already been gained by some method and is in a directly usable form. Other more sophisticated learning systems first acquire the knowledge from examples or from

advice and then memorize it. Thus all learning systems are build on a rote learning process that stores, maintains, and retrieves knowledge in a knowledge base.

B Learning by Instruction (Learning by being told):

A sightly more complex form of learning is by direct instruction. This type of learning requires more inference than rote learning since the knowledge must be transformed into an operational form before being integrated into the knowledge base. We use this type of learning when a teacher presents a number of facts directly to us in a well organized manner.

C Learning by deduction:

It is accomplished through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationship are logically drived. For example, we could learn deductively that Sue is the cousin of Bill, if we have knoweldge of Sue and Bill's parents and rules for the cousin relationship. Deductive learning usually requires more inference than the other methods. The inference method used is, of course, a deductive type, which is a valid form of inference.

D Learning by Analogy:

Analogies are similarities or likenesses between things otherwise different. Things which are similar in some respects tend to be similar in other respects. The things or participants in analogies

are unlimited. They may be physical objects, concepts, problems and their solutions, plans, situations, episodes, and so forth. Analogies play a dominant role in human reasoning and learning processes.

Analogies appear in different guises and at varied levels of abstraction. Simplex analogies are the word-object of geometric ones often found in STA or GRE tests.

They take the form:

A ~ B (A is like B) or more generally

A:B :: C:D (A is to B as C is to D),

where one of the components is missing.

For examples, the type of word-object and geometrical analogies typically found in aptitude of GRE tests are given by

| | |
|--------------|-----------------|
| (a) | (b) |
| House : Hut | Water : Dam |
| Tree : _____ | _____ : Battery |
| (c) | (d) |
| Green : Go | _____ |
| Red : _____ | # O * |
| | O * |
| | * # O |
| | _____ |

Analogical learning is the process of learning a new concept or solution through the use of similar known concepts or solutions. We use this type of learning when solving problems on an exam where previously learned examples serve as a guide or when we learn to drive a truck using out knowledge of car driving. We make frequent use of

analogical learning. This form of learning requires still more inferring than either of the previous forms, since difficult transformations must be made between the known and unknown situations.

Patrick Winston [17] developed programs that reason about relationships, motives, and consequent actions that occur among people. Using relationships and acts of actors in one story (such as Macbeth) the program was able to demonstrate that analogous results occurred in different stories (such as Hamlet) when there were similarities among the relationships and motives of the second group of characters. The programs could also learn through the analogical reasoning process. For example, when a teacher declared that voltage, current; and resistance relationships were the those of water pressure, flow, and pipe resistance, the system was able to learn basic results in electrical circuits and related laws such as Ohm's law from laws of hydraulics

E Inductive Learning:

Inductive learning also one that is used frequently by humans. It is powerful form of learning which the analogical learning, also requires the use of inductive inference, a form of invalid but useful inference. We use inductive learning when we formulate a general concept after seeing a number of instances or examples of the concepts. For example, we learn the concepts of color or sweet taste after experiencing the sensations associated with several examples of colored objects or sweet foods.

In the abstract, we can view the output of inductive learning as a set of production rules. These are rules of the form :

In <situation> Do <action>

The "Action" may be something over, or an internal action, or even an inference. We will some times use the word concept for the right-hand side of such a rule, and the phrase concept definition or pattern for the left-hand side.

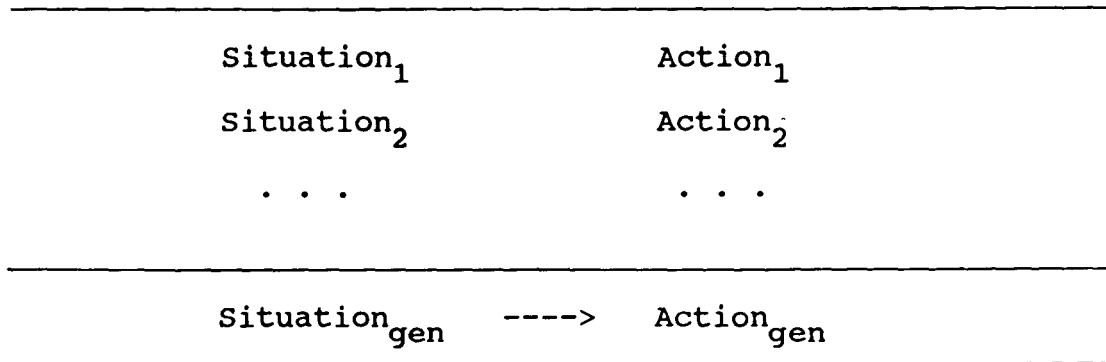


Figure 2 : Inductive inference in the abstract

We schematize inductive learning as shown in Figure 2 . From one or more instances in which action_i was the appropriate action in response to situation_i, we infer that the general version, action_{gen} is the appropriate type of action in reponse to the general situation type, situation_{gen}. As an example, consider Figure 3 .

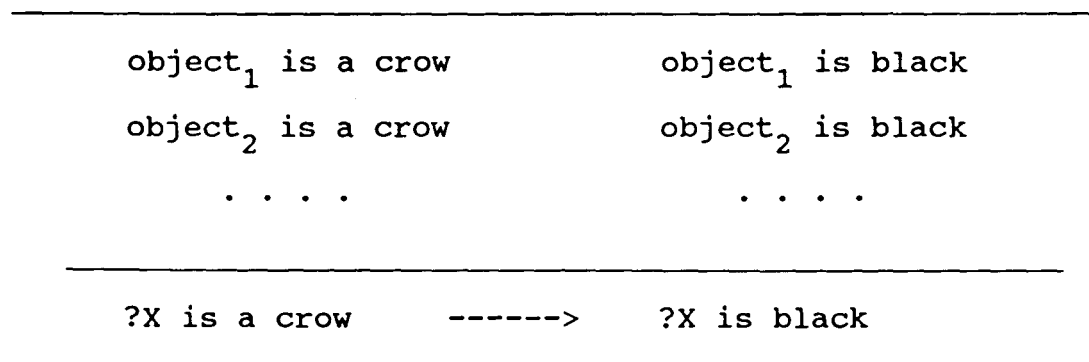


Figure 3 : Inductive inference of "All crows are black"

Here the "action" is to make an inference, namely, that an object

is black. (In this case,our "concept/definition" terminology is misleading, since being a crow is not part of the definition of being black.)

1.1.3 Neural Network Based Learning:

Neural networks were originally inspired as being models of the nervous system. They are greatly simplified models to be sure (neurons are known to be fairly complex processors). Even so, they have been shown to exhibit many "intelligent" abilities, such as learning, generalization, and abstraction.

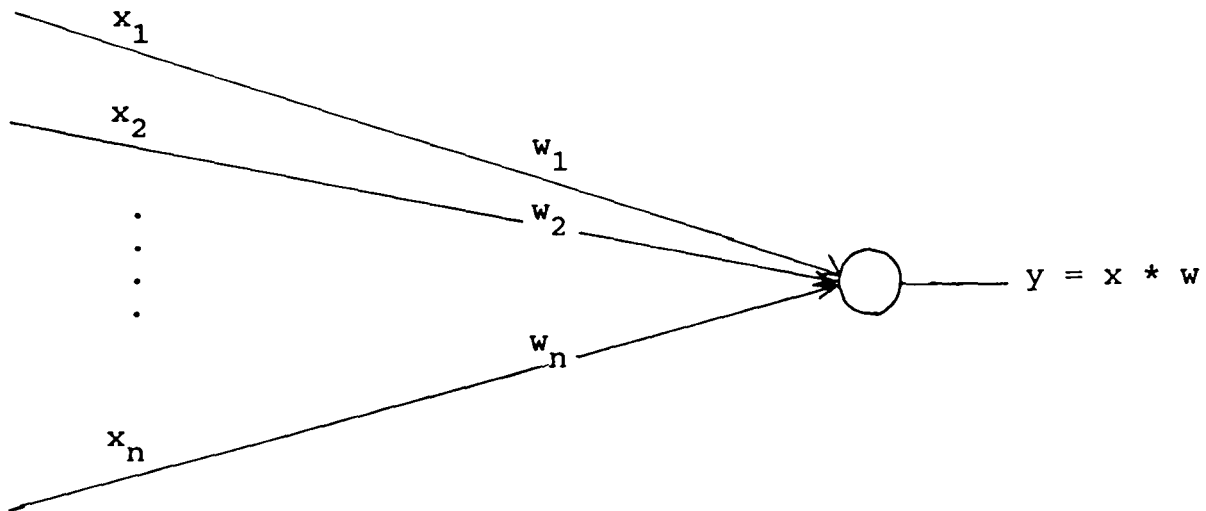


Figure 4 : Model of a Single Neuron (Node)

Neural networks are large networks of simple processing elements or nodes which process information dynamically in response to external inputs. The nodes are simplified models of neurons. The knowledge in a neural network is distributed throughout the network in the form of internode connections and weighted links which form the inputs to the nodes. The link weights serve to enhance or inhibit the input stimuli

values which are then added together at the nodes. If the sum of all the inputs to a node exceeds some threshold value T , the node executes and produces an output which is passed on to other nodes or is used to produce some output response.

A single node is illustrated in Figure 4. The inputs to the node are the values $x_1, x_2, x_3, \dots, x_n$, which typically take on values of $-1, 0, 1$, or real values within the range $(-1, 1)$. The weights $w_1, w_2, w_3, \dots, w_n$ correspond to the synaptic strengths of a neuron. They serve to increase or decrease the effects of the corresponding x_i . Input values the sum of the products $x_i * w_i, i=1, 2, \dots, n$, some as the total combined input the mode. if this sum is large enough to exceed the threshold amount T , the node fires, and produces an output y , an activation function values placed on the nodes output links. This output may then be the input the other nodes or the final output response from the network.

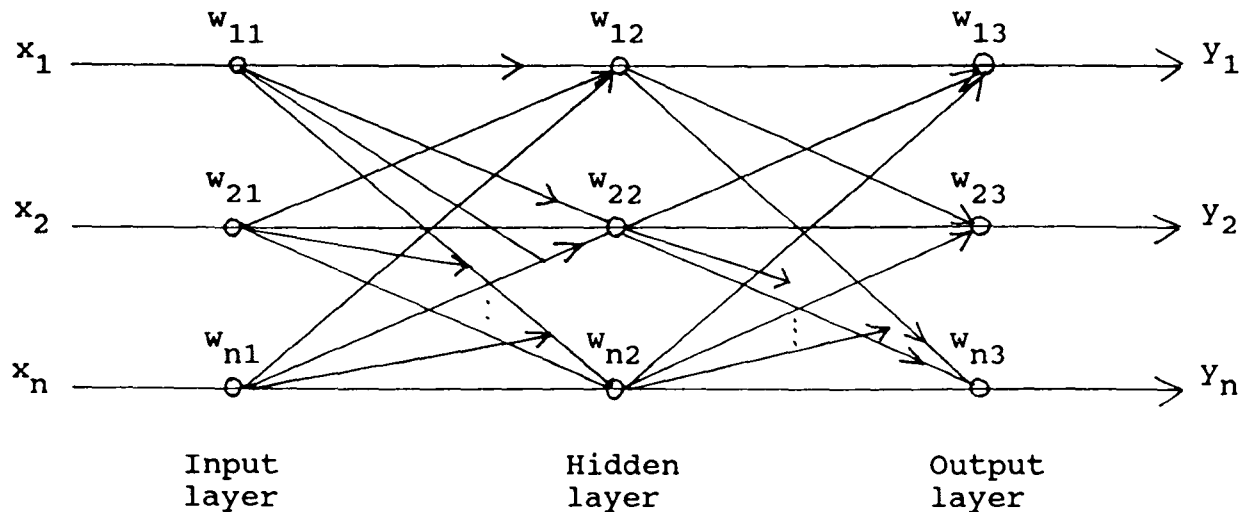


Figure 5 : A Multilayer Neural Network

This Figure 5 illustrates three layers of a number of

interconnected nodes. The first layer serves as the input layer, receiving inputs from some set of stimuli. The second layer (called the hidden layer) receives inputs from the first layer and produces a pattern of inputs to the third layer, the output layer. The pattern of outputs from the final layer are the network's responses to the input stimuli patterns. Input links to layer j ($j= 1,2,3$) have weights w_{ij} for $i= 1,2,\dots,n$.

The interconnections and weights W in the neural network store the knowledge possessed by the network. These weights must be present or learned in some manner. When learning is used, the process may be either supervised or unsupervised. In the supervised case, learning is performed by repeatedly presenting the network with an input pattern and a desired output response. The training examples then consist of the vector pairs (x,y') , where x is the input pattern and y' is the desired output response pattern. The weights are then adjusted until the difference between the actual output response y and the desired response y' are the same, that is until $D= y-y'$, is near zero.

In unsupervised learning, the training examples consists of the input vectors x only no desired response y' is the available to guide the system. instead the learning process must find the weights w_{ij} with no knowledge of the desired output response.

1.1.4 Genetic Algorithm Based Learning:

Genetic algorithm learning are based on models of neural adaptation and evolution. These learning systems improve their performance through processes which model population genetics and

survival of the fittest. They have been studied since the early 1960s (Holland 1962, 1975)

In the field of genetics, a population is subject to an environment which places demands on the members. The members which adapt well are selected for mating and reproduction. The offspring of these better performers inherit genetic traits from both their parents. Members of this second generation of offspring which also adapt well are then selected for mating and reproduction and the evolutionary cycle continues. Poor performers die off without leaving offspring. Good performers produce good offspring and they, in turn, perform well. After some number of generations, the resultant population will have adapted optimally or at least very well to the environment.

Genetic algorithm systems start with a fixed size population of data structures which are used to perform some given tasks. After requiring the structures to execute the specified tasks some number of times, the structures are rated on their performance and a new generation of data structures is then created. The new generation is created by mating the higher performing structures to produce offspring. These offspring and their parents are then retained for the next generation while the poorer performing structures are discarded. The basic cycle is illustrated in Figure 6.

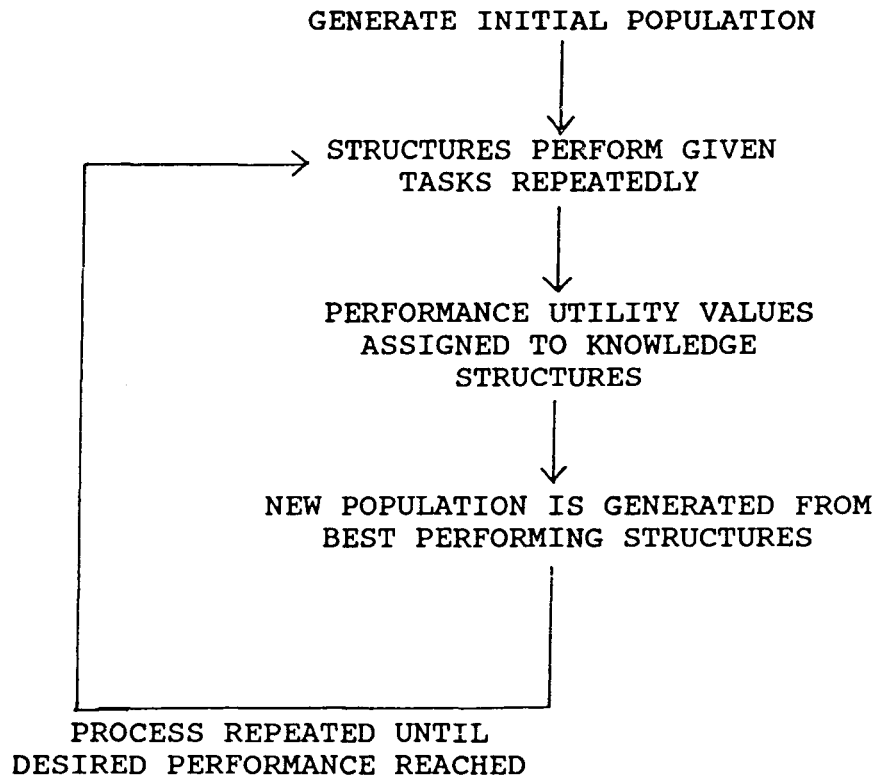


Figure 6 : Genitic Algorithm

1.2 INTELLIGENT TUTORING SYSTEM (ITS):

An Inteligent Tutoring System is a computer program that uses AI teachniques for representing knowledge and carrying on an interaction with a student(Sleeman & Brown [15]). Among the most well_known systems are WHY(Collins [10],uses Socratic principles for teaching causal reasoning in domains like meteorology), SOPHIE (Brown, Burton, & Bell [5], provides a "simulated workbench" in which a student can test eletronics troubleshooting skills), and WEST(Burton & Brown [7], coaches a game_player on methods and strategies for exploiting game rules). This work derives from earlier efforts in computer_aided instruction,but differs in its attempt to use a principled or

theoretical approach. First and foremost, this entails separating subject material from teaching method, as opposed to combining them in ad_hoc programs, By starting teaching methods explicitly, one gains the advantages of economical representation (the methods can be applied flexibly in many situations and even multiple problem domains) and the discipline of having to lay out subject material in a systematic, structured way, independently of AI to these instructional systems is in the representation of teaching methods and domain knowledge. Ideally, this enterprise involves having a theory of teaching and nature of the knowledge to be taught.

When we separate domain knowledge from the procedures that will use it, we say that we are representing knowledge "declaratively" (Winograd [16]) (with respect to those procedures). For example, in a medical domain, we would represent links between data and diagnoses so they could be accessed and used for solving any given problem. A strong advantage of this approach is that the tutoring system can cope with arbitrary student behavior: no matter what order the student chooses to collect data (or troubleshoot a circuit, or make moves in a game), the program can evaluate partial solutions, and use its teaching knowledge to respond. Typically, the declaratively stated knowledge base of diagnostic rules, causal relations, and the like is used during a tutorial to generate an "expert's solution", which, when compared to the student's behavior, provides a basis for advising the student. The combination of a knowledge base of this kind and an interpreter for applying it to particular problems constitutes an expert system, with an intelligent tutoring system having an expert system inside it (Figure 7).

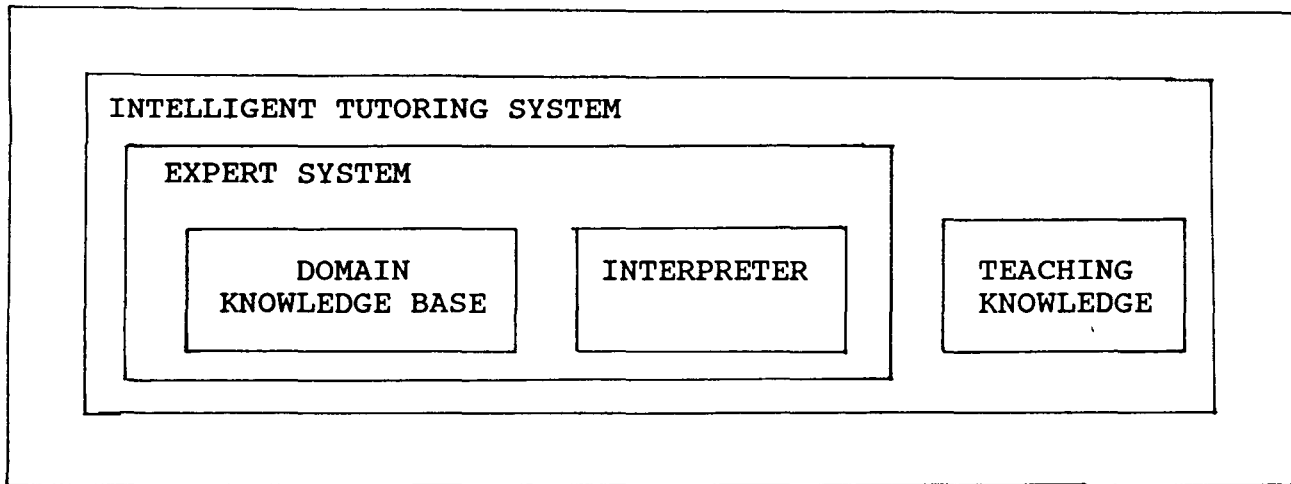


Figure 7 : Components of an Intelligent Tutoring System

CHAPTER 2.

ROTE LEARNING BY VERIFICATION

2.1 INTRODUCTION AND MOTIVATION :

When we acquire some new facts or propositions, we do not accept it immediately, we try to verify it. Verifying is done by our cognitive skills with some related facts which we are already knowledgeable about. To be more explicit the work is accomplished by one of the learning methods which have been discussed in the earlier chapter, they are learning by analogy or learning by induction. For this verification of facts there is a much simplistic scheme which we follow very often in our regular way of acquiring knowledge; that is by verification with our fellowmates or all the more better if we can get it confirmed from a source on whose knowledge we can rely.

After acquiring a fact which we are doubtful about we try to confirm it from an expert and then only digest it or set it in our memory as a fact, whose authenticity we are sure about. Otherwise we forget the doubtful fact, obliterating it from our memory, which we can say is selective forgetting. The basic nature of this learning scheme is learning by rote, but with a variation, that is verification or confirmation of the newly acquired unreliable knowledge [3]. An implementation of the learning scheme "Rote learning by verification" has been discussed in the context of a Learning Tutoring System (LEARNUTOR).

Most existing expert systems are based on knowledge obtained from a human expert. Feigenbaum, one of the pioneers of expert systems

work and head of most probably the world's largest group building such systems, puts it as follows:

The knowledge engineer works intensively with an expert to acquire domain-specific knowledge and organize it for use by a program. The expert is called upon to perform a most exacting task with which he is also unfamiliar. He must set out the sources and methodologies of his own expertise and do so in such way that it makes sense to a non-expert (knowledge engineer).

Knowledge acquisition involves problem definition, implementation, and refinement, as well as representing in a computer program the concepts, relations, procedures and problem solving strategies elicited from domain specialists. Currently the most popular method to tackle this problem is incremental approach. A scheme for knowledge acquisition is given below:

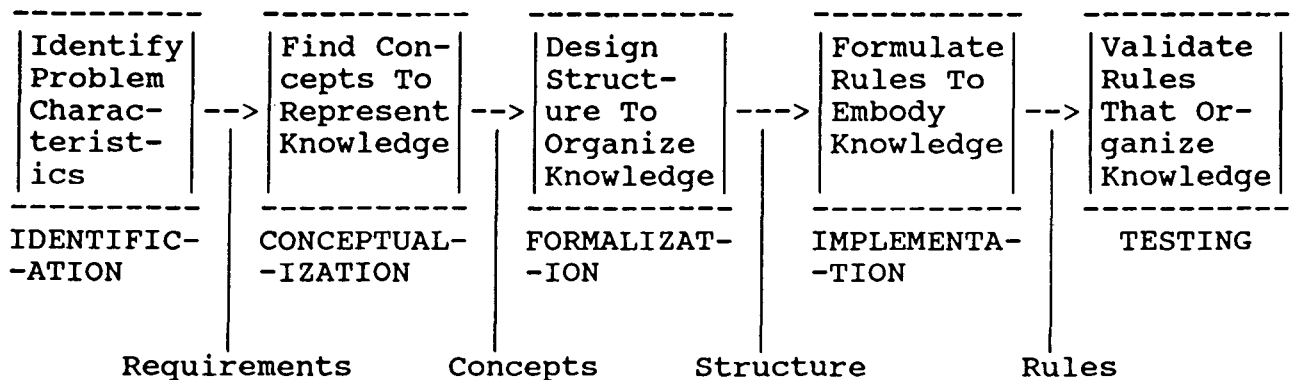


Figure 8 : Stages of Knowledge Acquisition

Acquisition and maintenance of a large knowledge base is a critical problem for knowledge-based systems. It is true that the power of an intelligent program is primarily a function of the quality and completeness of knowledge-base. Just putting the initial knowledge-base together in a suitable representation is a formidable task. However in large, open-ended problem areas, such as medicine and mathematics, the task is never-ending. Feigenbaum goes on to say that:

The acquisition of domain knowledge is the bottleneck in the building of applications-oriented intelligent agents.

Thus the system builder must be given powerful tools for keeping the knowledge base accurate and current, otherwise the system should itself have automatic capability to do the task of acquisition and maintenance according to the increased needs with time by its own learning system.

2.2 THE LEARNING TUTORING SYSTEM : LEARNUTOR

2.2.1. Architecture :

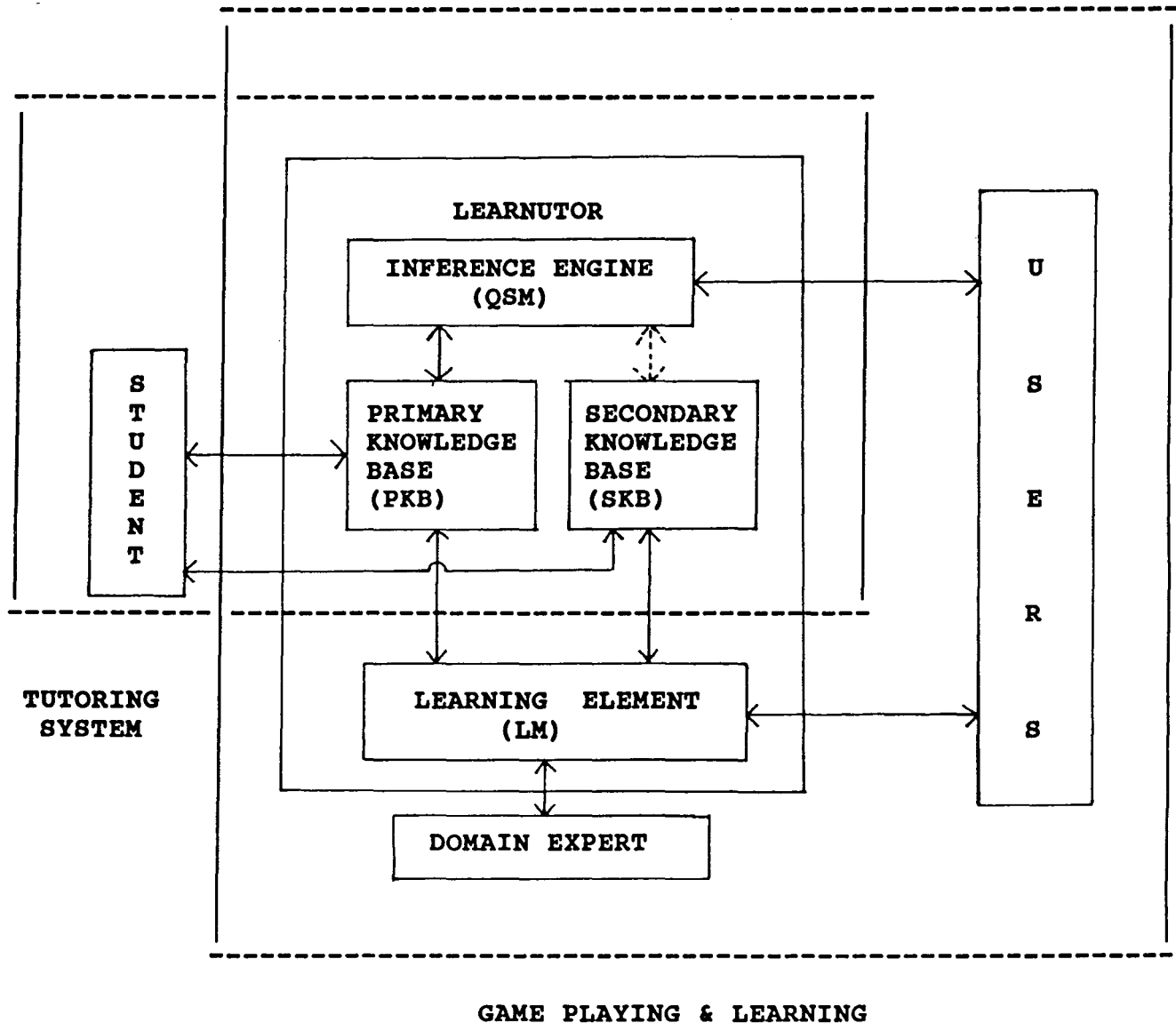


Figure 9: ARCHITECTURE OF THE LEARNING TUTORING SYSTEM : LEARNUTOR

2.2.2 What the LEARNUTOR does?

There is a game in which a person (passive player) thinks of the name of a personality of distinction and his opponent (active player) guesses the name by asking indirect questions related to the characteristics of the personality. The passive player keeps on answering the questions by yes or no. In this way the active player who is guessing the name narrows down his domain of personalities and pinpoints the personality.

For the present case, the expert system is the active player and the user the passive player. The user may have some vague and insufficient idea about some person in his mind and he wants to recollect the name and other related facts about the person. The expert system keeps on giving hints about the person's traits and the user answers according to his present knowledge, which may be incomplete or wrong. The expert system ultimately guesses the name of the personality.

2.2.3 Where is the learning element introduced?

The expert system model in consideration starts without any knowledge. Its knowledge base is empty. On being confronted with the passive player it declares that it has no knowledge regarding the person the passive player is thinking. Here the Learning element is introduced. After declaring its incompetence it tries to extract some information from the passive player regarding the personality and stores the unconfirmed knowledge in its **secondary knowledge base(SKB)**.

And in this way the secondary knowledge base is augmented. When again, restarting a new session the expert system asks the user whether he/she is knowledgeable in the domain of personalities and if so, the expert system fetches the unjustified knowledge from its secondary knowledge base gets it verified by the user (who is a domain expert), and if no incongruency is found the expert system stores the verified knowledge in its **primary knowledge base (PKB)** as a justified belief or fact. This is how the expert system learns with time and increases its expertise.

2.2.4 Parallelism with a student learning model

The present learning model in consideration can be compared with the student learning model.

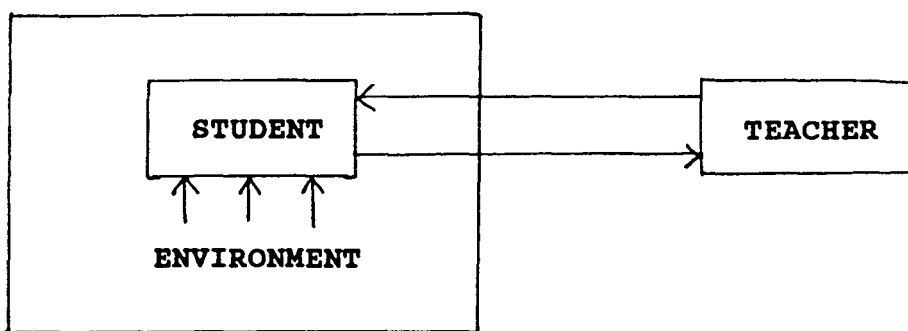


Figure 10 : Student Learning Model

A student when exposed to an environment in which he/she interacts with different persons, some of them are his/her friends, some elders. He comes across new facts, whose truth he is not sure about and thus he cannot accept them. When he meets his teacher or his



parents (who can be also considered as his teacher) upon whose knowledge he has faith, he gets his doubts cleared and unreliable knowledge confirmed and then he accepts it, by storing it permanently in his memory as a true justified belief. Thus the student learns gradually and he learns according to his requirements.

Here the expert system is exposed to an environment of users (passive players) and behaves as the student. It gains knowledge incrementally by confronting the users and by periodical interaction with the domain experts (the teacher in the student learning model) and eventually as the student qualifies to become a teacher himself, the expert system LEARNUTOR also becomes a qualified expert. But like a student its incessant urge for acquiring knowledge is unstopped. The expert system after a time though qualifying as an expert still remains a student always eager to gain knowledge. The knowledge which it gains is exactly what it requires to cater to the needs of its users. This happens as it accrues that knowledge which it lacks in and is pointed out by the users.

When a query is evaluated with respect to a knowledge base to find out about a personality, it is customary to assume that the knowledge base contains complete information about all personalities of distinction. Thus failure to find information in the knowledge base is interpreted as negative information. In this case when the information about the person in consideration is not found in the knowledge base, it would be concluded that the person is not a personality of distinction. This hidden assumption was pointed out and examined and has been labelled the closed world assumption. In general, however, this assumption is not justified and cannot be used.

For anything but idealized microworlds a knowledge base will have to be an incomplete account of domain discourse. Given this state of affairs, we want to be able first express our lack of information, and second, ask questions about it. This most obvious source of incompleteness is lack of information about the domain of discourse. This incompleteness is gradually replenished as explained in the student learning model. But the last word that the knowledge base is complete can never be said as in the real world assumption.

2.3 LEARNUTOR : Organization

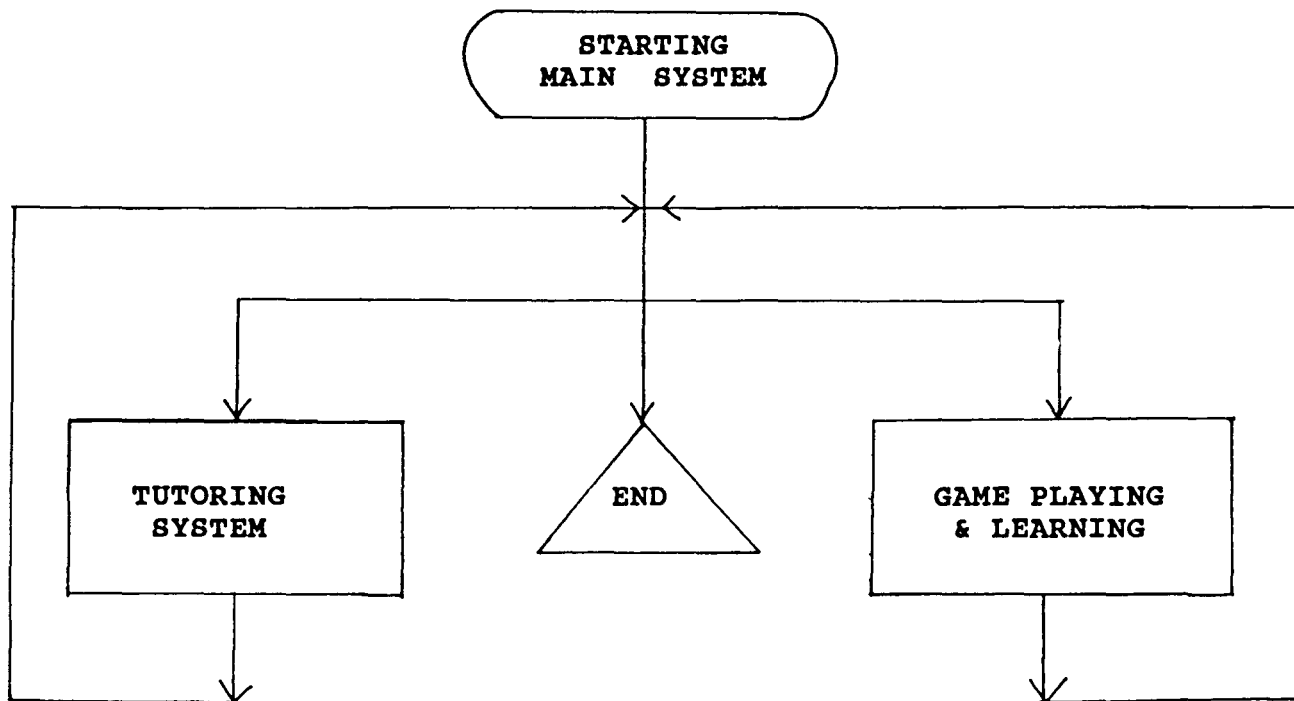


Figure 11 : LEARNUTOR -- HOW IT WORKS

2.3.1 Game Playing & Learning System : Organization

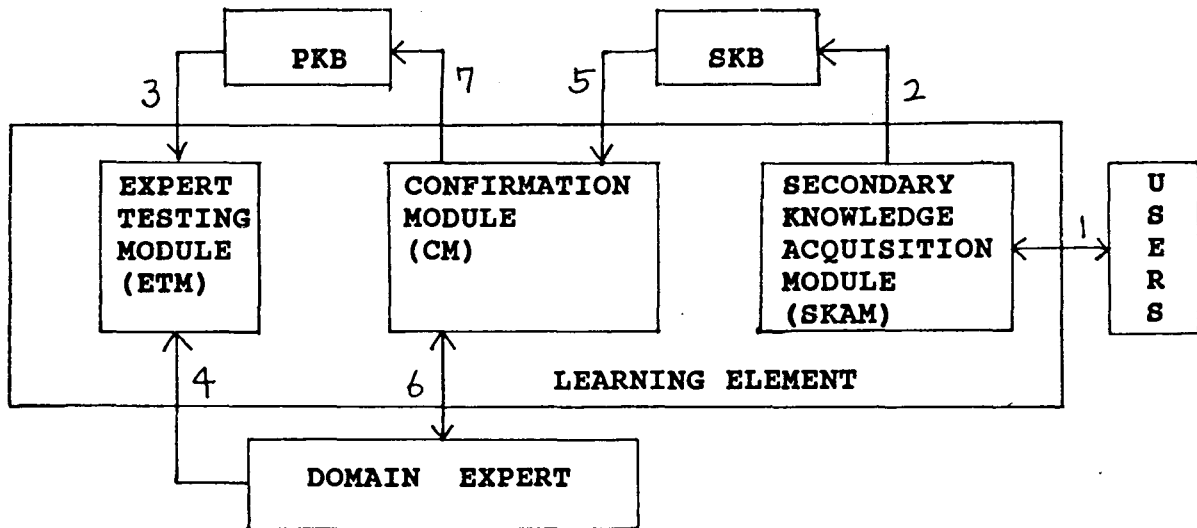


Figure 12 : Learning Element : Organization & Execution

INDEX : STEPS IN KNOWLEDGE ACQUISITION

- 1 : Unreliable knowledge acquisition from the users
- 2 : Augmentation of the unreliable knowledge in the SKB
- 3 : Knowledge access for Expert Testing from the PKB
- 4 : Testing the expertise of the Domain Expert
- 5 : Fetching of unreliable knowledge from SKB
- 6 : Verification of unreliable knowledge
- 7 : Augmentation of confirmed knowledge in the PKB

2.3.2 Game Playing & Learning System : Flow Pattern

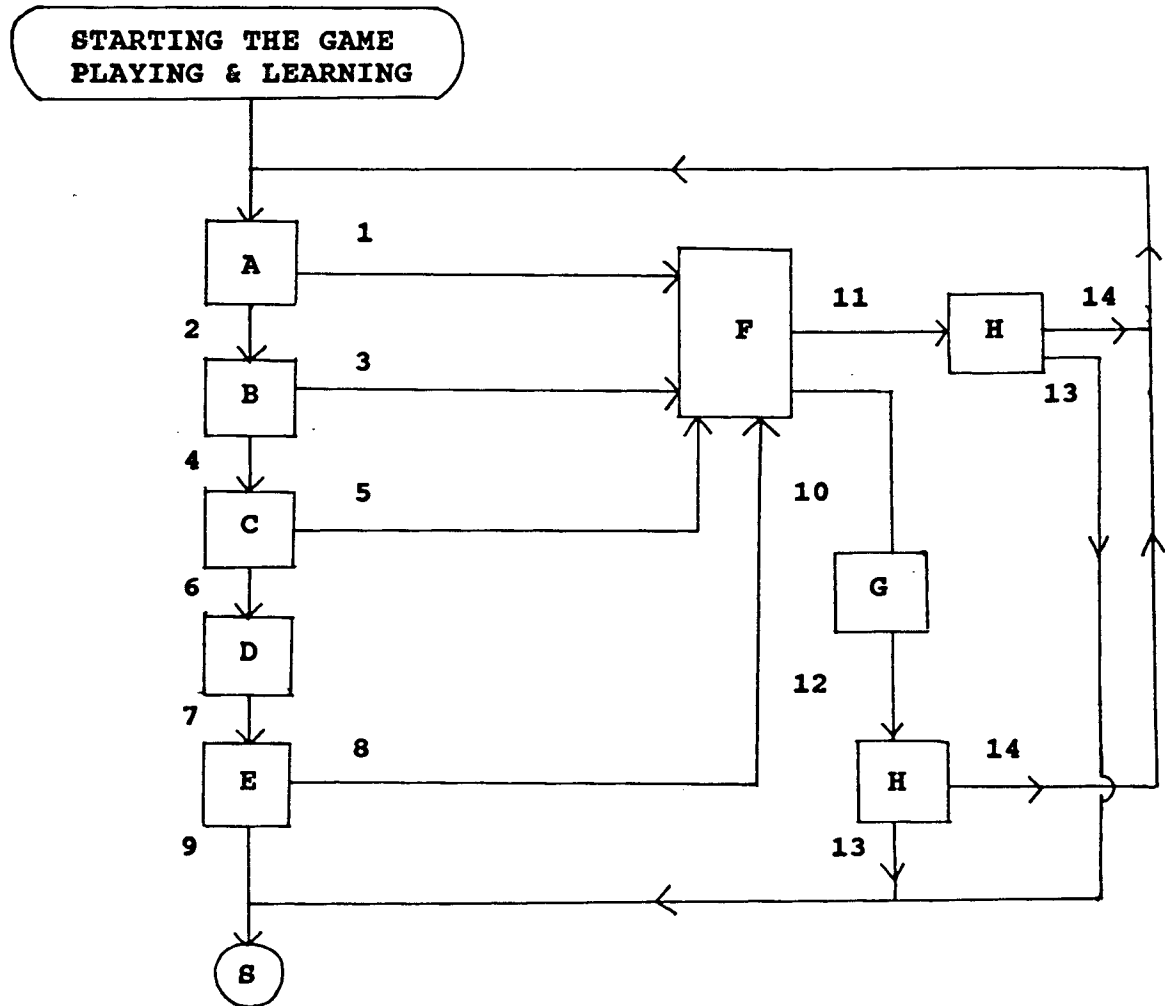


Figure 13 : GAME PLAYING & LEARNING -- HOW IT WORKS

INDEX : OF MODULES

A : SKB checking

B : Asking whether the user wants to help in confirming the unreliable knowledge in the SKB

C : EXPERT TESTING SESSION
D : CONFIRMATION SESSION
**E : Whether to start the query session or end the game playing
and learning**
F : QUERY SESSION (Personality finding : Game session)
:- INFERENCE ENGINE
G : SECONDARY KNOWLEDGE ACQUISITION SESSION
H : Restart or end the game playing and learning
S : STARTING MAIN SYSTEM

INDEX : OF EXECUTION FLOW

1 : No knowledge found in SKB
2 : Some unconfirmed knowledge found in SKB
3 : Does not want to help
4 : Would like to help
5 : Proved a non-expert
6 : Proved an expert
**7 : Confirmation done; unconfirmed knowledge verified and stored
the primary knowledge base (PKB)**
8 : Start the query session
9 : End the game playing and learning
10 : Expert fails to find out the personality
11 : Success of the expert in finding out the personality

2.3.3 Tutoring System : Flow Pattern

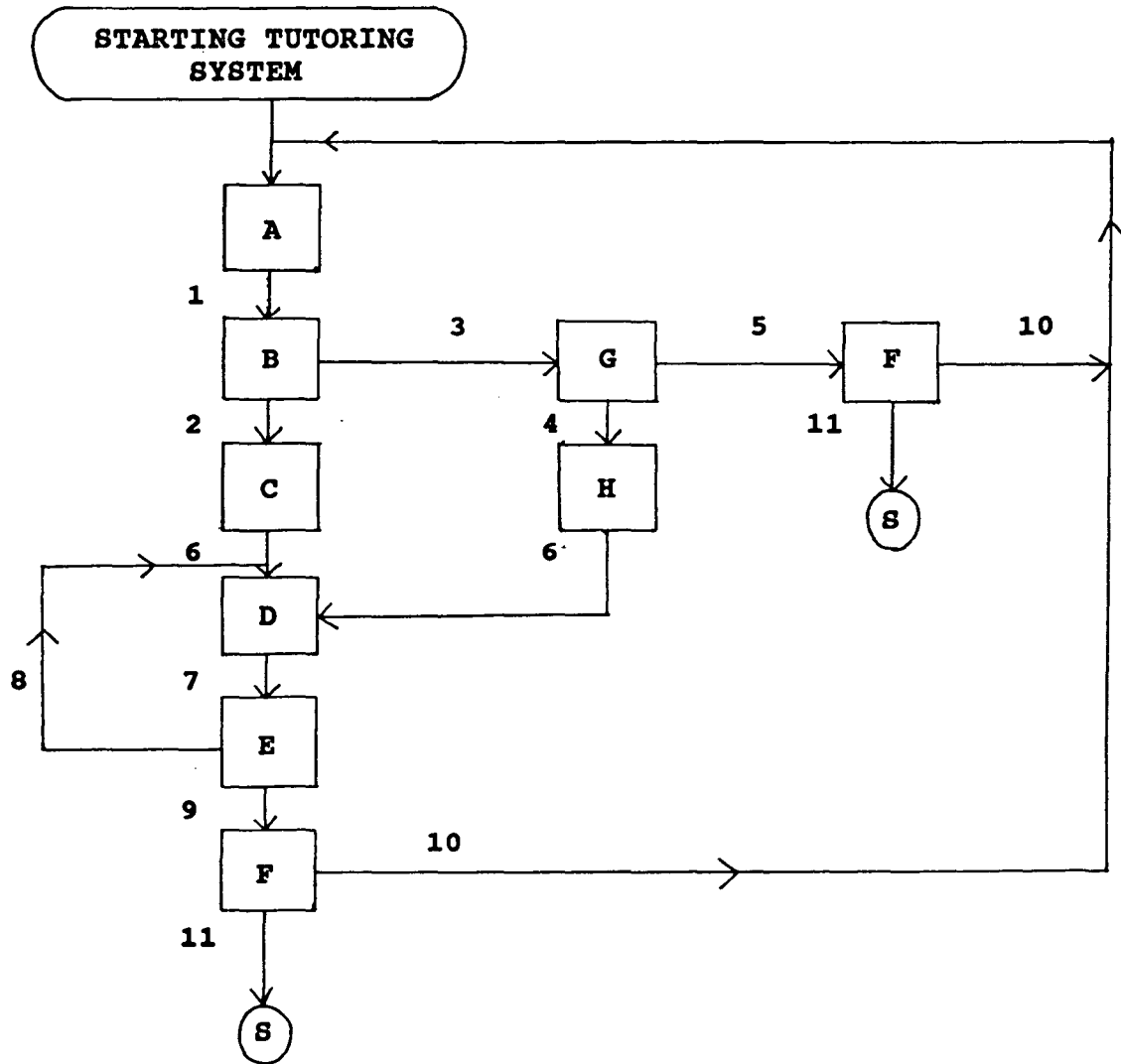


Figure 14 : TUTORING SYSTEM -- HOW IT WORKS

INDEX : OF MODULES

A : Display four options i.e., Politics, Arts & Aesthetics ,
Scholastics, Sports.

B : Display list of names for the choice made at (A) using PKB
C : Search PKB for the name choosen
D : What kind of trait do you want know ?
E : Display trait value.
 Do you want know more about him ?
F : Restart or end the tutoring session
G : Display list of names for the choice made at (A) using SKB
H : Search SKB for the name choosen
S : STARTING MAIN SYSTEM

INDEX : OF EXECUTION FLOW

1 : Choice made out of four options
2 : Name is chosen
3 : No choice of the name at (B)
4 : Name is chosen
5 : No choice of the name at (G)
6 : Name found
7 : Trait chosen
8 : Yes
9 : No
10 : Restart tutoring system
11 : End the tutoring system

CHAPTER 3.

IMPLEMENTATION

A menudriven system based on PROLOG has been implemented. Some of general features have been implemented for LEARNUTOR as discussed below:

3.1 Query-Session-Module (QSM) : The Inference Engine

The modularity of the inference engine have been maintained. The inference engine which is the QSM is an independent module and is not overlapped with other organs of the system i.e., the knowledge base. The control structure for finding out the personality as thought by the passive player(the user) is built into this module. QSM interacts with the Primary Knowledge Base (PKB) and derives the answer. The backward chaining method is implemented for QSM. All the essential features for an expert system has also been implemented for LEARNUTOR in the QSM.

The control structure of the QSM follows backward chaining scheme. When it scans the PKB and retrieves an entity, it starts with the hypothesis that the entity it has retrieved is the one to be considered. After requesting information about the attributes of the entity for confirmation or denial, it stores the knowledge acquired in a temporary knowledge base. When it starts with another entity from the PKB, the questions it has already asked regarding the attributes of the earlier entity, it does not ask for this and thus the feature of an expert system of not inquiring about the same attribute twice is

maintained for LEARNUTOR.

The temporary store of the knowledge contains all the information about the attributes for which the information has already been asked from the user, the passive player. So when retrieving another entity, falling in the earlier ones, it immediately matches the attributes of the new-fetched entity with the information about attributes already there in the temporary knowledge base. If it fails to match, the QSM immediately rejects the entity and retrieves another.

In the QSM of LEARNUTOR, the feature to report when confronted with a question, why it is asking a particular information about the attributes (which are true for the personality) after scanning the temporary knowledge base report why it is following a particular line of reasoning, is inbuilt. When confronted with the question why it is asking a particular information about the entity, it reports the attributes which are true for the personality after scanning the temporary knowledge base and also reports the name of the personality who has been currently retrieved, because all the feature attributes of the personality matches with the feature attributes stored so far in the temporary knowledge base.

Though this explanation subsystem is very primitive, but its inclusion helps to investigate the decision making process of LEARNUTOR. The user, in this case the passive player, uses it to generate confidence in the feasibility of each guess.

3.2 The Knowledge Base

The knowledge Base of LEARNUTOR has two modules. The Primary Knowledge base (PKB) and the Secondary Knowledge Base (SKB).

3.2.1 The Primary Knowledge Base (PKB)

The PKB contains the justified and confirmed facts, which LEARNUTOR acquires on confirmation from the Domain Expert. The knowledge representation scheme followed for PKB is indexed sequential list structure and thus the access to any entity in the PKB can be both random and sequential. The entities in the SKB are of Object-Attributes format i.e.,

```
person("Name",[attr("sex","male"),attr("mortalstatus","dead"),
attr("country","India"),attr("broad field of distinction",
"politics"),attr("specific field of distinction","international
politics"),attr("date/year of birth","02/10/1869"),attr("date/yea
of death","1948"),attr("Other Traits","Specification)...])
```

the object being the name of the personality and the attributes is a list of trait-specification pairs.

The PKB is an independent module, and it can be accessed by any other module of LEARNUTOR. The domain independency is also established. The PKB can at the users need be replaced by another PKB. Suppose the first PKB consists of personalities of distinction of personalities of distinction of European origin and so forth. Thus a knowledge base module can be "plugged" and "unplugged" with the users needs.

3.2.2 The Secondary Knowledge Base (SKB)

The SKB contains the unconfirmed facts which are extracted from

the passive player (the user) during the Secondary Knowledge Acquisition Session by the Secondary Knowledge Acquisition Module (SKAM)

The independency feature and the knowledge representation structure is similar to the PKB. The only difference being that the SKB entities which are the queries/unconfirmed knowledge stored in the object-attributes pair format can have the object or some of the attributes-pair's specification part doubtful; as, when extracting knowledge from the users; the user's knowledge could have been vague and thus they may have answered with a doubt in their mind or may not have answered at all to the queries of the SKAM, thus filling in the trait-specification part by "?". The object can be filled up by "?", i.e.,

```
person("?", [attr("sex", "?"), attr("mortal status", "alive"), ...])
```

3.2.3 Learning Module (LM)

The learning module is the special feature for this expert system which endows the name LEARNUTOR (LEARNING TUTORING SYSTEM) to the expert system implemented. The LM is independent of the other subsystems of LEARNUTOR and it interacts independently with the SKB and PKB, and the users and domain expert.

The overall modularity of the LM is maintained and the submodules of LM i.e., Expert Testing Module (ETM), SKAM, & CONFIRMATION MODULE (CM) are also inter-modular in nature, such that they can be used for other purpose where and when necessary.

3.3 HOW IT ALL WORKS : IMPLEMENTATION DETAILS

Now that we have the necessary background about the LEARNUTOR functionings, we can delve into the actual implementation details. At the implementation level I felt that Prolog being a rule-based language would be the most efficient implementation tool for building the expert system. Moreover the very powerful pattern-matching facilities of the Prolog helps in the implementation of the expert system where lot of matching is done. In the present case Turbo-Prolog is used.

3.3.1 Structuring the Knowledge Base

Temporary knowledge base : The first step to creating an expert system is to define the structure of the dynamic or temporary knowledge base. The built-in dynamic database facility of Turbo-Prolog is used. The knowledge base is

```
        person(string,list2)
        /* name of the personality, attributes */
```

where list2 will be defined to be list of trait-specification pairs.

The QSM must keep track of all attributes that belong to the goal, those that do not and the ones for which the goal is not reached and thus have to be stored as a query in the secondary knowledge base. Hence the entire database declaration of expert system is

```
database
  person(string,list2)
    /* name,attributes */
  yes(attribute),      /* trail of yes response */
  no(attribute),       /* trail of no response */
```

```
q(attribute),          /* trail of yes response but
                        for which the goal is not reached */
```

where attribute will be defined to be a functor

```
attr(string,string)
    /* trait, specification */
```

The primary knowledge base and the secondary base are created as appendable and entity retrievable files. They are kept in the disk as "PKB.DAT" (primary knowledge base file) and "SKB.DAT" (secondary knowledge base file).

3.3.2 Constructing the QSM : Inference Engine

The Inference Engine is the driving force of an expert system. The one which is developed here is though very simple is quite efficient. The top level predicate of the QSM is query.

Despite its simple appearance query is quite sophisticated because its operation is built on Turbo-Prolog's backtracking capabilities. It works as follows. First, a body predicate of query **pdbread** retrieves an entity from PKB and finds the entity's object and attributes parts to Name and Attributes. Assuming that there is something in the PKB this step will succeed. Next the routines **trailyes** and **trailno** screen out personalities that do not meet the current state of the system. **Trailyes** checks Name's attributes' list to make sure that it contains all the attributes that the user has told the system the personality must have; and **trailno** checks to make sure that Name's attributes' list do not contain any attribute pair that has been rejected. The predicate **try** asks the user(passive

player) about the attributes. If they all match then the personality has been found. The predicate **clearfacts** then clear the temporary databases. If they do not match **try** fails and backtracking occurs with a new entity retrieved from the PKB. Finally if all the objects in the PKB have been exhausted then the QSM passes on the flow to the Learning Module's SKAM.

The second clause of query is provide enter the Learning Module and eventually to the SKAM through the predicate **query-capturing** The third clause of the query is used to restart the whole session and the fourth clause is to exit from the session on failing the other query clauses.

The **try** portion uses the support predicate **process**, which takes different actions based on user's response. **process** also explains why a certain question is being asked. **process** responds to the command why by printing out what the personality it is currently pursuing and the current list of attributes.

If one is reviewing the knowledge base, one will find that the attributes of two personalities can be identical, expert that one has some additional attribute. Therefore even if the user, really has a personality in mind who is placed below in the PKB but where attributes cover all the features of a personality who is placed in earlier part of PKB. The expert system as built will always report the firstly placed personality. This may not be acceptable. This list holds names of the personalities that the system has already examined, and it will prevent the knowledge from being reexamined from the top when the user is not satisfied with the first answer.

3.3.3 Building the Learning Module (LM)

The LM comprises the Expert Testing Module (ETM), the Secondary Knowledge Acquisition Module (SKAM) and the Confirmation Module (CM). These modules are independent of each other and thus maintains the inter-modularity of the system.

3.3.3.1 ETM Construction

The primary predicate of this module is **expert-testing-session**. The number of entities stored in the PKB is maintained in the disk file RECORD.DAT. The first step in this module is retrieval of the information regarding the number of entities and then passing on this as a parameter LIMIT to the predicate test. This predicate randomly access PKB and retrieves one of the entities and passes on the entity as a parameter to the predicate in its body question for posing questions to the user to verify the later's expertise in the domain of personalities.

3.3.3.2 SKAM Construct

The access to the SKAM is from the QSM. The top level predicate here is query-capturing, which is having in the body the predicates find-sex, find-mortal status, and other attribute extracting predicates; and the predicate find-name. This module is thus ment to extract unjustified knowledge from the users (the passive players) to be stored in the SKB.

3.3.3.3 CM Construct

In this confirmation session the top level active predicate is go-confirm. In its body the predicates present acts as follow;the first predicate sdbread retrieves an entity from the SKB and passes on the entity as a parameter to the predicate attribute-verification. The second clause of attribute-verification verifies the attribute specifications from a domain expert and the first clause appends the confirmed knowledge by pdbassert into the PKB after extracting some more traits from the domain expert by the predicate find-more-traits.

The go-confirm works in a loop until all the unconfirmed/unsure knowledge in the SKB is either verified and placed in the PKB; or discarded which is selective-forgetting.

3.4 MENUS

LEARNUTOR is a menu driven system. The main program menu & the tutoring system menu are implemented as given below :

```
/* MAIN PROGRAM MENU */
```

```
menu1 :-  
  clearwindow,  
  makewindow(1,11,71,"MAIN MENU",0,0,25,80),  
  field_str(5,31,20,"PLAYING GAME"),  
  scr_attr(5,31,15),  
  field_str(10,31,20,"TUTORING SYSTEM"),  
  scr_attr(10,31,15),  
  field_str(15,31,20,"EXIT TO SYSTEM"),  
  scr_attr(15,31,15),  
  field_str(20,11,56,"PLEASE SELECT YOUR CHOICE BY FIRST HIGHLIGHTED LETTER"),  
  field_attr(20,11,56,71),  
  readchar(Reply),  
  char_int(Reply,Value),
```

```

    selct(Value).

selct(80) :-
    clearwindow,
    playing_game.

selct(84) :-
    clearwindow,
    tutoring.

selct(69) :-
    clearwindow,
    field_attr(11,35,9,180),
    field_str(11,35,9,"THANK YOU"),
    field_attr(12,30,22,180),
    field_str(12,30,22,"YOU ARE ALWAYS WELCOME").

selct(_) :-
    clearwindow,
    field_attr(11,35,9,180),
    field_str(11,35,9,"THANK YOU"),
    field_attr(12,30,22,180),
    field_str(12,30,22,"YOU ARE ALWAYS WELCOME").

/*      TUTORING SESSION  MENU      */

menu2 :-
    makewindow(1,11,71," ABOUT WHAT KIND OF PEOPLE DO YOU WANT TO KNOW ? ",0,0,25,
    field_str(2,28,20,"POLITICS"),
    scr_attr(2,28,15),
    field_str(7,28,20,"ARTS AND AESTHETICS"),
    scr_attr(7,28,15),
    field_str(12,28,20,"SCHOLASTICS"),
    scr_attr(12,28,15),
    field_str(17,28,20,"SPORTS_MAN"),
    scr_attr(17,30,15),
    field_str(21,11,56," PLEASE SELECT YOUR CHOICE BY FIRST HIGHLIGHTED LETTER "),
    field_attr(21,11,56,71).
    readdevice(keyboard),
    readchar(Reply),
    char_int(Reply,Value),
    select(Value).

select(80) :-
    clearwindow,
    makewindow(3,11,71," ***  DIALOGUE  WINDOW  *** ",0,0,25,80),
    name("politics").

```

```
select(65) :-
  clearwindow,
  makewindow(3,11,71," *** DIALOGUE WINDOW *** ",0,0,25,80),
  name("arts and aesthetics").
```

```
select(83) :-
  clearwindow,
  makewindow(3,11,71," *** DIALOGUE WINDOW *** ",0,0,25,80),
  name("scholastics").
```

```
select(79) :-
  clearwindow,
  makewindow(3,11,71," *** DIALOGUE WINDOW *** ",0,0,25,80),
  name("sports_man").
```

3.5 Game Playing & Learning Session

3.5.1 Session1 : (Query Session)

Would you like to start the query session ? (y/n) Y

< starting query session >

Person's sex : -----male ? (y/n/d/why) : Y

Person's dead/alive : --dead ? (y/n/d/why) : WHY

I think the person may be

Monhandas Karamchand Gandhi because the person has
the following attributes :

sex : male

dead/alive-----dead ? (y/n/d) : N

Person's dead/alive : --alive ? (y/n/d/why) Y

Person's country : -----India ? (y/n/d/why) N

Sorry, I am not aware of the person you are thinking of.

Would you tell me something regarding the person ? (y/n) Y

Which country is the person from ? (country name /?) KOREA

What is the person's broad field of distinction ?

e.g., Politics / Art & Aesthetics / Scholastics / Sports
POLITICS

What is the person's specific field of distinction ?

e.g., Politics ----- (national/international)
Arts & Aesthetics ---- (fine arts/drama/film/mucial/etc.)
Scholastics ----- (writer/poet/education/etc.)
Sports ----- (tennis/volley ball/foot ball/etc.)

NATIONAL

What is the date/year of birth of the person ? : (dd/mm/yyyy)

03/11/1930

What is the trait ?

READING

What is the trait specification ?

READING NOVEL

Any other traits you know about the person ? (y/n) :

N

Are you aware of the the name of the person ?

If known tell me and if not known type '?'

YOUNG SAM KIM

Want one more round ? (y/n) :

N

Press the SPACE bar

**Thank you
You are always welcome**

**3.5.2 Session 2 : (Correct Expert Code and
Confirms SKB Knowledge)**

Are you an expert in the field of personalities ? (y/n)

Y

Would you like to help me in clearing some of my doubt ? (y/n)

Y

Would you mind to prove yourself as an expert ? (y/n)

N

Please enter your Expert Code :

SHIN

Now I am confirmed that you are an expert.

Sorry for the inconvenience, but I had to be sure.

Should we start off with the confirmation of some of the
questions ? (y/n)

Y

Lee Sun Sin - sex : male ? :: (y/n) Y
 Lee Sun Sin - dead/alive : dead ? :: (y/n) Y
 Lee Sun Sin - country : korea ? :: (y/n) Y
 Lee Sun Sin - field of work : scholastics ? :: (y/n) N
 Specify character :: field of work POLITICS
 Lee Sun Sin - specific field of work : national ? :: (y/n) Y
 Lee Sun Sin - data of birth : 11/06/1601 :: (y/n) Y
 Lee Sun Sin - data of death : 07/08/1670 :: (y/n) Y
 Any other traits you know about the person ? (y/n) N
 .
 .
 .
 < It will confirm about all person in SKB continuously >
 Would you like to start the query session ? (y/n) Y
 < Starting query session >

3.5.3 Session 3 : (Expert Code not correct and the system tests expertise of the user)

Are you an expert in the field of personalities ? (y/n) Y
 Would you like to help me in clearing some of my doubt ? (y/n) Y
 Would you mind to prove yourself as an expert ? (y/n) N
 Please enter your Expert Code : any code
 Sorry your code is incorrect.
 You have to prove your expertise.
 Would mind if I test your expertise ? (y/n) N
 < Expert testing session : System asks some questions
 randomly chosen from the PKB >
 Mohandas Karamchand Gandhi --- country ? INDIA
 You are correct.

Try the next question.

Young Sam kim ----- field of work ?

POLITICS

You are correct.

Try the next question.

Rajiv Gandhi ----- sex ?

MALE

You are correct.

Try the next question.

Aristotle ----- dead/alive ?

DEAD

You are correct.

Try the next question.

Abul Fazal ----- date of birth ?

1551

You are correct.

Try the next question.

< If all answer is correct : >

Now I am confirmed that you are an expert.

Sorry for the inconvenience, but I had to be sure.

Should we start off with the confirmation of some of the
questions ? (y/n)

Y

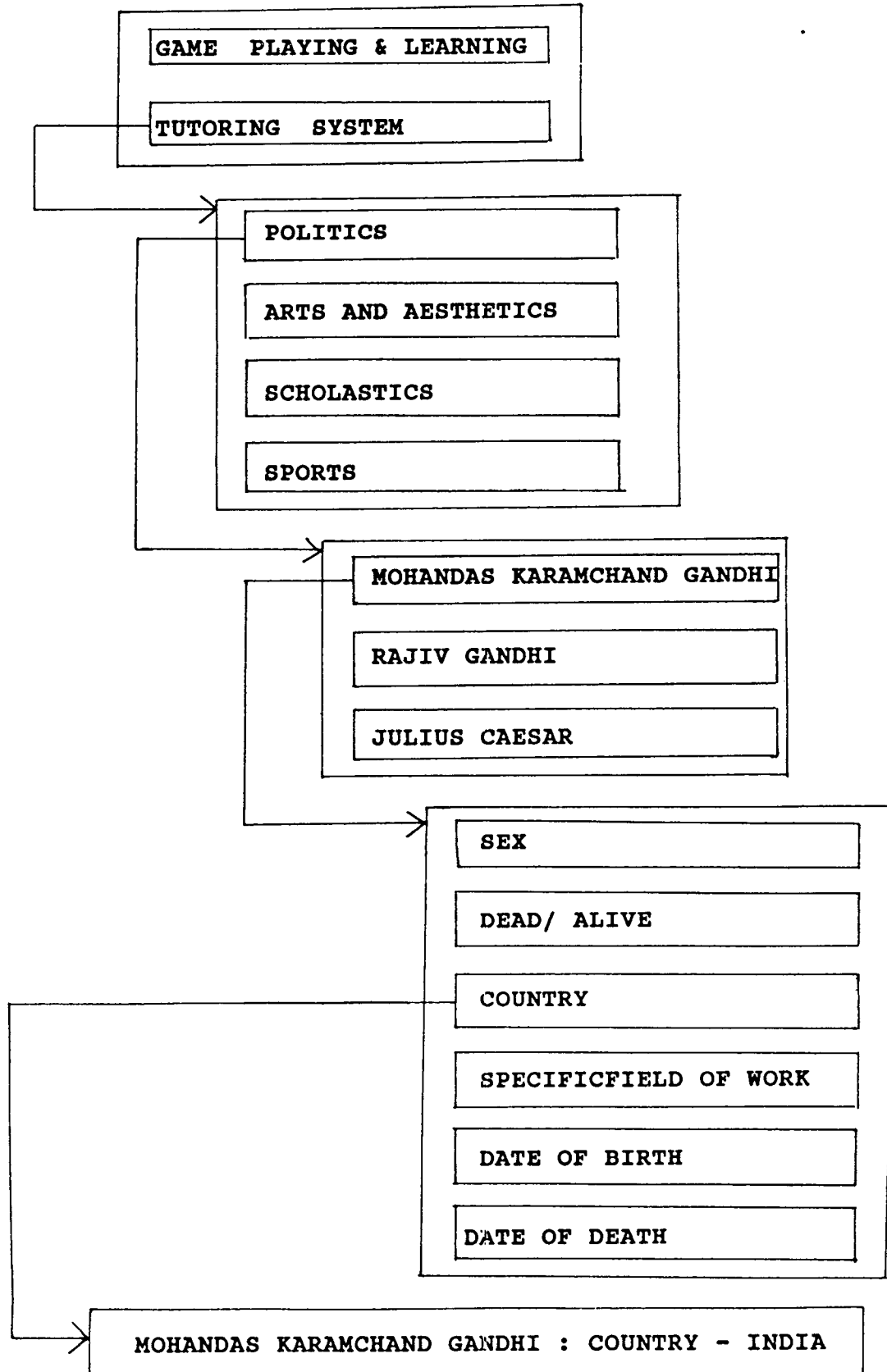
< go to Confirmation session >

< If any answer is not correct : >

You are wrong.

I am sorry, you would not be of much help to me.

3.6 Tutoring Session



CHAPTER 4.

CONCLUSION

A PROLOG based system LEARNUTOR (LEARNing tUTOR) is developed as a prototype implementation of the learning scheme - Rote Learning by Verification. The system operates in two phases, namely Game Playing & Learning and Tutoring System. The first phase of the system makes the system capable of acquiring knowledge in a particular domain of discourse and eventually qualifies as an expert. During the second phase of the system, Tutoring System, it is shown how the proposed system LEARNUTOR can be viewed as a tool in education. A limitation of LEARNUTOR is that it cannot learn rules. But if this ability can be implanted in the system, then the system would answer just not to some fact specific queries, but would also answer to queries which would infer on those facts. These can be implemented by one of the established learning methods learning by being told or learning by induction. One of the important future direction of research would be to incorporate into the system the concept of the "Learning from Imperfect Data" in general and "Learning from an Imperfect Teacher" in particular within the Tutoring System Phase.

REFERENCES

- [1] ASHBY, W. ROSS, 1976, DESIGN FOR BRAIN, THE ORIGIN OF ADAPTIVE BEHAVIOR, JOHN WILEY AND SONS, INC..
- [2] BLOCK, H. D., 1975, "THE PERCEPTRON: A MODEL OF BRAIN FUNCTIONING, I", REV. MATH. PHYSICS, VOL.34, NO.1, pp. 125-155,
- [3] BHARADWAJ K. K., GHOSH SURYANIL "LEAREX : A LEARNING EXPERT SYSTEM" (UNDER PREPARATION).
- [4] BRATKO, I. & MULEC, P., 1981, "AN EXPERIMENT IN AUTOMATIC LEARNING OF DIAGNOSTIC RULES", INFORMATIKA.
- [5] BROWN, J. S. BURTON, R. R., & BELL, A. G., 1974, SOPHIE: A SOPHISTICATED INSTRUCTIONAL ENVIRONMENT FOR TEACHING ELECTRONIC TROUBLESHOOTING. BBN REPORT NO. 2790.
- [6] BUCHANAN, B. G. & MITCHELL, T. M., 1978, "MODEL-DIRECTED LEARNING OF PRODUCTION RULES", PATTERN-DIRECTED INFERENCE SYSTEMS, WATERMAN, D. A. & HAYES-ROTH, F (EDS.), ACADEMIC PRESS, NEW YORK.
- [7] BURTON, R. R., & BROWN, J. S. , 1979, AN INVESTIGATION OF COMPUTER COACHING FOR INFORMAL LEARNING ACTIVITIES. INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, 11, 5-24.
- [8] CARBONELL, J. G., MICHALSKI, R. S., & MITCHELL, T. M., 1983, "AN OVERVIEW OF MACHINE LEARNING", MACHINE LEARNING, MICHALSKI, R. S., CARBONELL, J. G., AND MITCHELL, T. M. (EDS.), TIOGA.
- [9] COHEN, P. R. & FEIGENBAUM, E. A. (EDS.), 1982, THE HANDBOOK OF ARTIFICIAL INTELLIGENCE, KAUFMAN, LOS ALTOS, CA., VOL. III.
- [10] COLLINS, A., 1976, PROCESSES IN ACQUIRING KNOWLEDGE. IN R. C. ANDERSON, R. J. SPIRO, & W. E. MONTAGUE (EDS), SCHOOLING AND THE ACQUISITION OF KNOWLEDGE. HILLSDALE, NJ: LAWRENCE ERLBAUM ASSOCIATES.
- [11] DAN W. PATTERSON, 1990, INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEM, PRENTICE HALL, INC.
- [12] EUGENE CHARNIAK & DREW MCDERMOTT, 1985, INTRODUCTION TO ARTIFICIAL INTELLIGENCE, ADDISON-WESLEY.
- [13] MICHALSKI, R. S., CARBONELL, J. G., & MITCHELL, T. M. (EDS.), 1983, MACHINE LEARNING, AN ARTIFICIAL INTELLIGENCE APPROACH, TIOGA.
- [14] SCHILDT, H., 1987, "MACHINE LEARNING", ADVANCED TURBO PROLOG, OSBORNE MCGRAWHILL, pp. 199-221.

- [15] SLEEMAN, D., & BROWN, J.S., 1981, INTELLIGENT TUTORING SYSTEMS. LONDON: ACADEMIC PRESS.
- [16] WINOGRAD, T., 1975, FRAME REPRESENTATIONS AND THE DAELARATIVE / PROCEDURAL CONTROVERSY. IN D. G. BOBROW & A. COLLINS(EDS), REPRESENTATION AND UNDERSTANDING. NEW YORK: ACADEMIC PRESS.
- [17] WINSTON, P. H., 1980, LEARNING AND REASONING BY ANALOGY. COMMUNICATIONS OF THE ACM 23,12(DEC): 689-702.
