



Knowledge Representation Schemes and Comparative Schemes for Some Real Applications

**By:
Marwan H. Algeblawi**

Supervisor: Dr. Abdulhamed M. Abdulkafi

**A thesis submitted to the Department of Computer Science
In partial fulfillment of the requirements for the degree of
Master of Science**

**Al-Tahaddi University
Faculty of Science
Department of Computer science
Sirite, G. S. P. L. A. J.**

Academic year 2005/2006



التاريخ :

الموافق : 22 / 1 / 2008

الرقم الاشاري (الرجوع اليه) : 1000

Faculty of Science

Department of Computer science

Title of Thesis

**((Knowledge Representation Schemes and Comparative
Schemes for Some Real Applications))**

By
Marwan Hassan Algeblawi

Approved by: 90

Dr. Abdulhamed Mohamed Abdulkafi
(Supervisor)

Dr. Nage E. Bazina
(External examiner)

Dr. Idris S. El-Feghi
(Internal examiner)

Countersigned by:
Dr. Mohamed Ali Salem
(Dean of Faculty of Science)



Acknowledgements

There are many people to whom I owe gratitude for their friendship, encouragement and support in completing this thesis. The following people deserve special mention.

My teachers during master courses. They helped round out my background in Artificial Intelligence, database, networks and another subjects.

I would like to thank the staff members of department of computer science and the members of faculty of science at the University of Alrahaddi where I spent my early years.

I owe a special debt of thanks to my wife and my children's.

Finally, my friends special thanks are: Ammar Abani for his support.

Marwan H. Algeblawi

Abstract

Knowledge Representation (K.R.) techniques and inference have been the backbone of Artificial Intelligence(A.I). KR is a substantial sub field in its own right, on the borderline between AI and cognitive science. It is concerned with the way in which information might be stored in the human brain. The main criteria for assessing a representation of knowledge is logical adequacy, heuristic power and notational convenience. Logical adequacy means that the representation should be capable of making all the distinctions that you want to make. Heuristic power means that, as well as having an expressive representation language, there must be some way of using representations so constructed and interpreted to solve problems. KR is one of the most widely used methodologies for representing knowledge and analysis in AI. However, analysis and implementing of any problem solution without KR can be a time-consuming and error-prone task. Furthermore, analysis and implementing of any problem solution without chosen the best KR kind can be a time-consuming and error-prone.

There is no single theory to explain human knowledge organization or a best technique for structuring data in a conventional computer program, no single KR structure is ideal. One of the important responsibilities as a knowledge engineer is to choose the KR technique best suited for the given application.

This thesis compares between some kinds of KR such as Rule, Logic, Semantic Network, Frames, Fuzzy Logic, Neural Network and Genetic Algorithms representation for some real applications depending on theoretical perspective and practical perspective using some criteria such as variables, statement/relations, programming and reasoning technique. The main purpose of this thesis is to select the best KR to be used for solve given problem.

Keywords: Rule, Logic, Semantic Network, Frames, Fuzzy Logic, Neural Network and Genetic Algorithms representation.

Table of contents

List of figures	viii
List of tables	x
List of abbreviations	xi
Chapter One: Introduction	1
1.1 Knowledge	2
1.1.1 Definitions	2
1.2 Types of knowledge	2
1.3 Components of Knowledge	4
1.4 Knowledge Representation	6
1.4.1 Definitions	6
1.4.2 Characteristic of knowledge representation	6
1.4.3 Limitations of knowledge representation	6
1.4.4 Main type for knowledge representation	6
1.4.5 Advantages of Knowledge Representation	7
1.5 Related work in knowledge representation	8
1.6 Thesis goals	10
1.7 Thesis contribution	11
1.8 Thesis organization	11
Chapter Two: Rule representation	13
2.1 Definitions	13
2.2 Rule-based systems model	14
2.3 Structure of rule based system	15
2.4 Rule based system architecture	17
2.5 Representation knowledge in rule based system	18

2.6	Types of rules	20
2.7	Variable of rules	21
2.8	Rule sets	22
2.9	Advantages of rules	24
2.10	Disadvantages of rules	25
Chapter Three: Logic representation		26
3.1	Definitions	26
3.2	Propositional logic	26
3.3	Predicate logic	29
3.4	First order logic	30
3.5	Second order logic	30
3.6	Higher order logic	30
3.7	Difference between kinds of logic	32
3.8	Advantages of logic	33
3.9	Disadvantages of logic	33
Chapter Four: Semantic networks representation		34
4.1	Definitions	34
4.2	Expanding semantic networks	35
4.3	Inheritance in semantic networks	37
4.4	Default and inheritance in semantic networks	37
4.5	Exception handling in semantic networks	38
4.6	Advantages of semantic networks	40
4.7	Disadvantages of semantic networks	40
Chapter Five: Knowledge representation with frame		42
5.1	Definitions	42
5.2	Basic frame design	42

5.3 Class frame	44
5.4 Instance frame	45
5.5 Frame inheritance	46
5.6 Inheritance behavior	46
5.7 Hierarchical structure	47
5.8 Facets	48
5.9 Advantages of Frame	48
5.10 Disadvantages of frame	49
Chapter Six: Fuzzy logic representation	50
6.1 Definitions	50
6.2 Fuzzy variables(linguistic variable)	51
6.3 Fuzzy sets	53
6.4 Forming fuzzy sets	54
6.5 Fuzzy sets representation	55
6.6 Fuzzy sets operations	56
6.7 Fuzzy inference	58
6.8 Advantages of fuzzy logic	58
6.9 Disadvantages of fuzzy logic	59
Chapter Seven: Neural network representation	61
7.1 Definitions	61
7.2 Neural network components	61
7.3 Processing unit	62
7.4 Network topologies	62
7.5 Representation capability	63
7.6 Network structure design	63
7.7 Types of variables	63

7.8 Knowledge representation and neural network	64
7.9 Advantages of neural networks	67
7.10 Disadvantages of neural networks	68
Chapter Eight: Genetic algorithms representation	69
8.1 Definitions	69
8.2 Elements of genetic algorithms	69
8.3 Genetic algorithms Operators	71
8.4 Advantages of genetic algorithms	75
8.5 Disadvantages of genetic algorithms	76
Chapter Nine: Comparison of different representation	77
9.1 Criteria of comparison	77
9.2 Conclusions	80
References	83
Appendix I Computer diagnostic by rules	xii
Appendix II Computer startup by logic	xvi
Appendix III Elephant's inheritance by semantic networks	xx
Appendix IV Car registration in traffic office by KR with frames	xxi
Appendix V Car type determination by Fuzzy logic	xxiii
Appendix VI Digit Recognition by neural network	xxvi
Appendix VII Resolve equation by genetic algorithms	xxx

List of figures

Figure		Page
1.1	Component of knowledge	5
1.2	Relation between two type of knowledge representation	7
2.1	Rule-based model	15
2.2	Basic structure of a rule-based system	16
2.3	Rule-based system operation	17
2.4	Rule-based system architecture	18
2.5	Computer diagnostic rule sets	23
4.1	Semantic network of bird	35
4.2	Expanded semantic network of bird	36
4.3	Default and inheritance elephant	37
4.4	Semantic network operation	39
5.1	Report card form	43
5.2	General frame structure	44
5.3	Bird frame	45
5.4	Parrot frame	46
5.5	Frame world of birds	47
6.1	Membership function for fuzzy set "tall"	51
6.2	Membership function for fuzzy sets "short", "medium", "tall"	51
6.3	Fuzzy and sets of "young" people	54
6.4	Fuzzy sets on height	55
7.1	Neural network components	61
7.2	Processing unit	62
7.3	Feed-forward neural network	62

7.4	Recurrent neural network	63
7.5	Combined use of a receptive field and weight-sharing.	66
7.6	Block diagram of an invariant feature-space type of system	67
8.1	Basic genetic algorithm	71
8.2	A chromosome for the scheduling problem	74

List of tables

Table		Page
1.1	Some definitions of artificial intelligence	2
1.2	Different types of knowledge	3
2.1	Car diagnostic condition rules	19
3.1	Logic operators and symbols	26
6.1	Examples of linguistic variables with typical values	51
8.1	Power units and their maintenance requirements	73
8.2	Gene by Four bits	74
9.1	Criteria of comparison for rules representation	78
9.2	Criteria of comparison for logic representation	78
9.3	Criteria of comparison for semantic networks representation	79
9.4	Criteria of comparison for knowledge representation with frames	79
9.5	Criteria of comparison for fuzzy logic representation	79
9.6	Criteria of comparison for neural network representation	80
9.7	Criteria of comparison for genetic algorithm representation	80

List of abbreviations

AI	Artificial intelligence
KR	Knowledge representation
XP	Experience professional
IBM	International business machine
GB	Gega byte
RAM	Random access memory
MB	Mega byte
VGA	Video graphic accumulator
O-A-V	Object attribute value
XOR	Exclusive OR
GA	Genetic algorithms
MW	Megahertz watt

Chapter One

Introduction

Artificial Intelligence(A.I.) is one of the newest sciences. Work started in earnest soon after World War II, and the name itself was coined in 1956. AI currently encompasses a huge variety of sub fields ranging from general purpose areas, such as learning and perception to such specific tasks as playing chess, proving mathematical theory, writing poetry, and diagnosing diseases. "Definitions of AI according to eight textbooks are shown in table-1.1¹, these definitions vary along two main dimensions. Roughly, the ones on top are concerned with thought processes and reasoning, whereas the ones on the bottom address behavior. The definitions on the left measure success in terms of fidelity to human performance, whereas the ones on the right measure against an ideal concept of intelligence, which we will call rationality. A system is rational if it does the "right thing" given what it knows." [1].

Knowledge Representation (K.R.) techniques and inference have been the backbone of AI. KR is a substantial sub field in its own right, on the borderline between AI and cognitive science. It is concerned with the way in which information might be stored in the human brain. "The main criteria for assessing a representation of knowledge are logical adequacy, heuristic power and notational convenience. Logical adequacy means that the representation should be capable of making all the distinctions that you want to make. Heuristic power means that, as well as having an expressive representation language, there must be some way of using representations so constructed and interpreted to solve problems." [27].

¹ This table is borrowed from [1].

Systems that think like humans	Systems that think rationally
The exciting new effort to make computers think machines with minds, in the full and literal sense.	The study of mental faculties through the use of computational models.
The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning.	The study of the computations that make it possible to perceive, reason, and act.
Systems that act like humans	Systems that act rationally
The art of creating machines that perform functions that require intelligence when performed by people.	Computational intelligence is the study of the design of intelligent agents.
The study of how to make computers do things at which, at the moment, people are better.	AI is concerned with intelligent behavior in artifacts.

Table 1.1 Some definitions of artificial intelligence

1.1 Knowledge

1.1.1 Definitions

Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world [6]. Knowledge is a theoretical or practical understanding of a subject or a domain. Knowledge is also the sum of what is currently known, and apparently knowledge is power [13].

1.2 Types of Knowledge

The types of knowledge that they can best represent. Table 1.2 lists the different types of knowledge.

Procedural knowledge: describes how a problem is solved. This type of knowledge provides direction on how to do something. Rules, strategies, agendas and procedures, are the typical type of procedural knowledge used in expert systems.

Declarative knowledge: describes what is known about a problem. This includes simple statements that are asserted to be either true or false. This also includes a list of statements that more fully describes some object or concept.

Meta-knowledge: describes knowledge about knowledge. This type of knowledge is used to pick other knowledge that is best suited for solving problem.

Heuristic knowledge: describes a rule-of-thumb that guides the reasoning process. Heuristic knowledge is often called shallow knowledge. It is empirical and represents the knowledge compiled by an expert through the experience of solving past problems.

Structural knowledge: describes knowledge structures. This type of knowledge describes an experts overall mental model of problem. The experts mental model of concepts, sub concepts and objects is typical of this type of knowledge [2].

Procedural knowledge	Rules Strategies Agendas Procedures
Declarative knowledge	Concepts Objects Facts
Meta-Knowledge	Knowledge about the other types of knowledge and how to use them
Heuristic knowledge	Rules of thumb
Structural knowledge	Rule sets Concept relationships Concept to object relationships

Table 1.2 Different types of knowledge

Some other definitions related to the subject such as:-

Strategic knowledge: is knowledge about how to approach a problem by choosing an ordering on methods and sub goals which minimizes effort in the search for a solution.

Support knowledge: is typically knowledge involving a causal model of the domain of discourse which explains why certain contingencies typically hold.

Domain knowledge: is a collection of specific facts (including concepts and relations) and procedures related to the solution of specific problem [24].

Knowledge engineer: is someone who is capable of designing, building and testing expert system.

Inference: which means determining actions, proving or disproving facts (hypotheses), or quite frequently, generating new knowledge.

Programmer: is the person responsible for the actual programming, describing the domain knowledge in terms that a computer can understand.

Knowledge base: contains the domain knowledge useful for problem solving [13].

Reasoning: is a process of working with knowledge, facts, and problem solving strategies to draw conclusion [2].

Deductive reasoning: to deduce new information from logically related known information.

Inductive reasoning: to deduce new general information from sure group facts or logically related known information.

Analogical reasoning: to deduce new information from analogical general information [3].

1.3 Components of Knowledge

The components of knowledge are shown in figure 1.1. Naming, Describing, Organizing, Relating and Constraining. These components of knowledge are generally present with many KR system. However, each KR method deals with these components in its own way.

Naming: the function of naming usually performed by proper nouns, the name is really a symbol that can stand in place of the actual object. So it is easy to denote objects by names. These names should be unique. If there are more than one object called by the same name, some methods exist for picking out objects uniquely. These methods include:

- Adding more details to the name .
- Assigning a special code.
- Attaching properties to the name .

Describing: to describe the important properties that an object has in representing the knowledge about the object. The basic process of description is very straightforward, it can lead to different forms of expression in different KR methods .

Organizing: for organizing objects into conceptual categories, one way of organizing objects is to describe some objects as instances of more general objects.

Relating: after one has named, described, and organized objects, one need to relate them. Some types of relation, such as family relations, can be expressed naturally in terms of the organization of knowledge. Other types of relations are best expressed as procedures involving inference. The difference between properties of objects and relations between objects is sometimes a source of confusion. For example, the fact that "Ahmad is an employer" can be regarded as either a property of Ahmad or a relationship between Ahmad and each of his employees.

Constraining: constraining governs the properties of objects. They are used to limit ranges of values, relationships, and organizational structures.

After one acquires knowledge from the expert on some well-focused domain, one will want to encode a knowledge in the system. To do this, one will need to find a way of structuring the knowledge in the system that allows the system to solve the problem in a manner similar to that followed by the expert. This is the subject of the study and is formally known as KR.

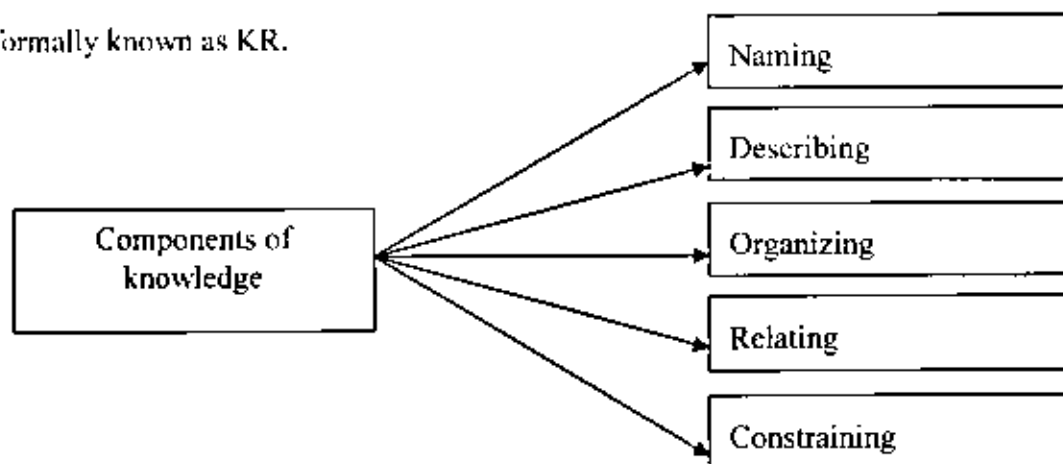


Figure 1.1 Components of knowledge

1.4 Knowledge Representation

1.4.1 Definitions

Knowledge representation, is the method used to encode knowledge in an expert systems knowledge base [2]. "It is a scheme or device used to capture the essential elements of problem domain." [24].

1.4.2 Characteristic of Knowledge Representation

The primary characteristic of KR are twofold :-

- What information is actually made explicit?
- How the information is physically encoded for subsequent use by the very nature of it?

Therefore, KR is goal directed. In real-world applications of intelligent machines, it can be said that a good solution depends on a good representation of knowledge [24].

1.4.3 Limitations of Knowledge Representation

To date in AI implementation suggest that, with modest application objectives, this is a reasonable assumption. However, several generic and difficult representation issues remain unresolved, for example:-

- What are the trade-offs between the sophistication of control structures and knowledge bases?
- How can an initial representation and control strategy be systematically refined on the basis of new knowledge, experience (learning), and perhaps recognition of mistakes? [24].

1.4.4 Main types of Knowledge Representation

There are two main types of KR. the first support analysis, the second uses in actual coding. The relation between the two types and knowledge engineering is represented in

figure-1.2². Knowledge analysis methods are using to support knowledge acquisition in the target definition stage as well as in the initial accumulation by initial and organize knowledge analysis, after finishing the organization process, the process coding will start by applying one or more of the coding methods. The semantic network is an example of analysis methods and the frames, rules is an example of coding methods [3].

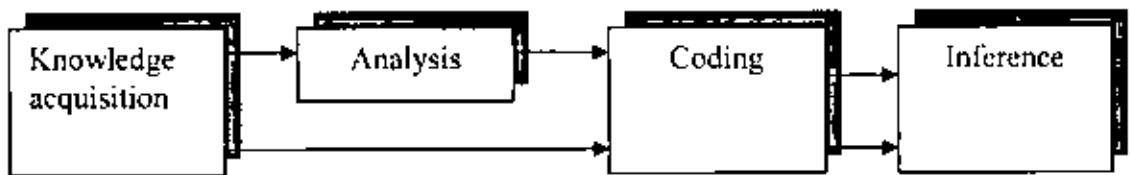


Figure 1.2 Relation between two types of KR

1.4.5 Advantages of Knowledge Representation

- Verification of the knowledge rights and ability of improvement.
- Test and discover the system errors due to perfect design and correct documentation.
- The assistance of knowledge completeness where the styles encourage the knowledge completeness and the permanent test of the system logic.
- The simple way for knowledge maintenance as a result of the little work to understand and coding concerning the knowledge in the rule itself.
- Increasing the quantity of produced software's as a result of good documentation and direct contact with the user and the improvement of the knowledge rule .
- Insure the easy way of the coding operations due to structural design of the knowledge inside the rule [3].

² This figure is borrowed from [3].

1.5 Related work in Knowledge Representation

One of the difficulties in this thesis is the lack of papers and researches studying the comparison between the kinds of KR. A survey of related work is presented. The papers focused on general idea about various KR techniques.

- One study has been done and appeared in the journal AI related to genetic algorithms, has concentrated with the application of techniques defined by genetic algorithms and applied to specific e-commerce applications. Problems and prospective solutions for encoding key characteristics of both the problem space and candidate solutions are presented and evaluated. This study has been concluded, the application of genetic algorithms to e-commerce applications is almost in its infancy [14].
- Another research related to the use of logic, concentrated to present a system that is able to perform cooperative information-seeking dialogues for the interrogation of a text knowledge base. In this system each event (utterance) is represented by logic programming facts which are used to dynamically update the previous user model. The system has to represent four levels of knowledge using dynamic logic programming. The knowledge levels are: Interaction, Domain, Information retrieval and Text. This study has been concluded:
 1. A system which is able to cooperatively participate in dialogues, namely in information-seeking dialogues.
 2. The system uses dynamic logic programming to represent and to reason about events.
 3. The interaction level is responsible for the dialogue management.
 4. Cooperation is achieved through the inference of user attitudes using the KR [18].

- Other work presented a new architecture for the World Wide Web as the Semantic Web. In broad terms, it encompasses efforts to populate the Web with content which has formal semantics. This will enable automated agents to reason about Web content, and produce an intelligent response to unforeseen situations. They believed that in order to build the Semantic Web, the sharing of ontological information is required. This allows agents to reach partial shared understanding and thus interoperate. The paper discussed required and desirable features of ontological languages, giving examples of the possible usage of frame-based representation and ontologism on the Semantic Web [19].
- Another research reviews motivations for introducing fuzzy sets and showed how this logic is an absolute requirement for understanding complex systems in terms of their components. This study has been concluded, conventional approaches represent complex systems in a reductionism manner by specifying well-defined components and their individual interactions [20].
- Previous work discussed the knowledge maintenance and frame problems, the author has concluded, that in AI there has long been recognized a basic problem in the internal representation of the world known as the "frame problem", the frame problem is one concerning internal representation about the world. It is not a problem in physics or the law of nature [22].
- The aim of work was to give some insight of workshop attends into the role and features of the rule of language, how it is evolving, and how it fits in the standard world, the study has been concluded, common rule interchange for rule engines remains an interchange proposition, with rule being developed to directly represent the rule executed in rule engine, commercial demand for standardized rule representation will likely increase. However, should note that rule representation is also directly to data representation, that separation of business

rules from data is required to support commercial application and that business rule as defined by business people are not necessarily equivalent to executable rules [23].

- Another study, evaluates the significance of recent development in the field of neural networks with respect to the field of AI. The study has been concluded, the AI community should regard neural networks as neither threat nor menace but rather as an opportunity for further, production interaction and exploration. It is certainly the case that recent empirical work emphasizes the good sense of this philosophy [26].

1.6 Thesis goals

There is no single theory to explain human knowledge organization or a best technique for structuring data in a conventional computer program, no single KR structure is ideal. One of the important responsibilities as a knowledge engineer is to choose the KR technique best suited for the given application. To accomplish this task:

- A person must have an understanding of the various KR techniques.
- Define various KR techniques.
- Determine advantages and disadvantages of the various KR techniques.
- Compare between various KR techniques.
- Determine types of KR that they might be the best to represent knowledge.

This thesis compares between some kinds of KR such as Rule, Logic, Semantic Network, Frames, Fuzzy Logic, Neural Network and Genetic Algorithms representation for some real applications depending on theoretical perspective and practical perspective using some criteria such as variables, statement/relations, programming and reasoning technique for selecting the best KR to be used for solve given problem.

1.7 Thesis Contributions

This thesis makes some contributions:

- It determines advantages and disadvantages of the various KR techniques.
- It determines the best of KR that they can be used to represent knowledge.
- It provides an application example to illustrate the various KR techniques.
- It compares between some kinds of KR through theoretical perspective and practical perspective.

1.8 Thesis Organization

This thesis is organized into nine chapter and they are follows:

Chapter two discusses the definition of rule representation, rule-based system model, structure of rule-based system, representing knowledge in rule-based systems, types of rules, variable of rules, rule sets, advantages and disadvantages of rules.

Chapter three focuses on topics such as the definition of logic representation and type of logic such as propositional, predicate, second order and higher order logic additional to difference between kinds of logic, advantages and disadvantages of logic.

Chapter four explains the definition of semantic networks representation, expanding and inheritance in semantic networks, exception handling, advantages and disadvantages of semantic network.

Chapter five presents the definition of frame, basic frame design, class frame, instance frame, frame inheritance, hierarchical structure and facets, more ever advantages and disadvantages of frame.

Chapter six presents the definition of fuzzy logic, fuzzy variables, fuzzy sets, forming fuzzy sets, fuzzy set representation, fuzzy set operation, fuzzy inference additional to advantages and disadvantages of fuzzy logic.

Chapter seven defines neural network, neural network components, network topologies, processing unit, representation capability, types of variables, advantages and disadvantages of neural network.

Chapter eight defines genetic algorithms and genetic algorithms operations, elements of genetic algorithms, advantages and disadvantages of genetic algorithms.

Chapter nine compares between the above knowledge representation using the criteria's variable, statement/relations, programming and reasoning technique. The thesis conclusion is including in the chapter nine .

Chapter Two

Rule Representation

2.1 Definitions

A rule is a knowledge structure that relates some known information to other information that can be concluded or inferred to be known. A rule is a form of procedural knowledge. It associates the given information to some action. This action may be the assertion of new information or some procedure to perform. In this sense, a rule describes how to solve a problem [2]. Any rule consists of two parts: the IF part, called the antecedent (premise or condition) and the THEN part called the consequent (conclusion or action).

The basic syntax of a rule is:-

IF <antecedent> THEN <consequent>

Examples:-

- IF "age of the customer <18"
THEN "signature of the parent is required"

For this simple example, if the age of the customer less than 18 years, then the rule infers that signature of the parent is required. In general, a rule can have multiple premises joined with AND statements (conjunctions), OR statements (disjunctions), or a combination of both. However, it is a good habit to avoid mixing conjunction and disjunctions in the same rule [13].

- IF <antecedent 1>
AND <antecedent 2>
.
.
AND <antecedent n>
THEN <consequent 1>

- IF <antecedent 1>
OR <antecedent 2>
.
.
OR <antecedent n>
THEN <consequent 1>

Example:-

IF the season is autumn
AND the sky is cloudy
AND the forecast is drizzle
THEN the advice is 'take an umbrella'

The consequent of a rule can also have multiple clauses:-

- IF <antecedent>
THEN <consequent 1>
<consequent 2>
.
.
<consequent m>

Conclusion can contain a single statement or a combination joined with an AND.

Example:-

IF [message for files and programs]="Abort. Ignore, Retry. Fail?"
THEN[the action1] = "press A for exit and close operation" OR
[the action2]="press I for ignore and continue work" OR
[the action3]="press R for radial with the same operation" OR
[the action4]="press F for end the current operation"

2.2 Rule-based system model

The production system illustrated in figure 2.1 using the following modules:-

- Knowledge base: which contains the domain knowledge useful for problem solving(models a humans long-term memory as a set of rules).
- Working memory: models a human's short-term memory and contains problem facts both entered and inferred by the firing of the rules.

- **Inference engine:** it links the rules given in the knowledge base with the facts provided in the database (models human reasoning by combining problem facts contained in the working memory with rules contained in the knowledge base to infer new information).

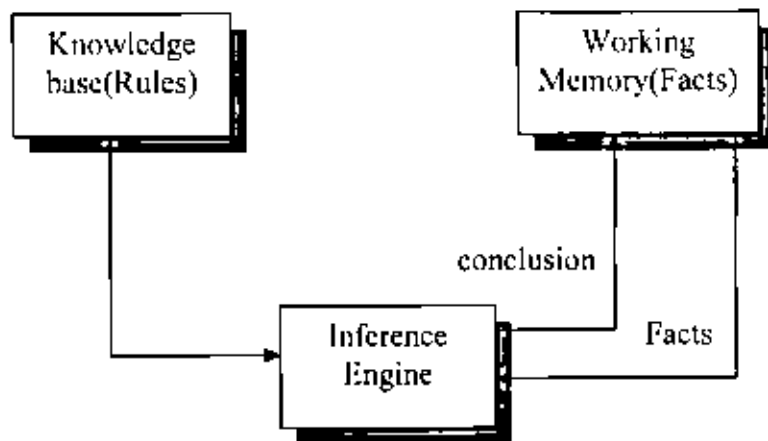


Figure 2.1 Rule-based model

2.3 Structure of rule-based system

The production model is based on the idea that humans solve problems by applying their knowledge to a given problem represented by problem specific information. The production rules are stored in the long-term memory and the problem-specific information or facts in the short-term memory. The production system model and the basic structure of a rule-based system are shown in figure 2.2. A rule based system have five components:-

1. **Knowledge base:-** contains the domain knowledge useful for problem solving.
2. **Database:-** includes a set of facts used to match against the IF(condition) parts of rules stored in the knowledge base.
3. **Inference engine:-** carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database.

4. Explanation facilities:- enable the user to ask the expert system how a particular conclusion is reached and why a specific fact is needed.
5. User interface:- is the means of communication between a user seeking a solution to the problem and an expert system.

In the rule-based system, the rules contained in the knowledge base represent the production contained in the long-term memory and the facts contained in the working memory represent the situation in the short-term memory. The inference engine acts the reasoning module of the production system model and compares the facts with the antecedents or premises of the rules to see which once can fire. Those rules that can fire have their conclusion added to the working memory and the process continues until no other rules have antecedents that match the facts contained in the working memory [13].

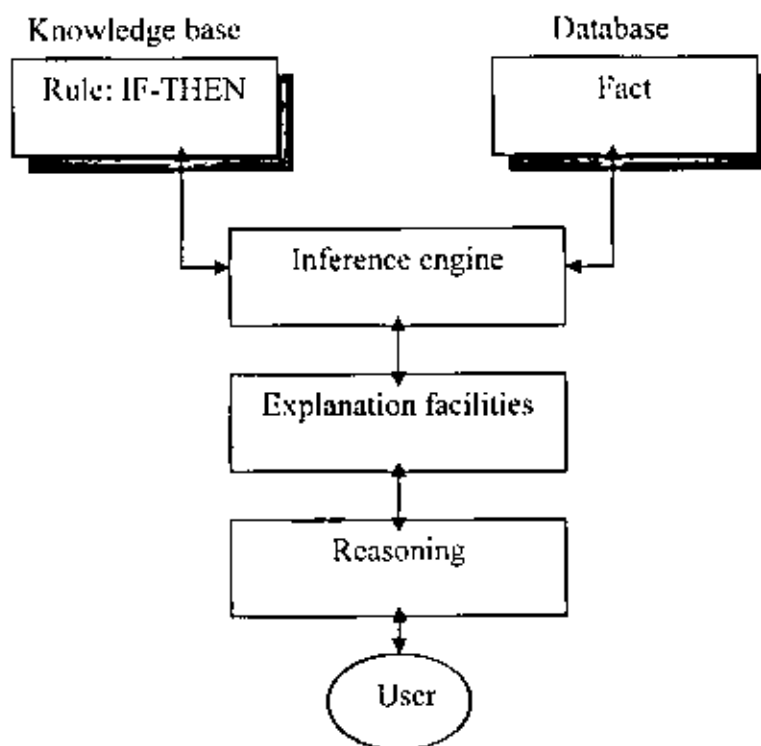


Figure 2.2 Basic structure of a rule-based system.

Figure 2.3 shows an example of this process.

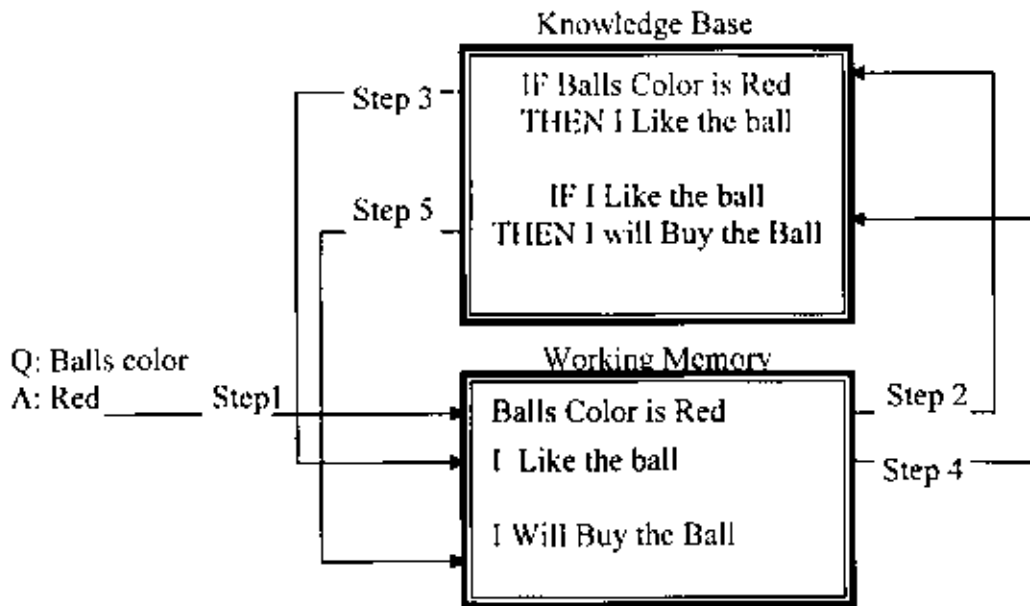


Figure 2.3 Rule-based system operation

The process begins with the system asking the user for the ball's color.

Step 1: The system then takes the answer of "Red" and enters this fact into the working memory .

Step 2: This assertion matches the premise of the first rule.

Step 3: The match causes the rule to fire and its conclusion "I like the ball" is added to working memory.

Step 4: This new information matches the premise of the second rule.

Step 5: This causes the rule to fire and the fact " I will buy the ball" is also added to the working memory. At this point, no other rules exist for the system to consider, so the processing stops.

2.4 Rule-based system architecture

The point 2.2 described a rule-based system as being composed of three modules: knowledge base, inference engine, and working memory. These three components comprise the heart of the system, but there are other subsystems that one will find in any

real system. The complete architecture of a rule-based system is shown in figure 2.4.

The additional subsystems are as follows:-

- User interface: it is the means of communication between a user seeking a solution to the problem and an expert system.
- Developer interface: it is the knowledge engineer develops the system.
- Explanation facilities: which enable the user to ask the expert system how a particular conclusion is reached and why a specific fact is needed.
- External programs: that programs such as database, spreadsheets. etc., that work in a support role for the system [2].

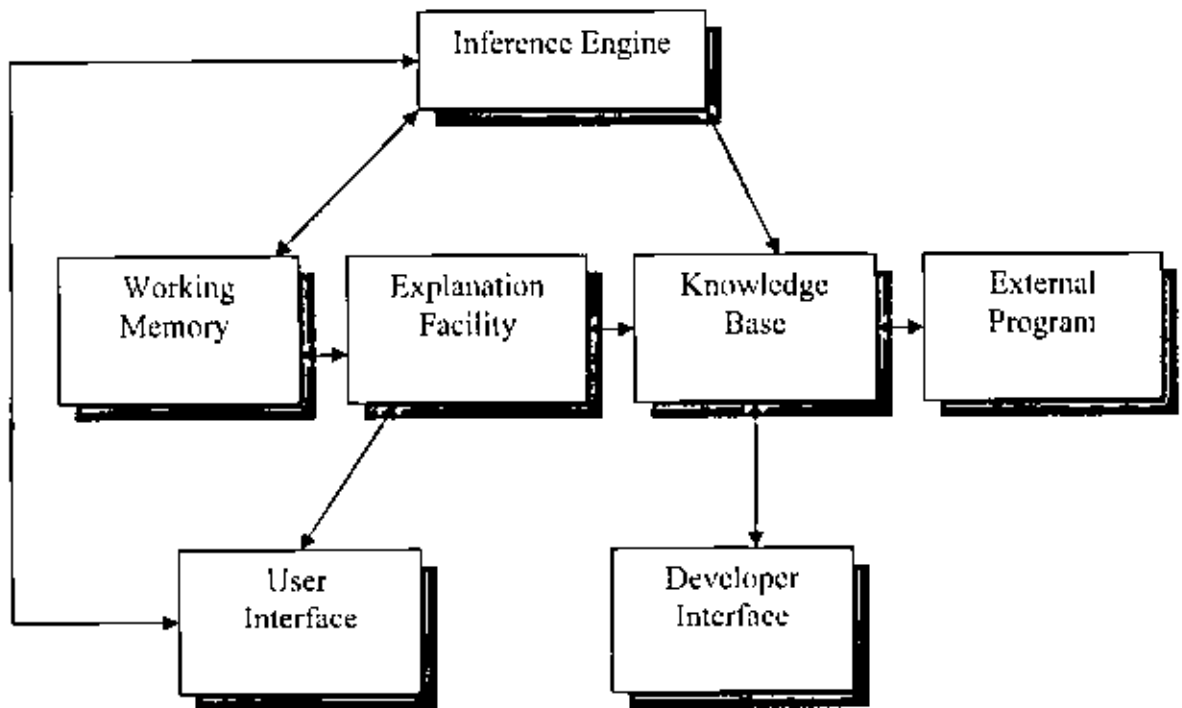


Figure 2.4 Rule-based system architecture

2.5 Representing knowledge in rule-based systems

The rule-based system used in the examples represents knowledge with production rules, so named because new facts are produced when a rule is proven true.

Example: To emulate the car diagnostic condition, replace human advisor with a computer, the rule can be written in the table 2.1:-

<p>RULE 1: IF the result of switching on the headlights is nothing happens or the result of trying the starter is nothing happens THEN the recommended action is recharge or replace the battery</p>	<p>RULE 2: IF the result of trying the starter is the car cranks normally and a benzene smell is not present when trying the starter THEN the benzene tank is empty</p>
<p>RULE 3: IF the benzene tank is empty THEN the recommended action is refuel the car</p>	<p>RULE 4: IF the result of trying the starter is the car cranks normally and a benzene smell is present when trying the starter THEN the recommended action is wait 10 minutes, then restart flooded car.</p>

Table 2.1 Car diagnostic condition rules

The complete rule statements for this example shown below:

RULE [Is the battery dead?]

IF [the result of switching on the headlights] = "nothing happens" OR
[the result of trying the starter] = "nothing happens"
THEN [the recommended action] = "recharge or replace the battery"

RULE [Is the car out of benzene?]

IF [the benzene tank] = "empty"
THEN [the recommended action] = "refuel the car"

RULE [Is the battery weak?]

IF [the result of trying the starter] = "the car cranks slowly"
AND [the headlights dim when trying the starter] = "true"
THEN [the recommended action] = "recharge or replace the battery"

RULE [Is the car flooded?]

IF [the result of trying the starter] = "the car cranks normally"
AND [a benzene smell] = "present when trying the starter"

THEN [the recommended action] = "wait 10 minutes, then restart
flooded car"

RULE [Is the benzene tank empty?]

IF [the result of trying the starter] = "the car cranks normally"

AND [a benzene smell] = "not present when trying the starter"

THEN [the benzene tank] = "empty"

2.6 Types of Rules

Rules can represent relations, recommendation, directive, strategy and heuristic as the following list illustrates:-

Relationship

IF The fuel tank is empty

THEN The car is dead

Recommendation

IF The car will not start

THEN Take a cab

Directive

IF The car is dead

AND The fuel tank is empty

THEN The action is refuel the car

Strategy

IF The car is dead

THEN The action is check the fuel tank;

Step 1 is complete

IF step1 is complete

AND the fuel tank is full

THEN the action is check the battery;

Step 2 is complete

Heuristic

IF The car will not start

AND First check out the fuel system then check out the electric system.

Rules can also be categorized according to the nature of the problem solving strategy- often called the problem solving paradigm [13]. The following list shows typical rules found in some of the common paradigms:-

Interpretation problem

IF Voltage of power supply is 220 volts

AND The power supply is ok

AND The electric is not dead

THEN The computer will be started

Diagnosis problem

IF The computer start

AND The bib of speakers long

AND not display on screen

THEN the display card is dead

Design problem

IF Current task is assigning a power supply

AND The position of the power supply in the cabinet is known

AND There is space available in the cabinet for the power supply

THEN Put the power supply in the cabinet

2.7 Variable of Rules

In some applications, one will need to perform the same operation on a set of similar objects. One could write a single rule for each object, but this approach is inefficient and

makes it difficult to maintain the system. Consider for example the following rule that concludes whether computer can install windows Xp:

```
IF Computer is IBM kind
AND Speed processor above 1 Gb
AND RAM above 128 Mb
THEN One can install windows Xp
```

This rule checks if kind of computer is IBM and speed processor above 1 Gb and RAM above 128 Mb then windows Xp can be installed. If someone wanted the system to perform the same check for other individuals, would a similar rule need to write for each kind of computer an inefficient process. In addition, if the basic knowledge changes, for example if the retirement kind of computer changes someone need to make the appropriate change to every rule a maintenance problem. To avoid these types of problems, choose one of the computer languages or available shells that offer powerful pattern-matching rules. These rules include variables that are to be used to match similar problem statements [2].

```
IF computer is ?A kind
AND speed processor above ?B GB
AND RAM above ?C MB
THEN one can install windows XP
```

2.8 Rule sets

Through out the experience, an expert forms several sets of rules to apply to a given problem. One set of rules may be applicable for a given problem, while being useless for another problem. For example, in a computer system may have separate sets of rules for the electrical and display system when encounters an electrical problem, then uses only those rules formed from past experiences when working on the computer electrical system. To illustrate the organization and use of rule sets, consider figure 2.5. This

figure shows various rule sets that a computer mechanic might use. They are organized in a hierarchical fashion-a style typically found in cognitive studies on human problem solving. Each block in the figure 2.5 represents a set of rules related to the blocks label, upper level blocks contain a set of abstract rules for general problem solving. This type of modular structure provides a natural ordering of the domains rule sets, it also proposes a top-down approach to problem solving, where specific rule sets are used only when appropriate. The principal advantage of this design :-

- It is that it represent a natural approach to solving complex problems.
- It eases the system's development and maintenance.
- It allows to integrate different KR techniques and inference strategies into the system.

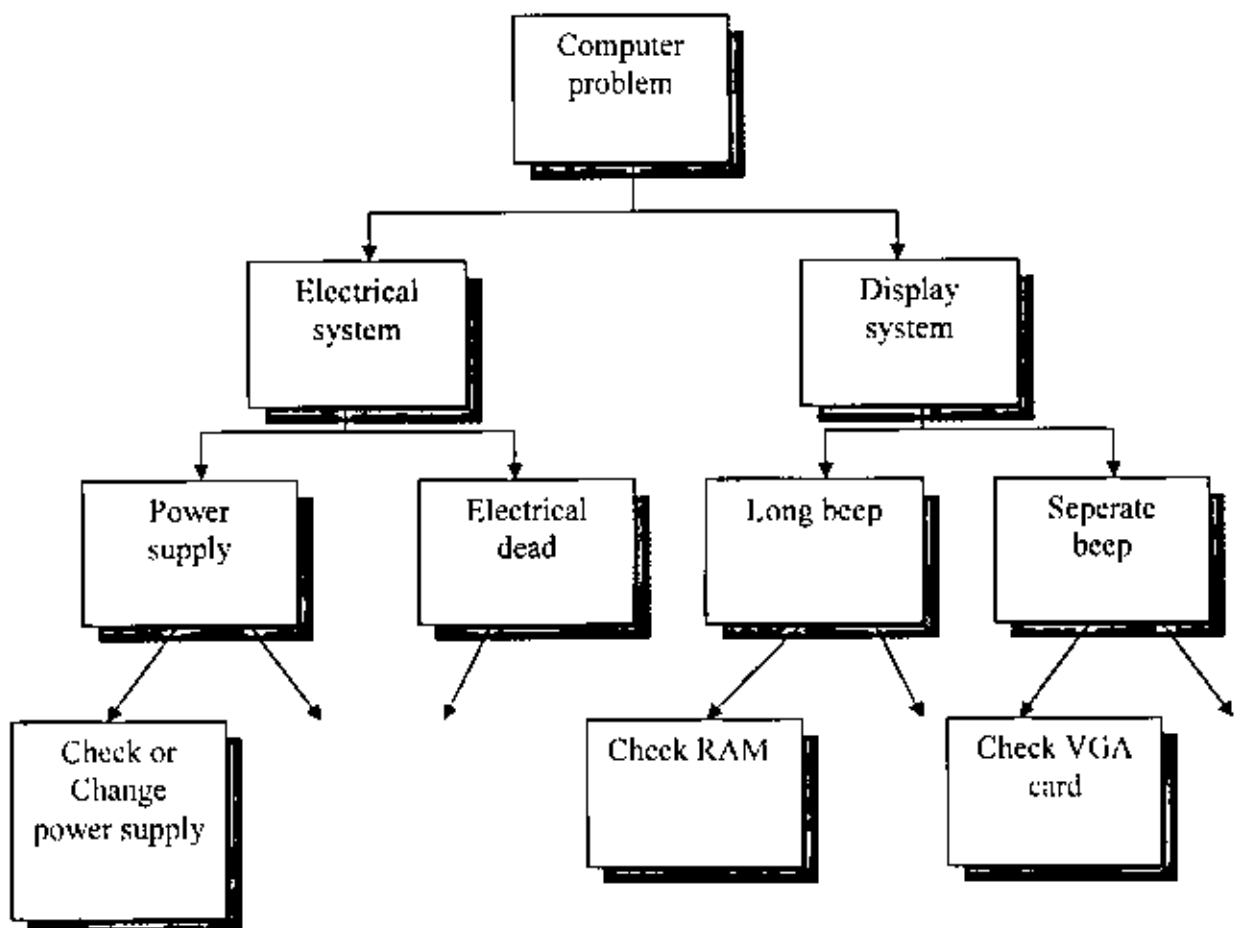


Figure 2.5 Computer diagnostic rule sets

2.9 Advantages of Rules

- Natural expression: ease of capturing a knowledge in a rule makes a rule-base approach an attractive choice for the design of the expert system.
- Separation of control from knowledge: permits one to change the systems knowledge or control separately.
- Modularity of knowledge: it can be easily reviewed, verified, and corrected.
- Ease of expansion: the separation of the systems knowledge from its control permits to easily add additional rules allowing for a graceful expansion of the systems knowledge.
- Propositional growth of intelligence: it is capable of telling the system something new about the problem from established evidence.
- Use of relevant knowledge: a rule-base system is likely to have many rules that can address a number of problem issues.
- Derivation of explanations from rigid syntax: it offers an opportunity to determine how some piece of information came to be placed in the working memory.
- Consistency checking: allows for consistency checking of the system to assure that the same situation does not lead to different action.
- They allow us to view through the syntax to the meaning. And easy for humans to read and understand .
- Rules may be changed without affecting other parts of knowledge base.
- Rules allow us to quickly create prototypes to test ideas and prove feasibility [4].

2.10 Disadvantages of Rules

- Require exact matching the antecedents of the available rule with the facts in the working memory.
- Have opaque rule relationships: it is often difficult to determine how rules are logically related through an inference chain.
- Can be slow: this occurs because when the inference engine is deciding which rules to apply, it must scan the entire set of rules.
- It provides only one choice for representing the problems knowledge in the expert system.
- Rules are not allowed to call other rules directly.

Example:- Computer Diagnostic

To design a computer diagnostic system, someone need to acquire knowledge about troubleshooting in computers. Computer manuals often include troubleshooting section, which consider possible problems with the system start-up, computer/peripherals (hard disk, keyboard, monitor, printer), disk drives(floppy disk, CD_ROM), files.

In this case study I will consider troubleshooting the general system, once the prototype expert system is developed, someone can easily expand it. In general a person should match the features of the problem with the capabilities of the tool. These tools range from high-level programming language such as LISP,PROLOG,JAVA and shells such as E2glite, shell provide us with the built-in inference engine, explanation facilities and the user interface. In this case study I am using E2glite shell. The complete computer diagnostic screen shot shown in APPENDIX I.

Chapter Three

Logic Representation

3.1 Definitions

A logic is a mathematical tool for constructing and manipulating symbolic expressions [5]. Logic is the analysis of methods of reasoning. The oldest form of KR in a computer is logic. Over the years, several logic representation techniques have been suggested and studied. The ones most often linked with intelligent systems have been propositional logic and predicate calculus. Both techniques use symbols to represent knowledge and operators applied to the symbols to produce logical reasoning. Table 3.1 lists the propositional logic operators and their common symbols, they offer a formal well-founded approach to KR and reasoning [2].

Operator	Symbol
AND	$\wedge, \&, \cap$
OR	$\vee, \cup, +$
NOT	\sim, \neg
IMPLIES	\supset, \rightarrow
EQUIVALENCE	\equiv

Table 3.1 Logic operators and symbols

3.2 Propositional Logic

It is a logic which deals with propositions. A proposition is a sentence which is either true or false. The "true" and "false" are called the truth values [7]. A propositional variable is a symbol used to represent a proposition that is considered to be indivisible for instance, such as: $A =$ The computer will start

The symbolic variable chosen could be of any form, such as : $A, M, K, TEMP, com_stat.$

In proposition logic, if one were concerned with the truth of the statement "The computer will start" he would check the truth of the variable A . For many problems one

is concerned with the truth of logically related propositions. Consider for example the following rule:-

IF The car will not start \longrightarrow A
 AND It is too far to walk to work \longrightarrow B
 THEN I will miss work today \longrightarrow C

Propositional logic provides logical operators, such as AND, OR, NOT, IMPLIES, EQUIVALENCE, that allow us to reason with various rule structures. Table 3.1 lists the propositional logic operators and their common symbols. Using the symbols of table 3.1, one can write the rule as: $A \wedge B \rightarrow C$

The first three operations of table 3.1 are similar to those found in Boolean logic, as shown in the following truth tables:-

AND		
A	B	A AND B
F	F	F
F	T	F
T	F	F
T	T	T

OR		
A	B	A OR B
F	F	F
F	T	T
T	F	T
T	T	T

NOT	
A	A NOT A
F	T
T	F

The EQUIVALENCE operator returns a T only when both propositions have the same truth assignment:-

EQUIVALENCE		
A	B	A \equiv B
F	F	T
F	T	F
T	F	F
T	T	T

IMPLIES		
A	B	A IMPLIES B
F	F	T
F	T	T
T	F	F
T	T	T

The IMPLIES operator is handled in a special manner. This function defines an implication between propositions of the form:-

$$C = A \text{ IMPLIES } B$$

$$C = A \rightarrow B$$

That is for the implication C, if A is true, then B is implied to be true. The above truth table illustrates the operation of implication. The IMPLIES or implication operator returns an F when A is T and B is F, otherwise it returns a T. Another way of viewing this is that C is true if A is false OR B is true, and C is false only when A is true and B is false[2]. That is, the following two statements are equivalent: -

$$A \rightarrow B \equiv \neg A \vee B$$

Example 3.2.1 :-

To better illustrate the operation of the implication operator, it is valuable to view the implication in the form of a rule such as:

IF The electrical is dead
THEN The computer wont start

From the first row in the implication truth table (A=T, B=T), our example rule could be interpreted:

Electrical not dead \rightarrow computer will start; implication T

From the second row:

Electrical not dead \rightarrow computer won't start; implication T

From the third row:

Electrical is dead \rightarrow computer will start; implication F

From the fourth row:

Electrical is dead \rightarrow computer wont start; implication T

Example 3.2.2:-

Order test A for all male over 70, smokers with family history of cancer, and women with chronic cough and family history of cancer. Otherwise, do not order it.

- Male: a person being male
- Old: a person being over 70

- Smoker: a person being a smoker
- Cough: a person having chronic cough
- FHC: a person having family history of cancer
- Order A: order test A

One can represent this problem by logic equation and truth table:-

$$(Male \wedge \neg Young) \vee (Smoker \wedge FHC) \vee (\neg Male \wedge Cough \wedge FHC) \leftrightarrow Order\ A$$

Male	Young(<=70)	Smoker	FHC	Cough	Order Test A
T	T	T	T	T	T
T	T	T	T	F	T
T	T	T	F	T	F
.....

3.3 Predicate Logic

A predicate logic is a branch of logic that allows modeling of the truth of statements based upon the values assumed by specific portions (or phrases) of the statements. Where a logic is concerned not only with sentential connectives but also with the internal structure of atomic propositions it is usually called a predicate logic [24]. A predicate logic also called predicate calculus, the predicate calculus like propositional logic, uses symbols to represent knowledge. These symbols may represent either constants, predicates, variables, or functions. Predicate calculus also permits to operate on these symbols using the propositional logic operators. In predicate calculus, a fact or proposition is divided into two parts: predicate and argument. The argument represents the object or objects of the proposition and the predicate an assertion about the object. For example, to represent the proposition "Ahmad likes Ali" one would write: Likes(Ahmad,Ali). In predicate calculus representation, the first word of each expression (i.e., likes) is a predicate denoting some relationship between the arguments within the

parentheses. The arguments are symbols denoting objects within the problem. The standard convention represents predicates with the first character in lowercase [2].

3.4 First order logic

First-order logic permits reasoning about the propositional connectives (as in propositional logic) and also about quantification ("all" or "some"). A classic, if elementary, example of what can be done with the predicate logic is the inference from the premises:-

- All men are mortal.
- Ahmad is a man.

To the conclusion :- Ahmad is mortal.

In order to proceed formally with our objective of exploring first order logic, the following additional definitions are necessary:-

- A term is either a constant, a variable, or an n -variable function.
- A function is an n -tuple of terms prefixed by a function symbol or a name that satisfies the definition of a function [24].

3.5 Second order logic

Which contains two-sorted language, this means that there are two distinct sorts of variables which are intended to range over the set $w = \{0,1,2,\dots\}$ of all natural numbers, variable of the second sort are known as asset variables, are denoted by x,y,z,\dots , and are intended to range over all subset of w [25].

3.6 Higher order logic

It is a simple technique for implementing languages with functional programming. Object variables and binders are implemented by variables and binders in the host language.

$$\text{Variable_predicate_name}(\text{Arg1}, \text{Arg2}, \dots, \text{Argn}) \quad (3.6.1)$$

A simple representation of the form of (3.6.1) is possible, the consequences of the manipulation of an entity of this type are significant. Without further constraints, it is quite easy to derive paradoxes in higher-order logic.

▪ **Relations**

The term relation is typically used of a predicate which is applied to more than one thing, e.g. "greater than", which is applied to two things to make a comparison, but can also be used for predicates taking one or zero things. The number of "things" involved (as arguments) is called the arity of the predicate or relation.

▪ **Variables**

Variables are used to represent general classes of objects or properties. Variables are written as symbols beginning with an uppercase letter. For example, to capture the propositions "Ahmad likes Ali" and "Moftah likes Daw" one can write: Likes(X,Y).

In this case, variables assume or instantiate the values:-

X = Ahmad, Moftah,

Y = Ali, Daw

In predicate calculus, variables can be used as arguments in a predicate expression or a function [2].

▪ **Operations**

Predicate calculus uses the same operators found in propositional logic. To illustrate their use, consider the following:-

proposition: Ahmad likes Ali likes(Ahmad,Ali) proposition:

Moftah likes Ali likes(Moftah,Ali)

These two predicates indicate that two different men like Ali. To account for the obvious jealousy that might occur, you could write:

likes(X,Y) AND likes(Z, Y) IMPLIES NOT likes(X,Z)

or likes(X,Y) \wedge likes(Z,Y) \rightarrow \neg likes(X,Z)

This is a general implication that captures the knowledge: "If two different individuals like the same person, then they dislike each other" Applying the implication produces the result: $\rightarrow \text{likes}(\text{Ahmad}, \text{Moftah})$.

3.7 Difference between kinds of logic

- Propositional logic simply maps the truth value of a statement into TRUE or FALSE. A predicate is a parameterized proposition, that is, a proposition with variables .
- In propositional logic, the statements or propositions are variable-free. But in predicate the statement or a proposition, involves variables.
- There was no other explicit dependence of the truth of the statement on the individual contents (i.e., words or phrases) of the statement. A predicate function is a function that maps elements of A into the set {TRUE, FALSE}.
- There was no mechanism that allowed the possibility of a substitution of an element in the statement with another, with the consequent mapping of the statement truth value as a function of this substitution. A predicate function is taken to include statements containing variables (i.e., predicates), or simply variables.
- In propositional logic there is no ability to relate the contents of a portion of a statement with the statement truth value. A predicate with a single variable or argument are used to represent relations.
- First order logic imposes and indicates one restriction .
- Second order logic imposes and indicates two restrictions .
- Higher order logic: the consequences of the manipulation of an entity of this type are significant. Without further constraints, it is quite easy to derive paradoxes in higher-order logic.

3.8 Advantages of Logic

- Logic can be inference, this ability makes it a unifying formalism.
- There is no way to access the components of an individual assertion.
- Allows expression to contain variables. Variable let create general assertions.
- It is precise.
- It allows inference to establish new relationships from old.
- It allows programs to be written which are declarative, they describe what is true and not how to solve problems.
- Problems are solved by general purpose inferences such as Prolog resolution refutation.

3.9 Disadvantages of Logic

- One problem with logic is that there is a lot of vocabulary to keep straight.
- Humans do not always reason by making logical inferences.
- Logic is too rigorous, and inflexible, to be of use in all AI problem domains.
- It is not a good way for organizing knowledge .
- The processing inefficiency .

The complete screen shot for the solution of the example 3.2.1 shown in

APPENDIX II.

Chapter Four

Semantic Network Representation

4.1 Definitions

A semantic network is a method of KR using a graph made up of nodes and arcs where the nodes represent objects and the arcs relationships between the objects[2]. The first computer implementations of semantic networks were developed in the early 1960s for use in bird translation [8]. A semantic network using for non picture KR is the most basic structure upon which an identification tree is based. Simply put, a semantic network consists of nodes representing objects and links representing any relations between these objects [9]. A semantic network provides a graphical view of a problems important objects, properties and relationships. It contains nodes and arcs that connect the nodes. The nodes can represent objects, object properties or property values. The arcs represent the relationships between the nodes. Both the nodes and arcs have labels that clearly describe the objects represented and their natural relationships. A node for example might have the name "Bird", arcs are commonly labeled with terms such as "IS-A" "HAS" etc., that clearly define the relationships between connected nodes. Figure 4.1 shows a simple example of a semantic network. This figure has two nodes that represent objects, and two representing properties. The "Bird" node is connected to both property nodes, providing the interpretation "Bird" has wings and can fly. The Canary node is linked to the "Bird" node via an IS-A arc, i.e., "Canary is a bird" by virtue of linking the "Canary" node to the "Bird" node through an IS-A link, an important feature of a semantic network is working behind the scene. Since birds have wings and can fly, and Canary are a type of bird, it seems only reasonable that they also have wings and can fly. When one code into a computer the semantic network shown in figure 4.1, the

computer obtains the same understanding. It knows not only about objects and their properties, it can infer information about related objects in the hierarchy [2].

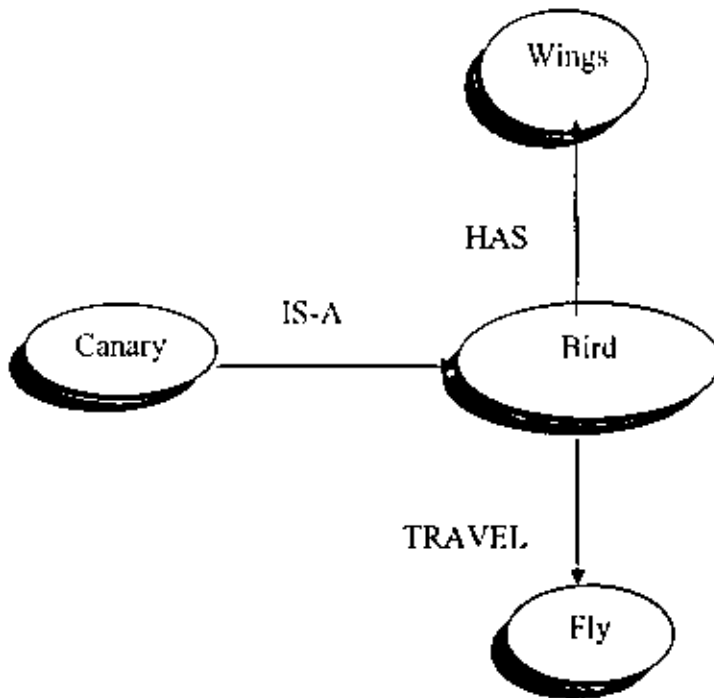


Figure 4.1 Semantic network of bird

4.2 Expanding Semantic Networks

It can be easily expanded a semantic network by simply adding nodes and linking them to their related nodes currently in the network. These new nodes represent additional objects or properties. One will usually add a new object node in one of three ways: -

- (1) a similar object.
- (2) a more specific object.
- (3) a more general object.

Consider figure 4.2 which shows all the three expansion approaches for our bird network. The network in figure 4.1 represents only one type of bird, i.e., "Canary" however, there are many other types such as penguin. As shown in figure 4.2, one can easily add nodes to the network that represents these new bird types. Here a "Penguin" node was added and linked to the "Bird" node through an IS-A arc. The IS-A link between the "Canary" and "Bird" nodes in figure 4.1 defines a specific-to-general

relationship between the two. That is, a canary is a specific type of bird. Using this idea, the second way you can expand a semantic network is to add a node that represents a more specific object. For example, as shown in figure 4.2 the node "Parrot" was added and linked to the "Canary" node through an IS-A arc. This figure not only tells you that parrot is a canary, but you can also infer that parrot is a bird, simply by following the IS-A links. The third way one can expand the network follows the previous idea. A node can be added which represents a more general object, and link it using an IS-A arc to the appropriate node. For example, as shown in figure 4.2 the node "Animal" was added and linked to the "Bird" node through an IS-A arc. This additional node adds a considerable amount of information to the network, someone not only knows that "a bird is an animal" but by tracing through the network one also knows that parrot is an animal and breathes air [2].

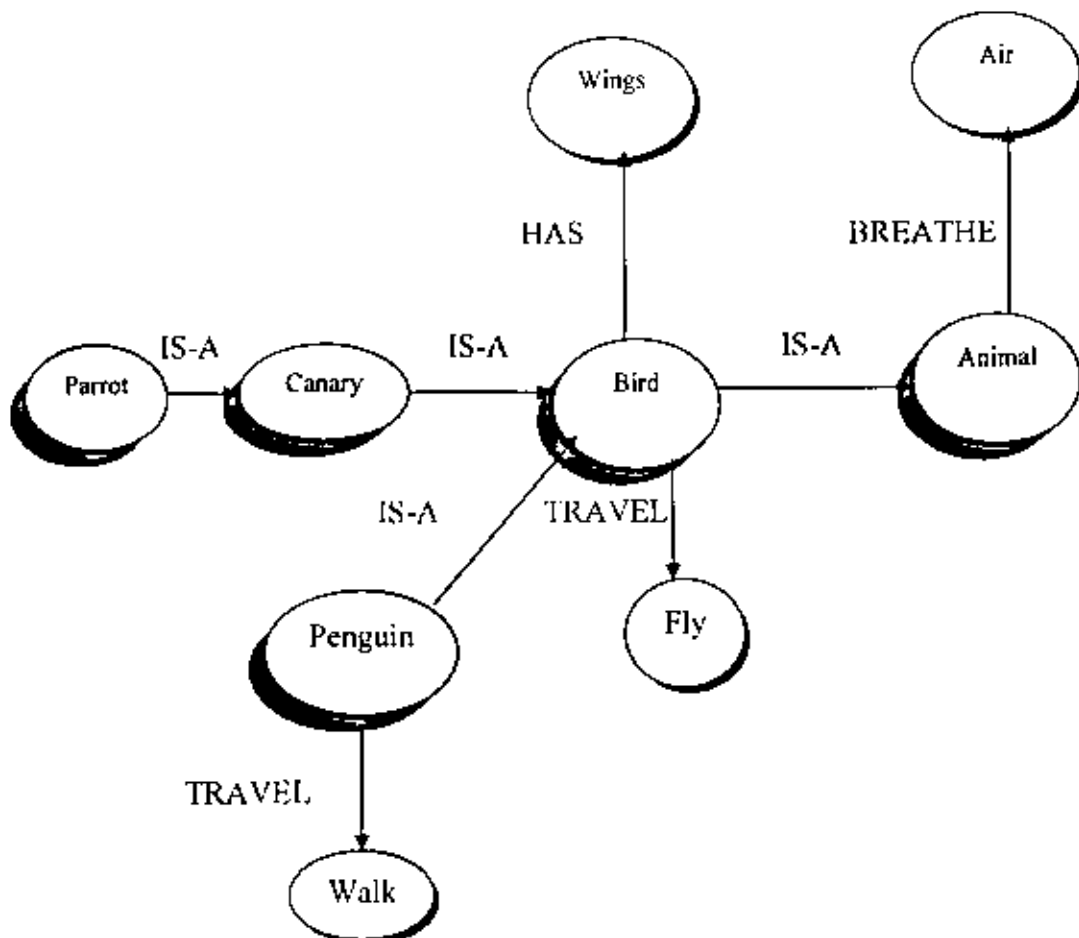


Figure 4.2 Expanded semantic network of bird

4.3 Inheritance in Semantic Networks

The last section in figure 4.2 illustrated how nodes added to a semantic network automatically inherit information from the network. For example, "Parrot" breathes air because it is an "Animal" this is an important feature of a semantic network and is formally called inheritance. The inheritance feature of a semantic network eases the task of coding knowledge. For example, if one adds some specific object node to the network (e.g., "Parrot"), it inherits information throughout the network via the IS-A links. In addition, if one adds a general object node (e.g., "Animal"), other nodes inherit its properties. It is this ability of a semantic network that made it attractive.

4.4 Defaults and inheritance semantic networks

Defaults and inheritance are ways of achieving some commonsense:-

- Inheritance is a way of reasoning by default, that is, when information is missing, fall back to defaults.
- Semantic networks represent inheritance.

Example:-

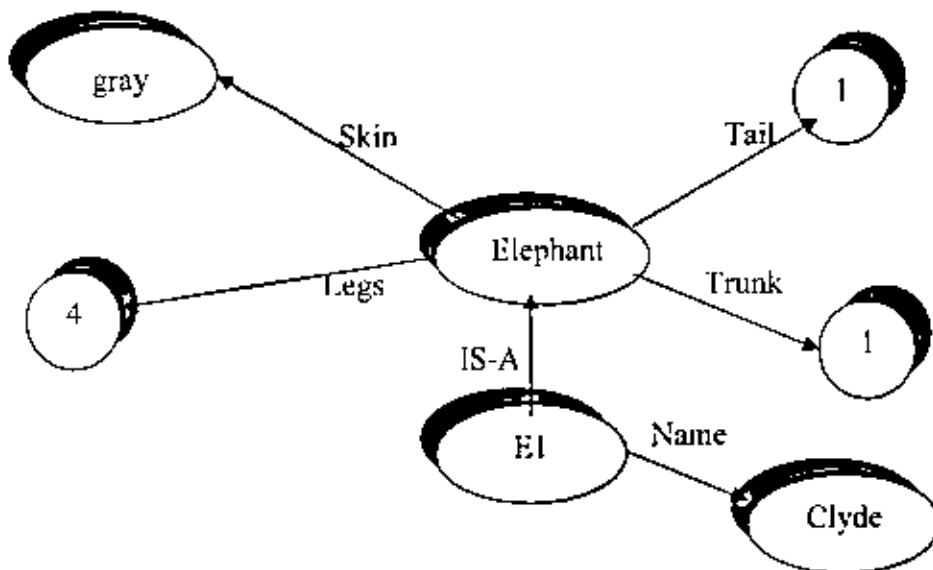


Figure 4.3 Defaults and inheritance of elephant

Using inheritance:-

- To find the value of a property of E1, first look at E1.
- If the property is not attached to that node, "climb" the isa link to the nodes parent and search there.
- IS-A signifies set membership: \in
- AKO signifies the subset relation: \subseteq
- Repeat, using IS-A/AKO links, until the property is found or the inheritance hierarchy is exhausted.
- Sets of things in a semantic network are termed types.
- Individual objects in a semantic network are termed instances.

4.5 Exception handling semantic networks

To illustrate the benefit of inheritance to a semantic network, lets look at one in action. Assume one coded into a computer the network of figure 4.2 one simple way to use a semantic network is to ask some node a question. For example, one could ask the "Bird" node, "How do you travel?" to answer the question, the node first looks for an arc labeled "travel", in this case that arc exists. The node then uses the information in the attached node as the answer, namely "Fly". Figure 4.4(a) illustrates this example as case 1. If the node is unable to locate the answer via a local arc, it then searches for an answer via its IS-A links. In a sense, the nodes related through IS-A links pass the question along until one of the nodes can provide an answer. Consider for example the same question posed to the "Parrot" node illustrated in figure 4.4(b) as case 2. The "Parrot" node has no way to answer the question so it asks the "Canary" node. Likewise, this node passes the question along to the "Bird" node. Here, as in the previous example, an answer of "Fly" is found, and is sent back along the links finally arriving at the user. As this last example illustrates, a semantic network equipped with an inheritance feature provides an efficient way to process information. Inheritance is a powerful feature in a

semantic network, but it can cause problems. Consider for example the "Penguin" node in figure 4.2 since this node is linked to the "Bird" node, it inherits the information "TRAVEL-Fly" in other words, I would expect that a penguin, by virtue of being a bird, can fly. This is obviously a mistake, one can correct using a technique known as exception handling. This technique requires account for exceptions on a local basis. When a node inherits incorrect information, one must link a new node to it with information that can effectively over-ride the inherited information. Exception handling is a simple technique used to avoid problems with a semantic network. However, the subject highlights an inherent weakness of semantic networks. If one fail to account for a necessary exception, obvious problems occur, one now has a flying penguin [2].

Case 1

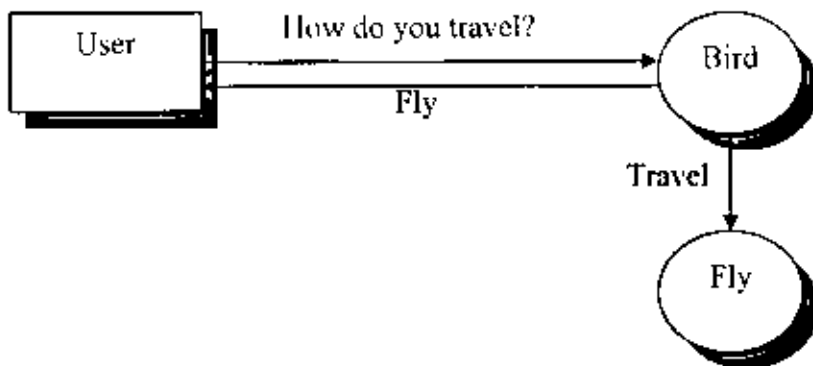


Figure 4.4 (a) Semantic network operation

Case 2

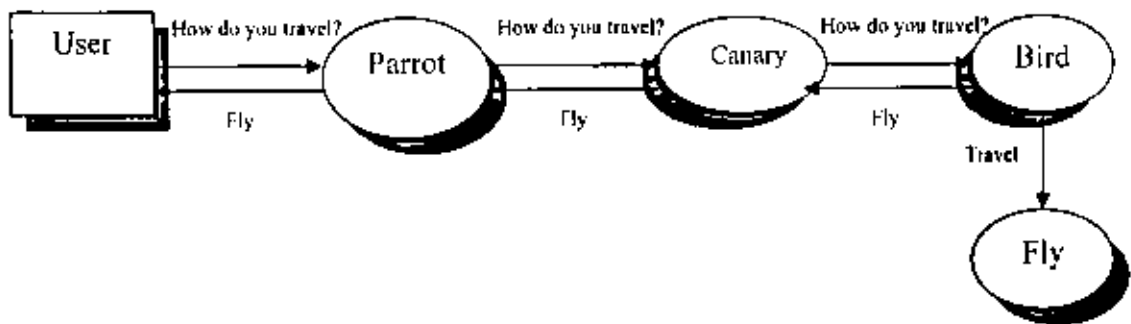


Figure 4.4 (b) Semantic network operation

4.6 Advantages of Semantic Network

- Provide a simple, economical, and relatively intuitive representation form, similar to the function of human information storage [22].
- Related knowledge is easily clustered.
- A semantic network provides a graphical view of a problems important objects, properties and relationships.
- Nodes belonging to a semantic network automatically inherit information from the network.
- Provides an efficient way to process information [3].

4.7 Disadvantages of Semantic Network

- The subject highlights an inherent weakness of semantic networks, if one fails to account for a necessary exception.
- There is no strict semantic information.
- somewhat misleadingly named.
- It only dealt with small pieces of knowledge.
- Knowledge that naturally comes in associated clusters is not easily dealt with inheritance (particularly from multiple sources and when exceptions in inheritance are wanted) can cause problems.
- Facts placed inappropriately cause problems.
- Representing procedural knowledge is a big problem.
- Complexity grows exponentially in nets with many relationships.
- Semantic networks are difficult to implement and difficult for programming.

Example: Elephant inheritance

The elephant has four legs and one tail with gray color skin, also has one trunk where it has circus performer under the name Clyde.

Elephant inheritance by semantic networks shown in figure 4.3. The complete screen shot in Prolog language for doing elephant inheritance and the following queries presented and the answers respectively, shown in APPENDIX III:-

- 1- Check an unambiguous fact about e1.
- 2- Check e1 has 3 legs, not 4.
- 3- Check e1 IS-A elephant.
- 4- Normal elephants have 4 legs
- 5- E1 IS-A elephant has gray skin.
- 6- IS-A elephant AKO mammal lactates.

Chapter Five

Knowledge Representation with Frame

5.1 Definitions

Frame is a data structure for representing stereotypical knowledge of some concept or object [2]. Frame is a natural extension of the semantic network in a schema, a schema is a unit that contains typical knowledge about some concept or object, and includes both declarative and procedural knowledge. For example, the schema of a bird might include knowledge that it has wings and legs, and that it hunts for food. A schema holds stereotypical information about a concept that can be applied to a specific situation for study. This representation supports the organization of knowledge into more complex units that reflect the organization of objects in the domain. According to Minsky, a frame is used when one encounters a new situation (or makes a substantial change in one's view of a problem) one selects from memory a structure called a "frame", this is a remembered framework to be adapted to fit reality by changing details as necessary. A frame may be viewed as a static data structure used to represent well-understood stereotyped situations. Frame-like structures seem to organize our own knowledge of the world [8].

5.2 Basic Frame Design

A frame is similar in appearance to many forms, such as the student report card shown in figure 5.1. This form has a name "Report card" fields such as "student name" and slots for entering field values. It provides the general type of information. The basic structure of a frame is shown in figure 5.2 like a form, it has a name "object1" that is the name of the object represented by the frame. For example, if you wanted to represent a person named Ahmad, one could use his name. A frame also has fields, "property1", "property2", etc., that are features or attributes that describe the object represented for our Ahmad

frame, these properties might be items such as his address, age, etc., each property has a slot for entering a specific value, i.e., "value 1". For example in Ahmad age slot, one might put a value of 19. Property values are of three general types: Boolean, string, or numeric. A frame also includes an optional field called "class", one can enter a value (i.e., "object2") that is the name of another frame related to "object". This relationship is usually of the "IS-A" type, discussed in the semantic network in chapter 4. That is, one can form an "object1 IS-A object2" relationship. For example, by placing the name "human" in the "class" slot of "Ahmad" one represents the relationship "Ahmad is a human", the value of establishing this link between the two frames is similar to that obtained when two semantic network nodes are linked together; "object1" inherits information from "object2" [2].

REPORT CARD	
Student Name:	<input type="text"/>
Address:	<input type="text"/>
Course	Grade
Artificial intelligence	<input type="text"/>
Data Base	<input type="text"/>
Data mining	<input type="text"/>
Networks	<input type="text"/>
.....	<input type="text"/>
.....	<input type="text"/>
.....	<input type="text"/>

Figure 5.1 Report card form

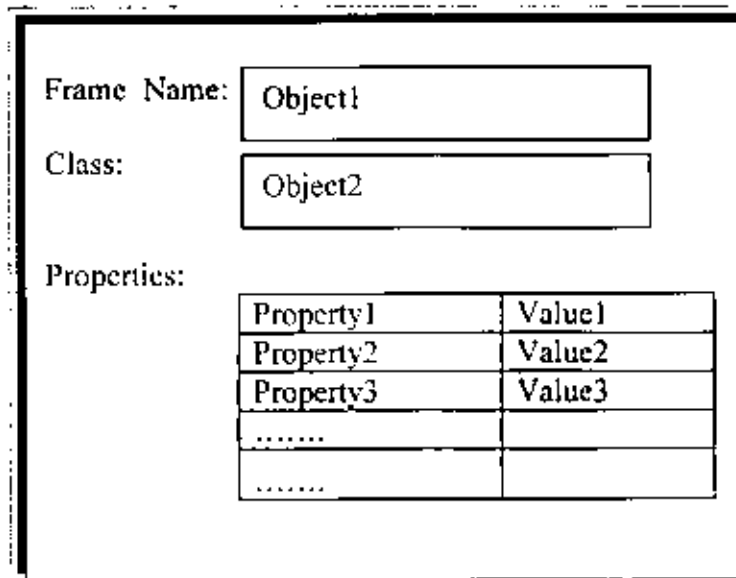


Figure 5.2 General frame structure

5.3 Class Frame

A class frame represents the general characteristics of some set of common objects. For example, one could create a class frame that describes such objects as cars, boats, or even intangible objects such as line pressures or temperatures. In each class frame one defines those properties that are common to all the objects within the class, and possibly default property values. Properties are of two general types: static or dynamic. A static property describes an object feature whose value doesn't change. A dynamic property is a feature whose value is likely to change during the operation of the system. Figure 5.3 shows an example of a class frame. The frame name "Bird" describes the object represented. The properties shown describe the general characteristics of most birds. Properties such as "Color" and "No. of wings" are static. They describe non changing features of the bird. The properties "hungry" and "activity" are dynamic during the operation of the system it is likely that these values will change. The property values shown in the figure are also typical for most birds. For example, most birds have two wings and can fly. A value of "Unknown" means that one can't ascribe a value to the property. For example, even though one knows that all

birds have a color. its not logical to assign a value to the "Color" property until one is representing a specific bird.

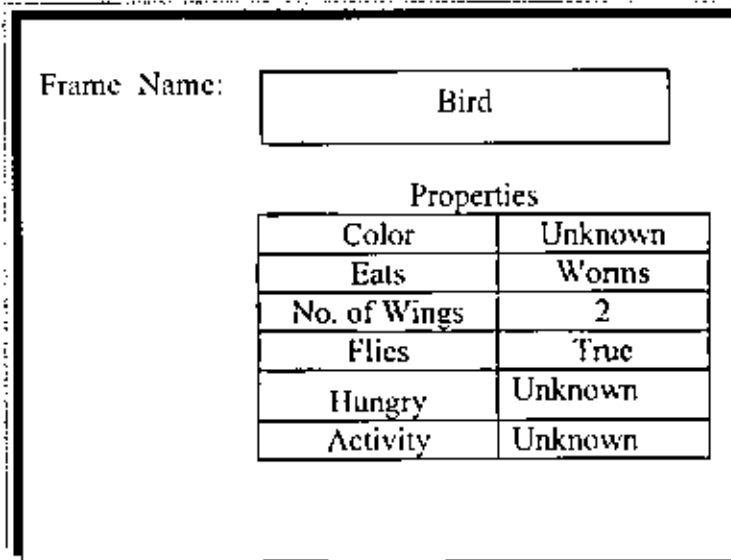


Figure 5.3 Bird frame

5.4 Instance Frame

An instance is a different type of frame to describe a specific instance of a class frame and-not surprisingly-it is called an instance frame. When someone creates an instance of some class, the frame inherits both properties and property values from the class. Can be changed the property and values to tailor the description of the object represented in the instance frame. It is possible to add additional properties to the instance if necessary. Figure 5.4 shows an example of an instance frame. In general, one can create many instances of the same class. One simply asserts that the new frame is an instance of the class, and it immediately inherits the class information. This approach greatly speeds system coding, especially when there are many instances to code.

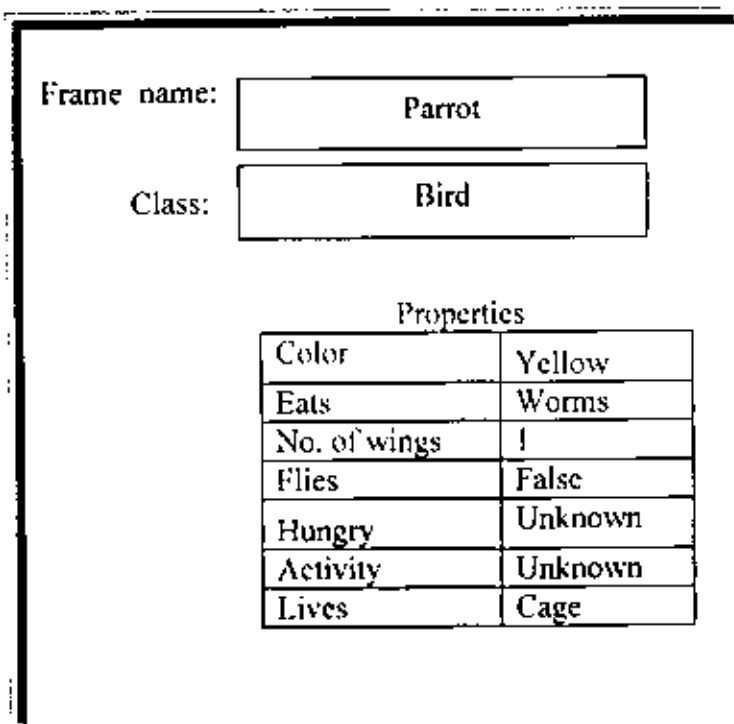


Figure 5.4 Parrot frame

5.5 Frame inheritance

Just as a node in a semantic network can inherit information from other nodes, an instance frame inherits information from its class frame. When someone creates an instance frame, beginning by asserting that the frame is an instance of some class. This assertion causes the instance frame to inherit all of the information from its class frame. Consider figure 5.4 which shows the frame of "parrot" an instance of a "Bird" since parrot is a bird, it inherits the information from figure 5.3 like most birds, parrot eats worms. However, if parrot has only one wing and thus can't fly. In general, one can allow an instance to accept the class default values or provide values unique to the instance. one can also provide unique properties in the instance. For example, if it is known that parrot lives in a cage, one can insert this information directly into the instance frame.

5.6 Inheritance behavior

Besides inheriting descriptive information from its class, an instance also inherits its behavior. To accomplish this, one need to first include within a class frame a procedure

(often called a method), that defines some action the frame performs. For example, in the class frame "Bird" of figure 5.4, the method can be written which tells every bird what to do if it becomes hungry.

5.7 Hierarchical structure

Besides having only a single class and its associated instances, complex frame structures can be created. Consider for example figure 5.5 which shows the world of birds arranged in a hierarchical structure. This structure organizes the concept of a bird at different levels of abstraction. The top level frame contains information common to all birds. The middle-level frames (called subclasses) contain information more specific to their individual category. That is, though robins and canaries share features common to all birds each possess some unique features that serve to discriminate between the two. The bottom frames are the specific bird instances, each instance inherits information from the top level "Bird" frame, and also features from their associated subclass. In general, an instance inherits information from its parent, grandparent, etc.

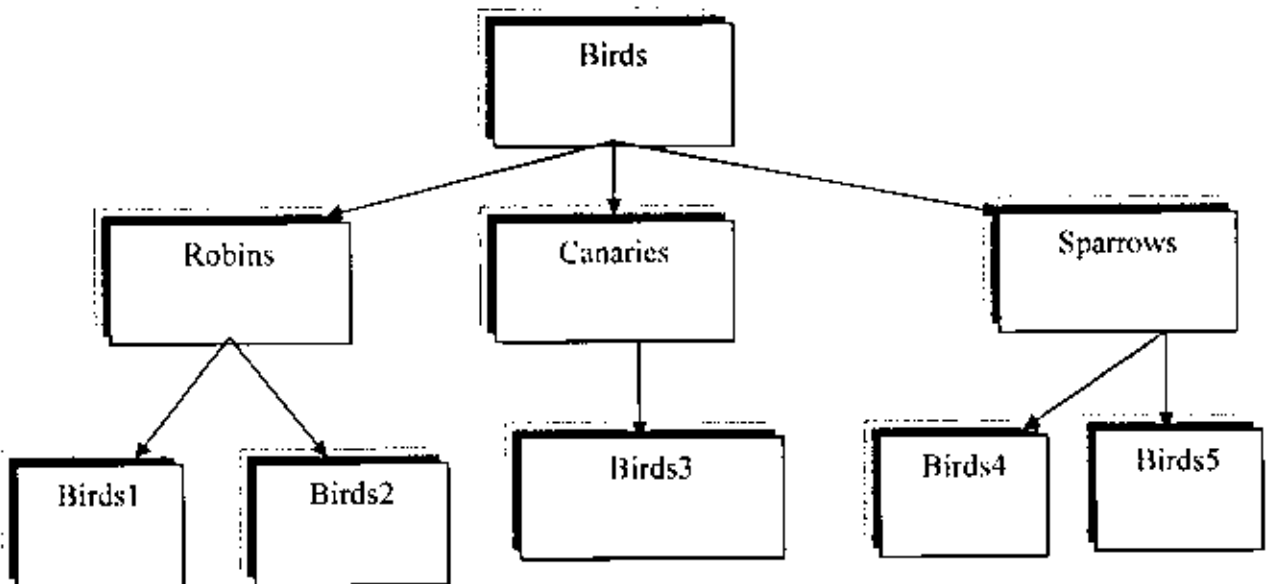


Figure 5.5 Frame world of birds

5.8 Facets

Facets provide one additional control over property values. One way someone can use a facet is to define a constraint on property value. For example, someone can limit a numeric propriety value to some set range or string value to one of a few possible values, also can use a facet to restrict the type of data that can be stored in a propriety value, i.e., string, numeric, or Boolean. Another very powerful way is to use a facet by instruct a property how to obtain its value or what to do if its value changes. These are formally called IF-NEEDED and IF-CHANGED facets.

IF-NEEDED: its used when the value depends on other property values that dynamically change during session.

Example: From figure 5.4

- IF Parrot: No. of wings<2
THEN Parrot: Flies=False
- IF Parrot: No. of wings=2
THEN Parrot: Flies=True

IF-CHANGED: its used when a frames property value impacts the property value of another frame. It can also be used to change a property value within its own frame.

Example:- From figure 5.4

- IF Self: Hungry = True
THEN Self: Activity = Eating # Self: Eats

This method first checks to see if the value of this property is equal to "True" if found, this method sets the "activity" property value to eating whatever the bird represented in the instance frame normally eats.

5.9 Advantages of Frame

- Frames add to the power of semantic nets by allowing complex objects to be represented as a single frame.

- Frames make it easier to organize knowledge hierarchically in a network, every concept is represented by nodes and links at the same level of specification. Very often, however, we may like to think of an object as a single entity for some purposes only [10].
- Procedural attachment is an important feature of frames because it supports the linking of specific pieces of code to appropriate entities in the frame representation.
- Frame systems support class inheritance. The slots and default values of a class frame are inherited across the class/subclass and class/member hierarchy.
- Frames can be reused in different application with minimum modification. This is due to the inheritance feature allowed in frames.
- Frames are easy to construct in general because of the fact that they represent the intuition thinking of human about concepts and objects.
- A frame may be an instance, i.e. it describes a particular object.
- Frames can inherit properties from generic frames.

5.10 Disadvantages of Frame

- Frames have drawbacks which have prevented them from being widely used in business: intelligibility and computing requirements.
- Maintenance is some times difficult with relationships between frames are complex .

Example:- Car registration in traffic office

To design a computer car registration program need, to acquire knowledge about car specifications and car owner. The knowledge can be manipulate by any computer language as shown in APPENDIX IV.

Chapter Six

Fuzzy Logic Representation

6.1 Definitions

Fuzzy logic is a branch of logic that uses degrees of membership in sets rather than a strict true/false membership [2]. Fuzzy logic is an infinite valued logic in that truth values can range from zero to one. Like classical logic, fuzzy logic is concerned with the truth of propositions [7]. Fuzzy logic is a mathematical technique for allowing items to be in more than one mathematical set [11]. Fuzzy logic is a superset of conventional (Boolean) logic that has extended to handle the concept of partial truth – truth values between “completely true” and “completely false”. The basis of fuzzy logic theory lies in making the membership function lie over a range of real numbers from 0.0 to 1.0. The fuzzy set is characterized by (0.0,0.1,0). Fuzzy logic operates on a concept of membership such as the statement Ali is old can be translated as Ali is a member of the set of old people and can be written symbolically as $m(\text{old})$, where m is the membership function that can return a value between 0.0 and 0.1 depending on the degree of membership. In figure 6.1 the objective term 'tall' has been assigned fuzzy values. At 150 cm and below, the person does not belong to fuzzy class while for above 180, the person certainly belongs to category 'tall'. However, between 150 and 180 the degree of membership for the class 'tall' can be assigned from the curve varying linearly between 0 and 1. The fuzzy concept 'tall ness' can be extended into 'short', 'medium' and 'tall' as shown in figure 6.2 [9].

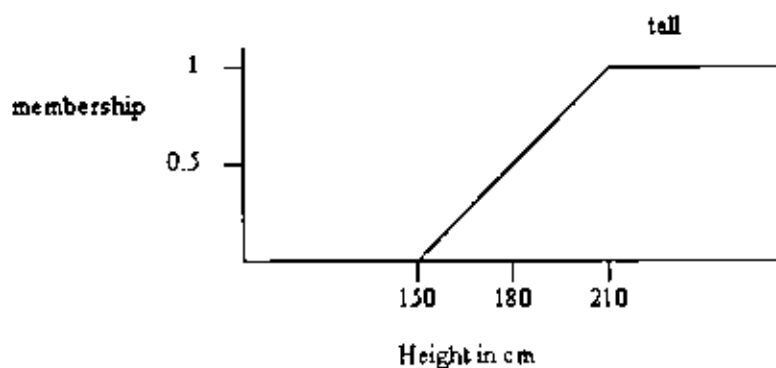


Figure 6.1 Graph showing membership functions for fuzzy set 'tall'

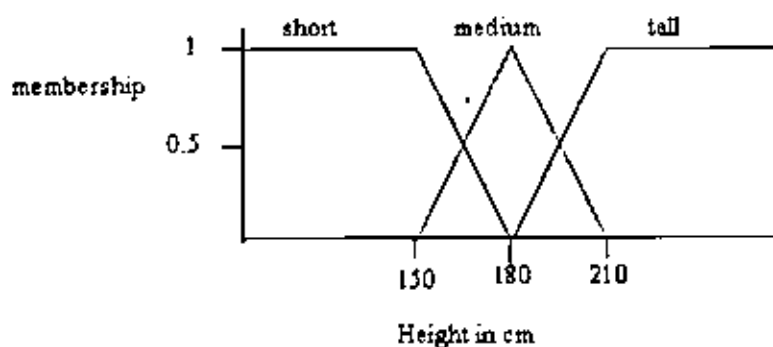


Figure 6.2 Graph showing membership functions for fuzzy sets "short", "medium" and "tall"

6.2 Fuzzy variables(linguistic variable)

Linguistic Variable	Typical Values
Temperature	Hot, cold
Height	Short, medium, tall
Speed	Slow, creeping, fast

Table 6.1 Examples of linguistic variables with typical values

Term used in our natural language to describe some concept that usually has vague or fuzzy values. Table 6.1 shows examples of linguistic variables and typical values that we might assign to them. In fuzzy expert systems, one uses linguistic variables in fuzzy rules. A fuzzy rule infers information about a linguistic variable contained in its conclusion from information about another variable contained in its premise. For example:-

Rule 1

IF Speed is slow.
THEN Make the acceleration high

Rule 2

IF Temperature is low
AND Pressure is medium
THEN Make the speed very slow

One calls the range of possible values of a linguistic variable the variables universe of discourse. For example, One might give the variable "speed" used in Rule 1 the range between 0 and 100 km/h. The phrase "speed is slow" occupies a section of the variables universe of discourse it is a fuzzy set [2]. In contrast to traditional variables that take on numeric values, the values taken on by a linguistic variable are word or phrases from natural language. The purpose of linguistic variables is to provide a linkage to the numerical/logical demands of the computer and the imprecise or uncertain facts and rules comprising most of our knowledge about the world and how to function in it. [21].

Example:-

The linguistic variable "age" may have the value "young" for one person, "very old" for another person, and "neither old nor very young" for a third. Other values that the linguistic variable "age" may take on include fuzzy numbers such as around 25 or roughly 35 to 40, and crisp values such as 36 years and 56 days old. A linguistic variable of this type has well defined universe of discourse, in this case of "age" this universe consists of real numbers between 0 and 120 representing actual chronological ages. Corresponding to each syntactically legal word or phrase in the natural language is a fuzzy set of values(e.g., chronological ages) from the universe of discourse of the linguistic variable.

6.3 Fuzzy sets

A fuzzy set is a set to which members of the universe of discourse may belong in varying degrees, opposed to the all or nothing membership characteristic of conventional "crisp" sets [21]. A fuzzy set assigns membership values between 0 and 1 that reflect more naturally a members association with the set. For example, if a person age is 5, one might assign a membership value of 0.9, or if the age is 13, a value of 0.1. In this example "age" is the linguistic variable and "young" one of its fuzzy sets. Other sets that one might consider are "old," "middle-age," etc. Each of these sets represent an adjective defined on the linguistic variable. In general, a fuzzy set provides a graceful transition across a boundary as illustrated in figure 6.3, the x-axis, or universe of discourse, represents a persons age. The y-axis is the fuzzy set membership value. The fuzzy set of "young" people maps age values into corresponding membership values. One can see from the figure that our 11 year-old person is no longer suddenly a child. The person is gradually removed from this classification as his age increases. Fuzzy logic assigns values to the event on the basis of a membership function defined as: $\mu_A(x) \rightarrow [0, 1]$.

In fuzzy logic, event or element x is assigned a membership value by a membership function μ . This value represents the degree to which element x belongs to fuzzy set A .

$$\mu_A(x) = \text{degree}(x \in A)$$

the membership value of x bounded by the following relationship:-

$$0 \leq \mu_A(x) \leq 1$$

The membership function μ returns a value between 0 and 1 that represents the degree of membership [12].

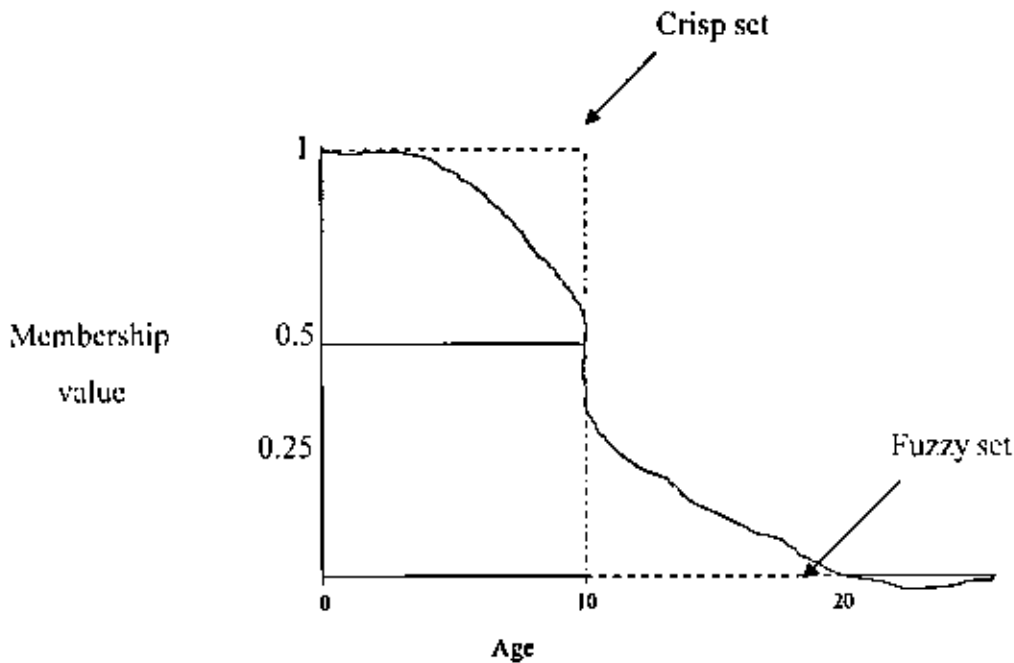


Figure 6.3 Fuzzy and sets of “young” people

6.4 Forming fuzzy sets

To represent a fuzzy set in a computer, one need to define its membership function. One approach that we can use is to Ahmad a group of people for their understanding of the term that are attempting to represent by the fuzzy set. For example, consider the concept of a tall person. One could ask each of these individuals to what degree they believe a person of a given height is tall. After acquiring answers for a range of heights, one could perform simple averaging to produce a fuzzy set of tall people. One can now use this function to ascribe a belief (or membership value) to a given individual that they belong to the fuzzy set of tall people. One could continue this polling to account for other height descriptions such as short, or medium. In this fashion one can obtain fuzzy sets that reflect the popular opinion of most people for each of these classifications. This point is illustrated in figure 6.4 where fuzzy sets are shown in a piecewise linear form for the issues of three different categories of an individuals height. When one defines multiple fuzzy sets on the same universe of discourse, the fuzzy literature often refers to them as fuzzy subsets. By forming fuzzy subsets for various vague terms, one can ascribe a membership value of a given object to each set. Consider figure 6.3 again. An

individual of a height of 5.5 feet is a member of medium persons with a membership value of 1, and at the same time a member of short and tall persons with a value of 0.25. This is an interesting result—a single object is considered a partial member of multiple sets [2].

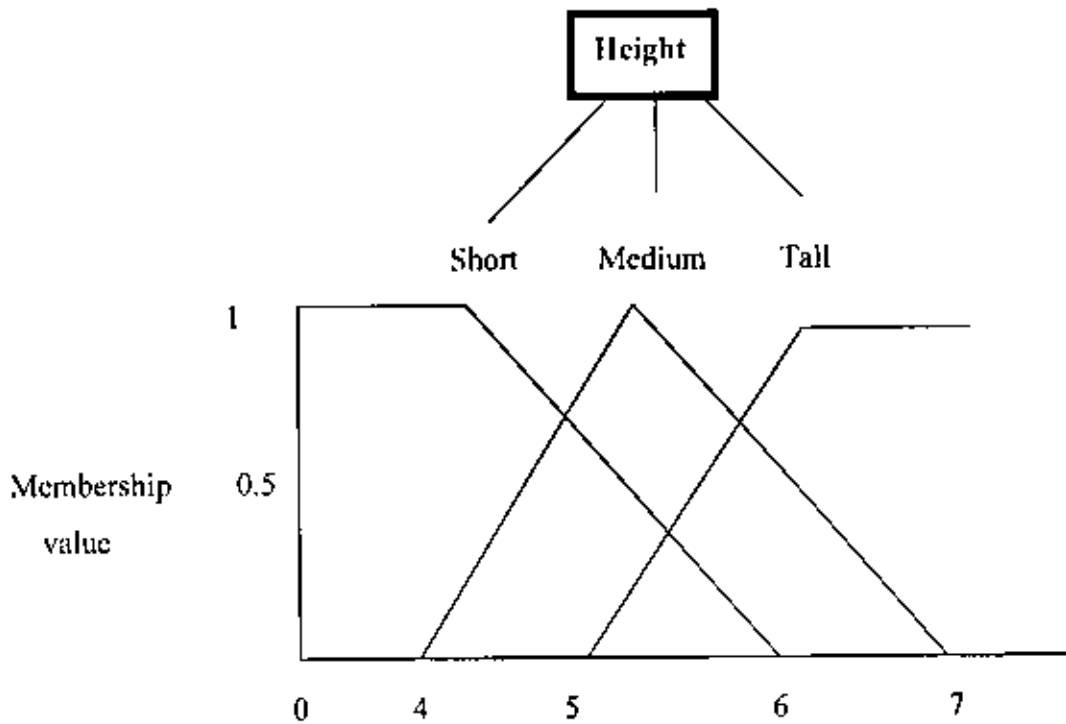


Figure 6.4 Fuzzy sets on height

6.5 Fuzzy set representation

Assume one have a universe of discourse X and a fuzzy set A defined on it. Further assume one has a discrete set of X elements $\{x_1, x_2, x_3, \dots, x_n\}$. The fuzzy set A defines the membership function $\mu_A(x)$ that maps the elements x_i of X to the degree of memberships in $[0,1]$. The membership values indicate to what degree x_i belongs to A (see equation 1). For a discrete set of elements, a convenient way of representing a fuzzy set is through the use of a vector:

$$A = (a_1, a_2, a_3, \dots, a_n) \text{ where } a_i = \mu_A(x_i) \text{ (equation 1)}$$

For a clearer representation, the vector often includes the symbol “ f ” which associates the membership value a_i with its x_i coordinate:

$$A = (a1/x1, a2/x2, a3/x3, \dots, an/xn)$$

As an example, consider the fuzzy set of tall people shown previously in figure 6.4

$$TALL = (0/5, 0.25/5.5, 0.7/6, 1/6.5, 1/7)$$

6.6 Fuzzy set operation

1. Intersection

The membership function of the intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of the two individual membership functions. This is called the minimum criterion. (The intersection operation in fuzzy set theory, is the equivalent of the AND operation in Boolean algebra).

$$\begin{aligned} \mu_{A \cap B}(x) &= \min(\mu_A(x), \mu_B(x)) \text{ for all } x \in X \quad (\text{equation 2}) \\ &= \mu_A(x) \cap \mu_B(x) \end{aligned}$$

According to figure 6.4 and equation 2

$$TALL = (0/5, 0.2/5.5, 0.5/6, 0.8/6.5, 1/7)$$

$$SHORT = (1/5, 0.8/5.5, 0.5/6, 0.2/6.5, 0/7)$$

$$\mu_{TALL \cap SHORT} = (0/5, 0.2/5.5, 0.5/6, 0.2/6.5, 0/7)$$

When one considers the term "tall and short" it is reasonable to interpret the term as meaning "medium" with this interpretation, one would expect the highest degree of membership to be in the middle of the set, and the lowest at the set limits. This example illustrates how two fuzzy sets can be combined to form a new set. The linguistic term one might apply to this new set might be medium height persons.

2. Union

The membership function of the Union of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of the two

individual membership functions. This is called the maximum criterion. (The Union operation in fuzzy set theory, is the equivalent of the OR operation in Boolean algebra)

$$\begin{aligned} \mu_{A \cup B}(X) &= \max(\mu_A(x), \mu_B(x)) \text{ for all } x \in X \quad (\text{equation 3}) \\ &= \mu_{A(x) \cup B(x)} \end{aligned}$$

According to figure 6.4 and equation 3

$$\text{TALL} = (0/5, 0.2/5.5, 0.5/6, 0.8/6.5, 1/7)$$

$$\text{SHORT} = (1/5, 0.8/5.5, 0.5/6, 0.2/6.5, 0/7)$$

$$\mu_{\text{TALL} \cup \text{SHORT}} = (1/5, 0.8/5.5, 0.5/6, 0.8/6.5, 1/7)$$

The results indicate the union membership attains its highest values at the limits and its lowest at the middle of the set. A linguistic interpretation of this new set might be not medium.

3- Complementation (Not)

The membership function of the complement of a fuzzy set A with membership function

μ_A is defined as the negation of the specified membership function. This is called the negation criterion. (The complement operation in fuzzy set theory is the equivalent of the NOT operation in Boolean algebra).

$$\mu_{\bar{A}}(X) = 1 - \mu_A(X) \quad (\text{equation 4})$$

According to figure 6.4 and equation 4

$$\mu_A(X) = \text{TALL} = (0/5, 0.2/5.5, 0.5/6, 0.8/6.5, 1/7)$$

$$\mu_{\bar{A}}(X) = \text{NOT TALL} = (1/5, 0.8/5.5, 0.5/6, 0.2/6.5, 0/7)$$

The following rules which are, common in classical set theory also apply to fuzzy set theory:- Associatively, Commutative, Distributive [9].

6.7 Fuzzy inference

Fuzzy inference as a process of mapping from a given input to an output. The process is performed in four steps:-

1. **fuzzification of input variables:-** is to take the crisp inputs, and determine the degree to which these inputs belong to each of the appropriate fuzzy sets.
2. **Rule evaluation:-** is to take the fuzzified inputs and apply them to the antecedents of the fuzzy rules.
3. **Aggregation of the rule outputs:-** is the process of unification of the output of all rules. In other word, One takes the membership functions of all rule consequents previously clipped or scaled and combine them into a single fuzzy set.
4. **Defuzzification :-** helps us to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. The input for defuzzification process is the aggregate output fuzzy set and the output is a single number [13].

6.8 Advantages of Fuzzy Logic

- Give flexibility in the way data are input by using linguistic variables.
- The linguistic variables have the flexibility to handle a knowledge at various levels of specificity or vagueness and in a natural and expressive vocabulary [21].
- It mimics the way in which humans interpret linguistic values.
- The transition from one linguistic value to a contiguous linguistic value is gradual rather than abrupt, resulting in continuity and robustness [12].
- Provides both an intuitive method for describing systems in human terms and automates the conversion of those system specifications into effective models [9].
- Poses a problem with one deal with concepts that are not sharply defined [17].
- The information flow in the system is completely transparent [29].

6.9 Disadvantages of Fuzzy Logic

- The entire system has to be built up manually.
- No automated training exists.
- When the system complexity increases, it becomes more challenging to determine the correct set of rules and membership functions to describe system behavior.
- A significant time investment is needed to correctly tune membership functions and adjust rules to obtain a good solution.
- For complex systems, more rules are needed, and it becomes increasingly difficult to relate these rules.
- For many systems, it is impossible to find a sufficient working set of rules and membership functions.

Example:- Car type determination

The car has passed nearby by, we would like to determine the type according to several questions:-

- 1- Is it open or close cabinet or both of them.
- 2- Is taxi or private or both of them.
- 3- How old the driver is?.
- 4- How tall the driver?.
- 5- How much the distance over seat.
- 6- How much the speed of the car.

The available information:-

- 1- The cars are two types, closed, open and both of them.
- 2- The cars are taxi, private and both of them.
- 3- The age of the driver is of the follow intervals:-
 - (12-14) years.

-(15-17)years.

-(18) years or above.

4- The height of the driver(Short, Medium and Tall).

5- The height over the seat being distant:-

(0-12) cm, (13-20)cm, (30) cm or above.

6- The speed of the car.

The complete car type determination screen shot using E2glite shell, shown in

APPENDIX V.

Chapter Seven

Neural Network Representation

7.1 Definitions

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use [6]. A neural network can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information progressing units, called neurons [13]. The field neural networks has arisen from diverse sources, ranging from the fascination of mankind with understanding and emulating the human brain, to broader issues of copying human abilities such as speech and the use of language [28].

7.2 Neural Network Components

Within the neural systems there are three types of units (layers) as shown in figure 7.1³:-

- 1- Input layer: which receive data from outside of the network.
- 2- Hidden layer: which send data out of the network.
- 3- Output layer: whose input and output signals remain within the network [3].

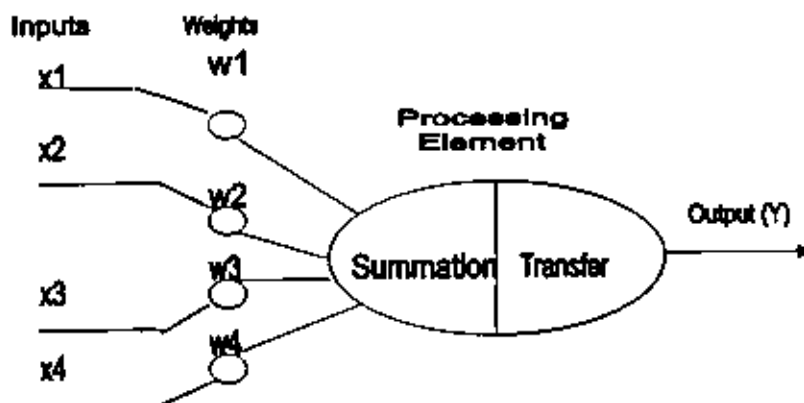


Figure 7.1 Neural network components

³ This figure is borrowed from [3].

7.3 Processing unit

A processing unit also called a neuron or node, performs a relatively simple job. it receives inputs from neighbors or external sources and uses them to compute an output signal that is propagated to other units as shown in figure 7.2. Each unit j can have one or more inputs $x_0, x_1, x_2, \dots, x_n$. but only one output z_j . An input to a unit is either the data from outside of the network, or the output of another unit, or its own output.

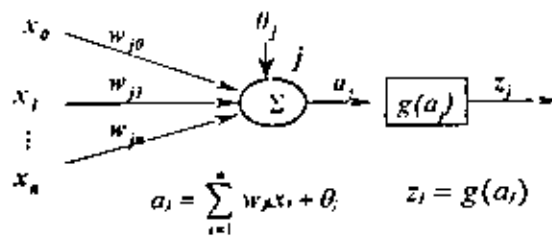


Figure 7.2 Processing unit

7.4 Network topologies

The topology of a network is defined by the number of layers, the number of units per layer, and the interconnection patterns between layers. They are generally divided into two categories based on the pattern of connections:

- 1) Feed-forward networks, where the data flow from input units to output units is strictly feed-forward, as shown in figure 7.3.
- 2) Recurrent networks, which contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important, as shown in figure 7.4.

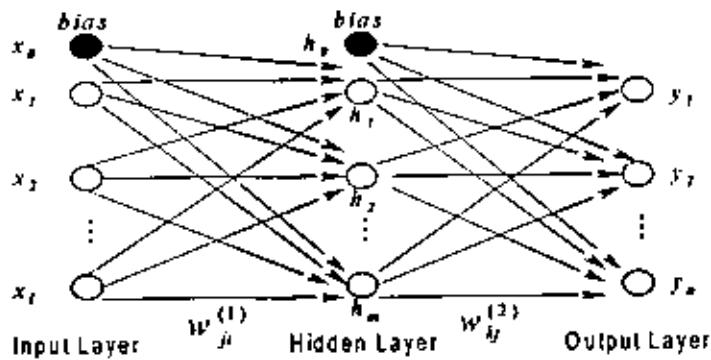


Figure 7.3 Feed-forward neural network

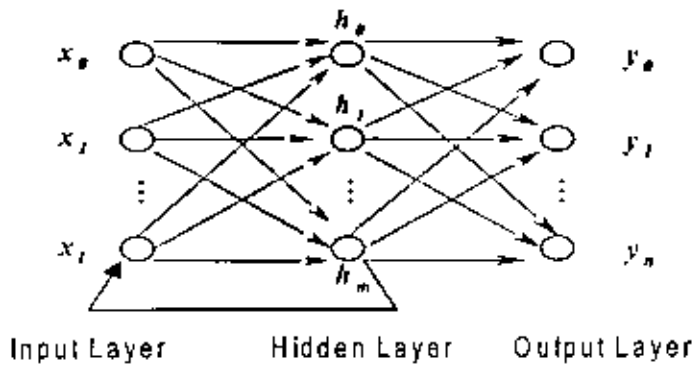


Figure 7.4 Recurrent neural network

7.5 Representation capability

The feed-forward networks provide a general framework for representing non-linear functional mapping between a set of input variables and a set of output variables. The representation capability of a network can be defined as the range of mappings it can implement when the weights are varied. Single-layer networks are capable of representing only linearly separable functions or linearly separable decision domains.

- Two hidden layered networks can represent an arbitrary decision boundary to arbitrary accuracy with threshold activation functions and could approximate any smooth mapping to any accuracy with sigmoid activation functions.
- One hidden layered network can approximate arbitrarily well any functional continuous mapping from one finite-dimensional space to another, provided that the number of hidden units is sufficiently large.

7.6 Network structure design

Though theoretically there exists a network that can simulate a problem to any accuracy, there is no easy way to find it. To define an exact network architecture such as how many hidden layers should be used, how many units should there be within a hidden layer for a certain problem is always a painful job.

7.7 Types of variables

Variables can be roughly divided into two categories:-

- **Categorical variables:** they do not have relationships like “greater than” or “less than”. Some of them come from some input values that do not have numerical values but have to transform to numerical values as input variables. For example, a variable called “color type”, which can take on the value “red”, “green”, and “yellow” is a categorical variable. Sex is a categorical variable to numerical data can also be categorical. Zip codes and telephone area codes are classic examples. In the above “color type” example, it requires three input variables, with the three colors represented by input values of (1,0,0), (0,1,0) and (0,0,1). Another way to encode categorical variables is to represent all the possible values to one continuous input variable. For example, the “red”, “green”, and “yellow” could be represented as 0.0, 0.5, and 1.0. The bad news for this method is that it imposes an artificial ordering on the data that does not exist. But for variables with a large number of categories, this can dramatically decrease the number of input units.
- **Ordinal variables:** have a natural ordering. Such data can be simply transformed directly into corresponding values of a continuous variable, either with or without scaling.

7.8 Knowledge Representation and Neural Networks

Neural networks represent a special class of intelligent machines. Typically, however, the possible forms of representation from the inputs to internal network parameters are highly diverse, which tends to make the development of a satisfactory solution by means of a neural network a real design challenge. A major task for a neural network is to learn a model of the world (environment), in which it is embedded and to maintain the model sufficiently consistent with the real world so as to achieve the specified goals of the application of interest. Knowledge of the world consists of two kinds of information:-

1. The known world state, represented by facts about what is and what has been known, this form of knowledge is referred to as prior information.
2. Observations (measurements) of the world, obtained by means of sensors designed to probe the environment in which the neural network is supposed to operate.

In a neural network of specified architecture, KR of the surrounding environment is defined by the values taken on by the free parameters (i.e., synaptic weights and biases) of the network. The form of this KR constitutes the very design of the neural network, and therefore holds the key to its performance.

❖ How to build information into neural network design

Unfortunately, there are currently no well-defined rules for doing this, rather one may use a combination of two techniques :-

1. Restricting the network architecture through the use of local connections known as receptive fields.
2. Constraining the choice of synaptic weights through the use of weight-sharing.

These two techniques, particularly the latter one, have a profitable side benefit, the number of free parameters in the network is reduced significantly. To satisfy the weight-sharing constraint, one merely have to use the same set of synaptic weights for each one of the neurons in the hidden layer of the network. Then, for the example shown in figure 7.5 with six local connections per hidden neuron and a total of four hidden neurons, one may express the induced local field of hidden neuron j as follows:

$$v_j = \sum_{i=1}^6 w_{i(i+j-1)} \quad j=1,2,3,4 \quad (7.1)$$

where w_i , $i=1$ to 6 constitute the same set of weights shared by all four hidden neurons, and X_k is the signal picked up from source node $k = i+j-1$ equation (7.1) is in the form of a convolution sum. It is for this reason that a feed forward network using

local connections and weight-sharing in the manner described herein is referred to as a convolution network.

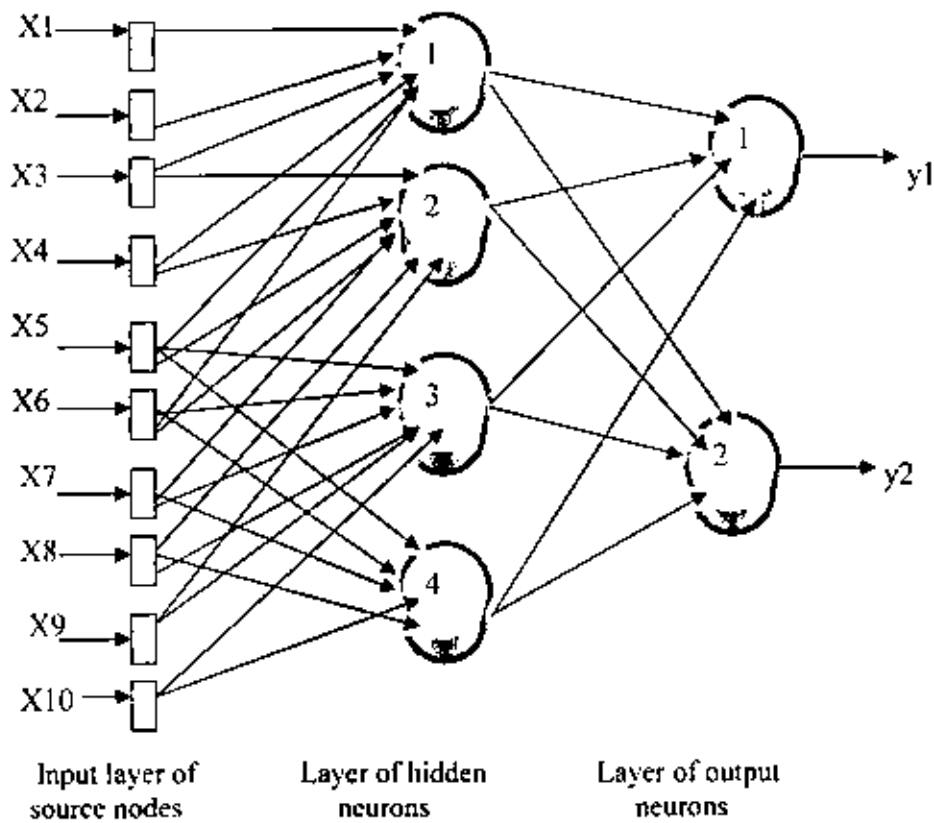


Figure 7.5 combined use of a receptive field and weight-sharing

❖ How to build invariance into neural network design

There are at least three techniques for rendering classifier-type neural networks invariant to transformations :

- **Invariance by structure:-** invariance may be imposed on a neural network structuring its design appropriately. Specifically, synaptic connections between the neurons of the network are created so that transformed versions of the same input are forced to produce the same output.
- **Invariance by training:-** the network is trained by presenting it a number of different examples of the same object, with the examples being chosen to correspond to different transformations (i.e different aspect views) of the object.

- Invariant feature space:- it rests on the premise that it may be possible to extract features that characterize the essential information content of an input data set, and which are invariant to transformations of the input, is illustrated in figure 7.6.

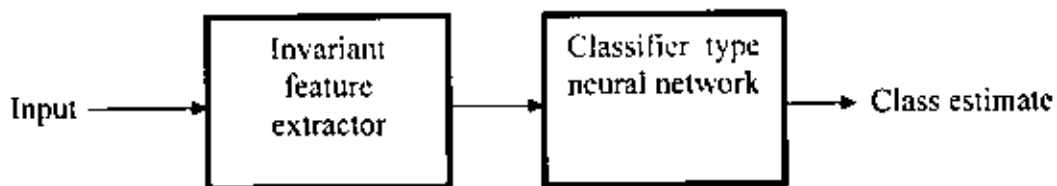


Figure 7.6 Block diagram of an invariant feature-space type of system

7.9 Advantages of Neural Network

- It is computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize [6].
- It provides a means of capturing truly complex behavior in a highly hierarchical fashion.
- It is possible to share theories and learning algorithms in different applications of neural networks.
- Modular networks can be built through a seamless integration of modules.
- The neural network not only provides the implicit model of the environment in which it is embedded, but also performs the information-processing function of interest.
- The number of features applied to the network may be reduced to realistic levels.
- The requirements imposed on network design are relaxed.
- Invariance for all objects with respect to known transformations is assured.
- An artificial neuron can be linear or nonlinear a neural network.
- Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.

- The design of a neural network is motivated by analogy with the brain, which is a living proof that fault tolerant parallel processing is not only physically possible but also fast and powerful. [2].

7.10 Disadvantages of Neural Network

- Similar inputs from similar classes should usually produce similar representations inside the network, and should therefore be classified as belonging to the same category.
- Items to be categorized as separate classes should be given widely different representations in the network.
- If a particular feature is important, then there should be a large number of neurons involved in the representation of that item in the network [6].

Example:- Digit recognition

This case study illustrated one of the most common applications of multiplayer neural networks. Modern character recognition systems are capable of processing different fonts. Optical character recognition is routinely used by office workers, lawyers, insurance clerks, journalists. In this application, each digit is presented by 5x7 bit map. The architecture and size of a neural network depend on the complexity of the problem. For the printed digit recognition problem, a three-layer network with a single hidden layer will give sufficient accuracy. The number of neurons in the input layer is decided by the number of pixels in the bit map. The bit map in this example consist of 35 pixels , and thus a person need 35 input neurons. The output layer has 10 neurons, one neuron for each digit to be recognized. The complete screen shot shown in APPENDIX VI.

Chapter 8

Genetic algorithms representation

8.1 Definitions

Genetic algorithms (GA) are a class of stochastic search algorithms based on biological evolution. Given a clearly defined problem to be solved and a binary string representation for candidate solutions[13]. The GA is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population on the principles of genetics and natural selection. The software utilizes three approaches to the optimization problem. First, examining areas of the solution space that an expert solving the problem would know to avoid. Second, finding a local optimum in the solution space. However, once this local high altitude point has been reached, this method can proceed no further. Third, assigns a genetic string made up of ones and zeros to each local optimum solution provided by the gradient approximation technique [11].

There are many variations of algorithms that could be classified as genetic algorithms. However, the following three operations are standard:-

- 1) Evaluation of individuals in the population,
- 2) Formation of a gene pool.
- 3) Recombination and mutation? [16].

8.2 Elements of Genetic Algorithms

Element of GA is rarely used in the chromosomes in a GA population typically takes the form of bit strings. Each locus in the chromosome has two possible alleles: 0 and 1. Each chromosome can be thought of as a point in the search space of candidate solutions. The GA processes populations of chromosomes, successively replacing one such population with another. The GA most often requires a fitness function that assigns a score (fitness)

to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand [17]. A basic GA can be represented as in figure 8.1, a GA applies the following major steps :-

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N , the crossover probability (p_c) and the mutation probability (p_m).

Step 2: Define a fitness function to measure the performance or fitness of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3: Randomly generate an initial population of chromosomes of size N : x_1, x_2, \dots, x_n .

Step 4: Calculate the fitness of each individual chromosome: $f(x_1), f(x_2), \dots, f(x_n)$

Step 5: Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness. Highly fit chromosomes have a higher probability of being selected for mating than less fit chromosomes.

Step 6: Create a pair of offspring chromosomes by applying the genetic operators crossover and mutation.

Step 7: Place the created offspring chromosomes in the new population.

Step 8: Repeat step 5 until the size of the new chromosome population becomes equal to the size of the initial population, n .

Step 9: Replace the initial (parent) chromosome population with the new (offspring) population.

Step 10: Go to step 4, and repeat the process until the termination criterion is satisfied

{13}.

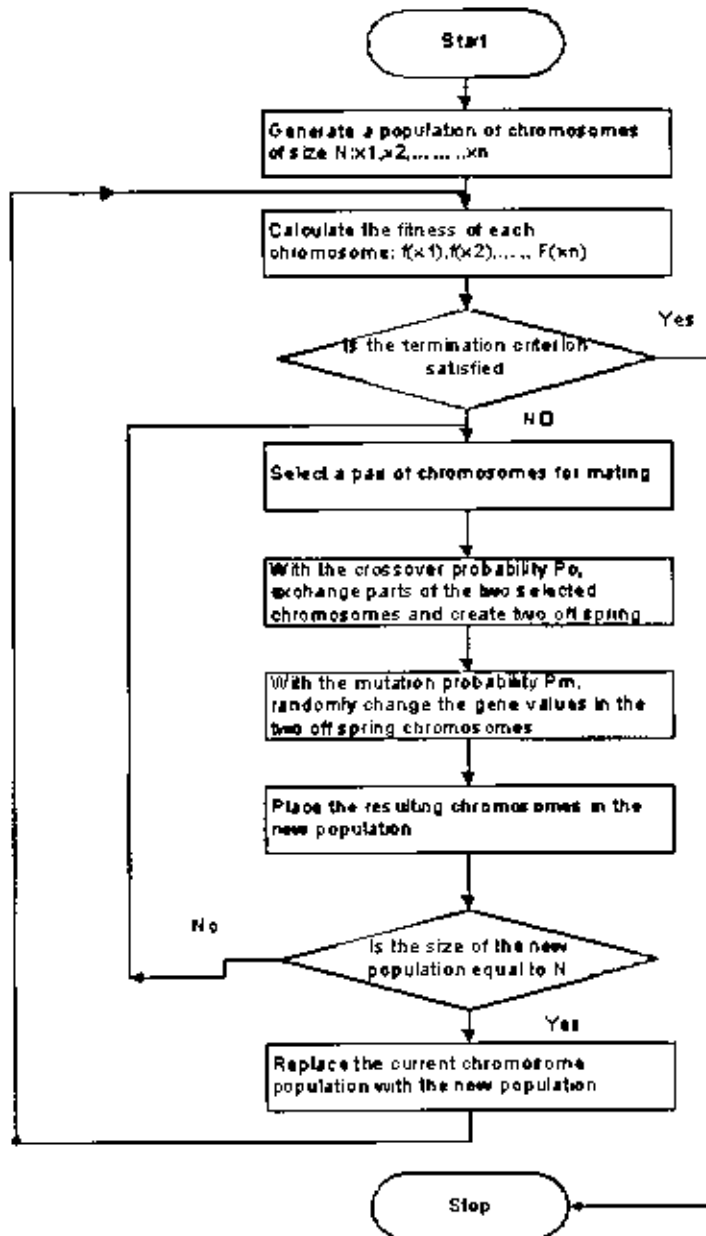


Figure 8.1 basic genetic algorithm

8.3 Genetic Algorithms Operators

The simplest form of GA involves three types of operators: selection, crossover (single point) and mutation.

Selection :- This operator selects chromosomes in the population for reproduction.

Crossover :- This operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. For example, the strings 10000100 and 11111111 could be crossed over after the third locus

in each to produce the two offspring 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single-chromosome (haploid) organisms.

Mutation:- This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001)[17].

Example: Maintenance scheduling with genetic algorithms.

The problem one discuss here is the maintenance scheduling in Modern power system, this task has to be carried out under several constraints and uncertainties, such as failures and forced outages of power equipment and delays in obtaining spare parts. The schedule often has be revised at short notice. Human expert system usually works out the maintenance scheduling by hand and there is no guarantee that the optimum or even near optimum schedule is produced.

Step 1: specify the problem, define constraints and optimum criteria.

Power system components are made to operate continuously throughout their life by means of preventive maintenance. The purpose of maintenance scheduling is to find the sequence of outages of power units over a given period of time (normally a year) such that the security of power system is maximized. Any outage in a power system is associated with some loss in security. The security margin is determined by the systems net reserve. The net reserve, in turn, is defined as the total installed generating capacity of the system minus the power lost due to a scheduled outage and minus the maximum load forecast during the maintenance period. For instance, if one assumes that the total installed capacity is 150MW and a unit of 20MW is scheduled for maintenance during the period when the maximum load is predicted to be 100MW, the net reserve will be 30MW. Maintenance scheduling must ensure that sufficient net reserve is provided for

secure power supply during any maintenance period. Suppose, there are seven power units to be maintained in four equal intervals, the maximum loads expected during these intervals are 80,90,65 and 70 MW, the unit capacities and their maintenance requirements are presented in table 8.1.

Unit number	Unit capacity MW	Number of intervals required for unit maintenance during one year
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

Table 8.1 power units and their maintenance requirements

The optimum criterion here is that the net reserve must be at the maximum during any maintenance period.

Step 2: Represent the problem domain as a chromosome. Our scheduling problem is essentially an ordering problem, requiring us to list the tasks in a particular order. Ours is to represent a complete schedule as a chromosome of a fixed length. A coding scheme is to assign each unit a binary number and to let the chromosome be a sequence of these binary numbers. A chromosome is a collection of elementary parts called genes. Traditionally, each gene is represented by only one bit and cannot be broken into smaller elements, for this problem one can adopt the same concept, but represent a gene by four bits as shown in table 8.2. In other words, the smallest indivisible part of our chromosome is a 4-bit string. This representation allows crossover and mutation operators to act according to the theoretical grounding of genetic algorithms. What remains to be done is to produce a pool of gene for each unit:-

Unit 1	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 2	1 1 0 0	0 1 1 0	0 0 1 1	
Unit 3	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 4	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 5	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 6	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
Unit 7	1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1

Table 8.2 Represent a gene by Four bits

The GA can now create an initial population of chromosomes by filling 7-gens chromosomes with genes randomly selected from the corresponding pools. A sample of such a chromosome is shown in figure 8.2.

Step3:- Define a fitness function to evaluate the chromosomes performance. The fitness function must capture what makes a maintenance schedule either good or bad for the user. In this problem, apply a fairly simple function concerned with constraint violation and the net reserve at each interval.

Unit1	Unit2	Unit3	Unit4	Unit5	Unit6	Unit7
0 1 1 0	0 0 1 1	0 0 0 1	1 0 0 0	0 1 0 1	0 0 1 0	1 0 0 0

Figure 8.2 A chromosome for the scheduling problem

The evaluation of a chromosome starts with the sum of capacities of the units scheduled for maintenance at each interval for the chromosome shown in figure 8.2, one obtains:-

$$\text{Interval 1: } 0 \times 20 + 0 \times 15 + 0 \times 35 + 1 \times 40 + 0 \times 15 + 0 \times 15 + 1 \times 10 = 50$$

$$\text{Interval 2: } 1 \times 20 + 0 \times 15 + 0 \times 35 + 0 \times 40 + 1 \times 15 + 0 \times 15 + 0 \times 10 = 35$$

$$\text{Interval 3: } 1 \times 20 + 1 \times 15 + 0 \times 35 + 0 \times 40 + 0 \times 15 + 1 \times 15 + 0 \times 10 = 50$$

$$\text{Interval 4: } 0 \times 20 + 1 \times 15 + 1 \times 35 + 0 \times 40 + 0 \times 15 + 0 \times 15 + 0 \times 10 = 50$$

Then these values subtract from total installed capacity of the power system (in our case, 150MW)

$$\text{Interval 1: } 150 - 50 = 100$$

$$\text{Interval 2: } 150 - 35 = 115$$

$$\text{Interval 3: } 150 - 50 = 100$$

Interval 4: $150-50=100$

And finally, by subtracting the maximum loads expected at each interval, we obtain the respective net reserves:-

Interval 1: $100-80=20$

Interval 2: $115-90=25$

Interval 3: $100-65=35$

Interval 4: $100-70=30$

Since all the result are positive, this particular chromosome does not violate any constraints, and thus represents a legal schedule. The chromosomes fitness is determined as the lowest of the net reserves; in our case it is 20. If , however, the net reserve is negative the schedule is illegal, and the fitness function returns zero [13].

8.4 Advantages of Genetic Algorithms

- The ability to find near optimal, as opposed to optimal, solutions is an important distinction. For some applications, the optimal solution is required.
- It has already been used in a wide variety of applications.
- GA are powerful tools for solving problems and for simulating natural systems in a wide variety of scientific fields.
- Can be promising methods for solving difficult technological problems, and for machine learning.
- It will be part of a new movement in computer science that is exploring biologically inspired approaches to computation.
- It is able to deal with complexity, able to learn.
- It will promise approaches for modeling the natural systems that inspired their design [16].

- It can allow scientists to perform experiments that would not be possible in the real world, and to simulate phenomena that are difficult or impossible to capture and analyze it [15].

8.5 Disadvantages of Genetic Algorithms

- Intriguing and producing stunning results when traditional optimization approaches fail miserably.
- The optimizer should use the experience of the past and employ these quick methods.
- Many realistic problems do not fall into this category.
- For problems that are not overly difficult, other methods may find the solution faster than the GA.
- The large population of solutions that gives the GA its power is also its bane when it comes to speed on a serial computer.

Example :-Resolve Equation

This study manipulates the data and assign the fitness value for the equation $x^2+y^2+z^2=?$ by GA and visual basic language. To find x, y and z such that $x^2+y^2+z^2=90$. The screen shot solution of equation illustrated in APPENDIX VII.

Chapter 9

Comparison of different representation

To use knowledge in real applications, a way of representing must be chosen. Also, need a notation that supports what we expect an expert system to do. The notation should make it easy for us to add and change knowledge, should be easy for us to read, and should support explanation generation. In addition, the notation should suggest ways in which it will be used and should allow us to write down different methods of use. The notation should also encourage us to separate declarative knowledge from procedural knowledge, yet it should support efficient problem solving.

9.1 Criteria of Comparison

- ❖ Variables.
- ❖ Statement/Relations.
- ❖ Programming.
- ❖ Reasoning technique.

Thus one can use these criteria to compare the various KR :-

1. Rules Representation

Criteria of Comparison	Description
Variables	Can be used as arguments in a rule. It is possible to choose one of the computer languages or available shells that offer powerful pattern-matching rules to match similar problem statements.
Relations	The rule has opaque rule relationships, it is often difficult to determine
Programming	<ul style="list-style-type: none"> • Easy to programming, They allow us to view through the syntax to the meaning and easy for humans to read and understand . • May be changed without affecting other parts of knowledge base . • It can be slow: it must scan the entire set of rules.
Reasoning	It is deductive reasoning techniques because the process begins with comparing the axioms with a set of implications to conclude new axioms.

Table 9.1 Criteria of comparison for rules representation

2. Logic Representation

Criteria of Comparison	Description
Variables	Are written as symbols beginning with an uppercase letter. In predicate calculus, variables can be used as arguments in a predicate expression or a function.
Relations	Are expressed with clauses and rules.
Programming	Has some specifics such as:- <ul style="list-style-type: none"> • Easy to programming, but if the knowledge base is large, the processing efficiency decreases . • It gives precise result, allows programs to be written which are declarative.
Reasoning	It is inductive reasoning techniques because the process begins with comparing the axioms with a sure group facts or logically related known information.

Table 9.2 Criteria of comparison for logic representation

3. Semantic Networks Representation

Criteria of Comparison	Description
Variables	Represented as a value or clause in node.
Relations	IS-A and AKO relationship relates between the nodes. The arcs represent the relationships between the nodes.
Programming	It is difficult to implement and difficult for programming
Reasoning	It is logical reasoning techniques because requires the ability to infer conclusions from available facts.

Table 9.3 Criteria of comparison for semantic networks representation

4. Knowledge Representation with Frames

Criteria of Comparison	Description
Variables	Represented as a value or clause in slot of object.
Relations	IS-A and AKO relationship relates between slots and objects. a relationships are expressed in terms of either slot values or inheritance.
Programming	The available software's are high-priced but they are easy to implement by high level languages.
Reasoning	It is logical reasoning techniques because requires the ability to infer conclusions from available facts

Table 9.4 Criteria of comparison for knowledge representation with frames

5. Fuzzy Logic Representation

Criteria of Comparison	Description
Variables	The sets in fuzzy logic are represented by linguistic variables. several linguistic variables might be involved in the antecedents and the conclusions of these rules.
Relations	Are expressed with clauses and rules.
Programming	Allows programs to be written which are declarative. they describe what is true and not how to solve problems by a matter of degree.
Reasoning	The exact reasoning is viewed as a limiting case of approximate reasoning.

Table 9.5 Criteria of comparison for fuzzy logic representation

6. Neural Network Representation

Criteria of Comparison	Description
Variables	Can be roughly divided into two categories:- Categorical variables and ordinal variables.
Relations	They do not have relationships like "greater than" or "less than". Some variables come from some input values that do not have numerical values but have to transform to numerical values as input variables.
Programming	A programming is difficult. one neural network project that met some marginal success was called the perception. Designed for character recognition.
Reasoning	A neural network can be as a model of reasoning based on the human brain.

Table 9.6 Criteria of comparison for neural network representation

7. Genetic Algorithm Representation

Criteria of Comparison	Description
Variables	Represents as a chromosome of a fixed length.
Relations	Relationship is a chromosome matches a schema when the fixed position in the schema match the corresponding position in the chromosome.
Programming	A programming of GA depends on a Darwinian theory of evolution via a programming structure.
Reasoning	The exact reasoning is viewed as a limiting case of approximate reasoning.

Table 9.7 Criteria of comparison for genetic algorithm representation

9.2 Conclusions

This work indicate that symbolic knowledge can be represented in several ways including rules, logic, semantic network, frames, fuzzy logic, neural network and genetic algorithms.

Rules allow us to see through the syntax to the meaning and easy for humans to read and understand and can change without affecting other parts of knowledge base and

allow us to quickly create prototypes to test ideas and prove feasibility. But it is difficult to represent algorithms and not allowed to call another rules directly.

Logic provides a natural common notation, also logic in the numeric case is a strong candidate notation, it allows inference to establish new relationships from old, it is precise and it allows programs to be written which are declarative. On the other hand it is not good way for organizing knowledge because the processing is not efficient.

Semantic networks provide a simple, economical, and relatively intuitive representation form, provides a graphical view of a problems. But it only deals with small pieces, and representing procedural knowledge is a big problem and complexity grows exponentially in nets with many relationships.

Frames extend semantic networks and it easier to organize our knowledge hierarchically. Frames can be reused in different application with minimum modification and implement semantic networks. But maintenance is sometimes difficult with relationships among frames are complex .

Fuzzy logic It is more general, it is mimics the way in which humans interpret linguistic values and the transition from one linguistic value to a contiguous linguistic value is gradual rather than abrupt, resulting in continuity and robustness.

Neural network it provides a means of capturing truly complex behavior in a highly hierarchical fashion, capable of robust computation. Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment and the design of a neural network is motivated by analogy with the brain, which is a living proof that fault tolerant parallel processing is not only physically possible but also fast and powerful. On the other hand, the similar inputs from similar classes should usually produce similar representations inside the network.

Genetic algorithm has an ability to find near- optimal, as opposed to optimal, solutions is an important distinction. For some applications, the optimal solution is required, GA

are powerful tool for solving problems and for simulating natural systems in a wide variety of scientific fields. On the other hand, the large population of solutions that gives the GA its power is also its bane when it comes to speed on a serial computer.

The combination of frames and logic is well suited to the task of building expert systems. Other advantage of linking frames and logic is that the frame system can be customized to taste or to the demands of a particular application by writing additional predicates if an alternative syntax was preferred .

The combination of genetic algorithms and neural networks (neuron genetic) is well suited to the task of building expert systems . While genetic algorithms are a powerful form of knowledge representation in generate genus and the best result after many iteration and a neural network is a powerful form of recognition knowledge representation and direct way of controlling the reasoning process. Thus the combination of neural networks and genetic algorithms function effectively when the fundamental knowledge structures are built with neural networks and genetic algorithms.

After this comparison the conclusion is that there is no method better than the other. It depends on the problem domain and the task that is to be performed, so each method is valid for certain application. If the application is:-

- Classified : using semantic networks or frame.
- Conditional with clear inputs: using rules or logic .
- Conditional with non-clear inputs: using fuzzy logic .
- Recognition with incomplete (a proximately)inputs: using neural networks.
- If the space to be searched is large or the fitness function is noisy or the space is smooth or the space is well understood: using genetic algorithms.

References

1. Stuart Russell and Peter Norvig. (2003). Artificial Intelligence a modern approach second edition, pearson education.
2. John Durkin. (1994) .Expert systems design and development, prentice-hall.
3. Mohamed F. Tolba and Other.(1998). Computer and artificial intelligence, delta.
4. Zean Abdulhadi.(200). Artificial Intelligence and expert system in libraries, academic library.
5. Thomas Dean and Other.(1995). Artificial intelligence Theory And practice, Addison wesly.
6. Simon Haykin.(1999). Neural networks comprehensive foundation, prentice-hall.
7. James j. Buckley and Esfandiar Eslami.(2002). An introduction to fuzzy logic and fuzzy sets, physica verlag.
8. George Luger.(2002). Artificial intelligence, structured and strategies for complex problem solving fourth edition, pearson education limited.
9. Ziad Alqadi and Rashad Rasras.(2003). Artificial intelligence, arab community library.
10. George Luger. Artificial intelligence.(2005). Structured and strategies for complex problem solving fifth edition, Addison wesly .
11. Nikolas G. Bourbakis.(1998). Artificial intelligence and automation, world scientific publishing co.
12. Sankar k. Pal and Sushmita Mitra.(1999). Neuro fuzzy pattern recognition Methods in soft computing, John wiley&sonc inc.
13. Michael negnevitsky.(2002). Artificial intelligence, a guide to intelligent

system ,addison wesly.

14. Francis Rickett.(1999). The Genetic Algorithm in E-Commerce Applications.
15. Melanie Mitchell. (1999). An introduction to genetic algorithms Massachusetts institute of technology.
16. D.E.O'leary and Paul R.Watkins.(1999).Expert systems in finance, elsevier science publishing company inc.
17. Kenneth D. Forbus.(1996). Introduction to expert systems, second edition , handbook of computer.
18. Paulo Quaresma and Irene Pimenta Rodrigues. (1999). Using dynamic logic programming to model cooperative dialogues.
19. Ora Lassila & Deborah McGuinness.(2002). the role of frame-based representation on the Semantic Web.
20. Christian Freksa.(1994). Fuzzy Systems in artificial intelligence, Braunschweig Wiesbaden.
21. Barry G. Silverman.(1987). Expert systems for business, Addison wesley.
22. Christopher Brewster.(2003). Knowledge Maintenance and the Frame Problem, University of Sheffield.
23. Paul Vincent.(2005). Structured rules language a commercial rule representation. fair isaac.
24. Robert j. Schalkoff.(1983). Artificial intelligence an engineering approach ,mcgraw hill.
25. Stephen G. Simpson.(2006). subsystem of second order arithmetic, second edition.
26. Neural networks versus AI the phone war.(2003).
27. Peter Jackson.(1990). Introduction to expert systems, Addison wesley.

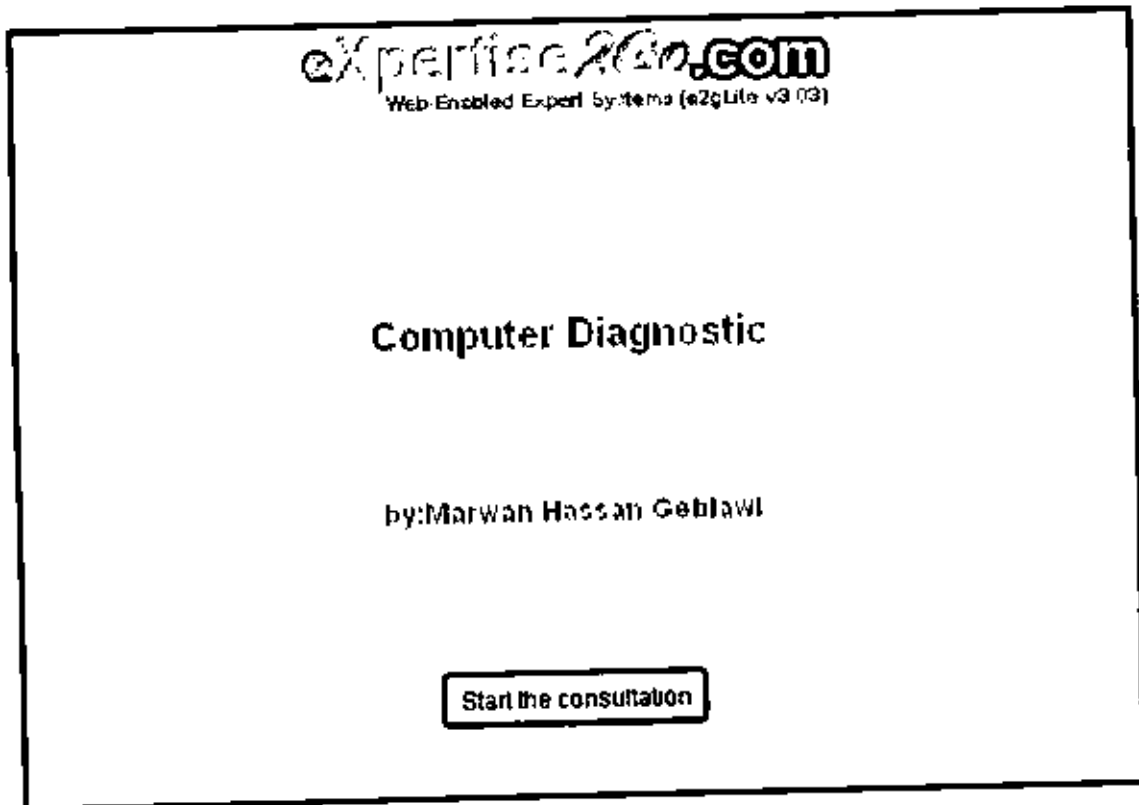
28. D. michie and D. J. Spiegelhalter.(1994). Machine learning neural and statistical classification , C.C taylor.
29. George G. Savii.(1998). Preliminary computer aided design using fuzzy logic.

Computer Diagnostics

Introduction to E2glite shell

E2glite is a Java applet that is embedded in a Web page. The applet loads a knowledge base from the server and then runs entirely on the browser. The Computer diagnostic screen shot illustrated as blew:-

1- Startup E2glite



2- Start the consultation

What is the kind of problem?

See Error Message Display on fatal errors
 Display error messages for Computer hardware problems
 BIOS error codes
 BIOS Audio error Codes (BIOS Error beep/fatal errors)
 problems of computer and its accessories
 viruses of computer
 I don't know would like to get answer

Very uncertain (50%) Very certain (100%)

Submit your response | Why ask? | Restart

3- Make selection (such as: dos error codes)

What is the kind of problem

- Dos Error Message (display/unfatal errors)
- Display error messages for computer hardware's problems
- DOS error Codes
- DOS Audio error Codes(BIOS Error beep/fatal errors)
- problems of computer and its accessories
- viruses of computer
- I don't know/would rather not answer

Very uncertain (50%) Very certain (100%)

Submit your response | Why ask? | Restart

4- When select choice (submit your response)

The DOS error Codes is

I don't know/would rather not answer

Very uncertain (50%) Very certain (100%)

Submit your response | Why ask? | Restart

5- Press  for view choices "dos error codes"

The DOS error Codes is

I don't know/would rather not answer

- mother board error Codes(from code 100 up to code 200)
- memory(RAM) error Codes(from code 200 up to code 300)
- Keyboard error Codes(from code 300 up to code 400)
- Display or Video error Codes(from code 400 up to code 700)
- Floppy disk drive error Codes(from code 700 up to code 7328)
- Math Coprocessor error Codes(from code 700 up to code 800)
- Parallel&Serial printer adapter error Codes
- printers errors codes

100%

6- After choice dos error code press (submit your response)

The DOS error Codes is

Keyboard error Codes(from code 300 up to code 400)

Very uncertain (50%) Very certain (100%)

Submit your response | Why ask? | Restart

7- This screen contains "keyboard errors codes"

The Keyboard error Codes (from Code 300 up to code 400) is

(301)
 (302)
 (303)
 (312)
 (313)
 I don't know/would it be not answer?

Very uncertain (55%) Very certain (100%)

Submit your response | Why ask? | Restart

8- Press on a code requirement

The Keyboard error Codes (from code 300 up to code 400) is

(301)
 (302)
 (303)
 (312)
 (313)
 I don't know/would it be not answer?

Very uncertain (50%) Very certain (100%)

Submit your response | Why ask? | Restart


9- Press on submit your response (this is a final result)

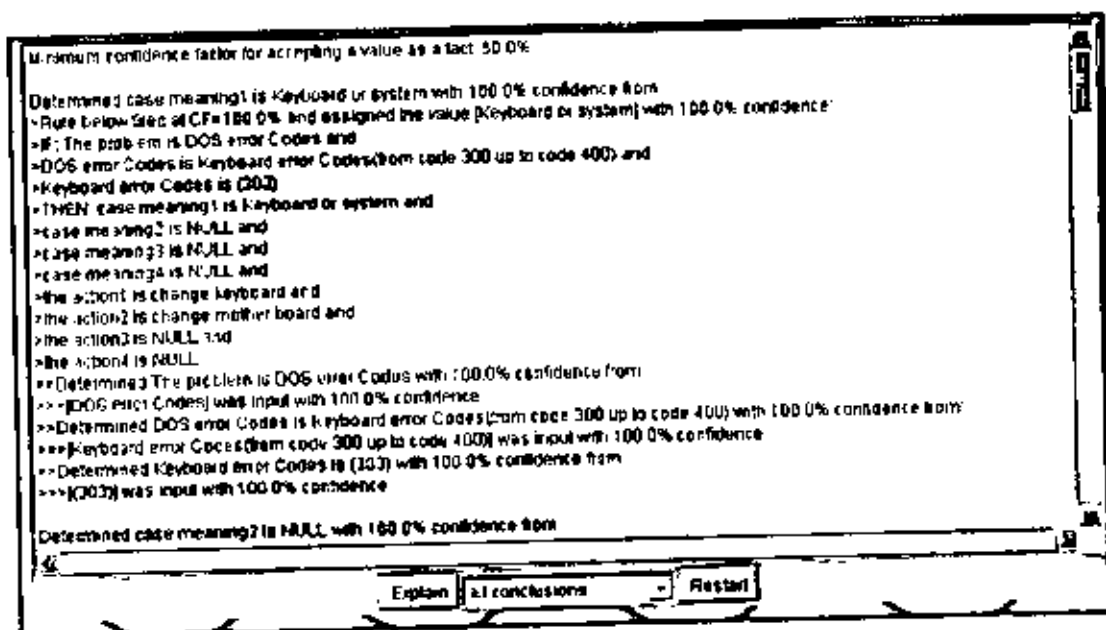
FINAL RESULTS:

Value 1 of case meaning1 is: Keyboard or system (100.0% confidence)
 Value 1 of case meaning2 is: NULL (100.0% confidence)
 Value 1 of case meaning3 is: NULL (100.0% confidence)
 Value 1 of case meaning4 is: NULL (100.0% confidence)
 Value 1 of the action1 is: change keyboard (100.0% confidence)
 Value 1 of the action2 is: change mother board (100.0% confidence)
 Value 1 of the action3 is: NULL (100.0% confidence)
 Value 1 of the action4 is: NULL (100.0% confidence)

Explain all conclusions | Restart

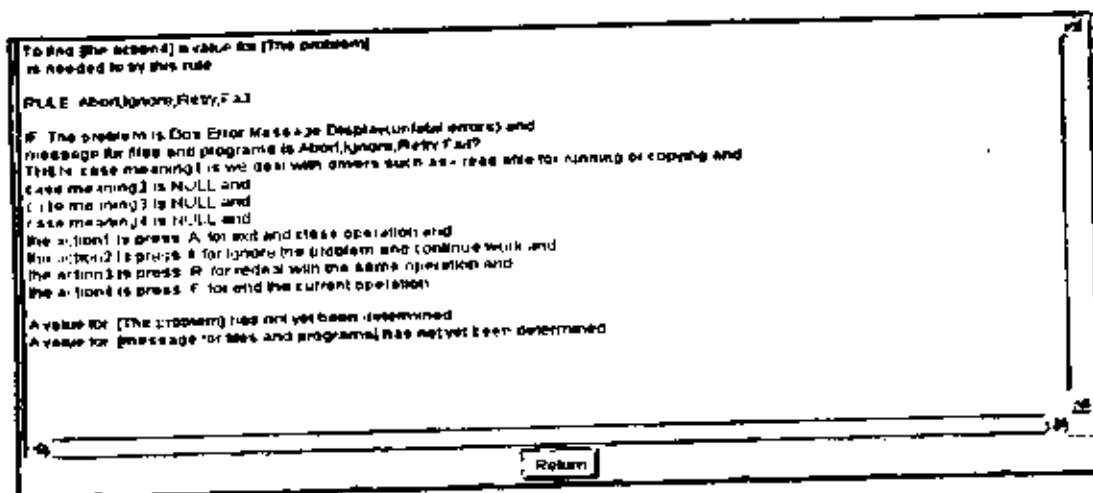
Remarks:-

- 1- If I want to view more details about any part from the total parts of conclusion this result (press  and press Explain). If I chosen the choice "all conclusion").



- 2- Choice "Restart" in any menu for restart from the beginning of Shell.

- 3- Press "way ask" in any menu for represent explain the action.



- 4- Choice return :- Return the execution to the previous menu.

1- Startup E2glite

XperTise .COM

COMPUTER WOULD START

[Start the consultation]

2- Electrical not dead → computer will start; implication T

case of electrical

dead
 not dead
 i don't know / would rather not answer

[Submit your response | Why ask? | Restart]

the computer case

will start
 won't start
 i don't know / would rather not answer

[Submit your response | Why ask? | Restart]

FINAL RESULTS

Value 1 or Implication is: T computer starting (100.0% confidence)

[Explain | all conclusions | Restart]

3- Electrical not dead → computer won't start; implication T

case of electrical

- dead
- not dead
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

the computer case

- will start
- won't start
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

FINAL RESULTS:

Value 1 of Implication is: T computer starting (100.0% confidence)

Explain | all conclusions | Restart

4- Electrical is dead → computer will start; implication F

case of electrical

- dead
- not dead
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

the computer case

- will start
- won't start
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

FINAL RESULTS:

Value 1 of Implicabon is: F computer not starting (100.0% confidence)

Explain | all conclusions ▾ | Restart

5- Electrical is dead → computer wont start; implication T

case of electrical

- dead
- not dead
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

the computer case

- will start
- wont start
- I don't know/would rather not answer

Submit your response | Why ask? | Restart

FINAL RESULTS:

Value 1 of Implication is: T computerstarting (100.0% confidence)

Explain all conclusions Restart

Elephant Inheritance

The computer language using for Elephant's inheritance in this case study named Prolog. prolog is a computer programming language that is used for solving problems that involve objects and the relationships

```

SWI-Prolog (Multi Threaded, version 5.6.0)
[File Edit Settings Run Debug Help]
Welcome to SWI-Prolog (Multi-threaded, Version 5.6.0)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic), or ?- apropos(Word).

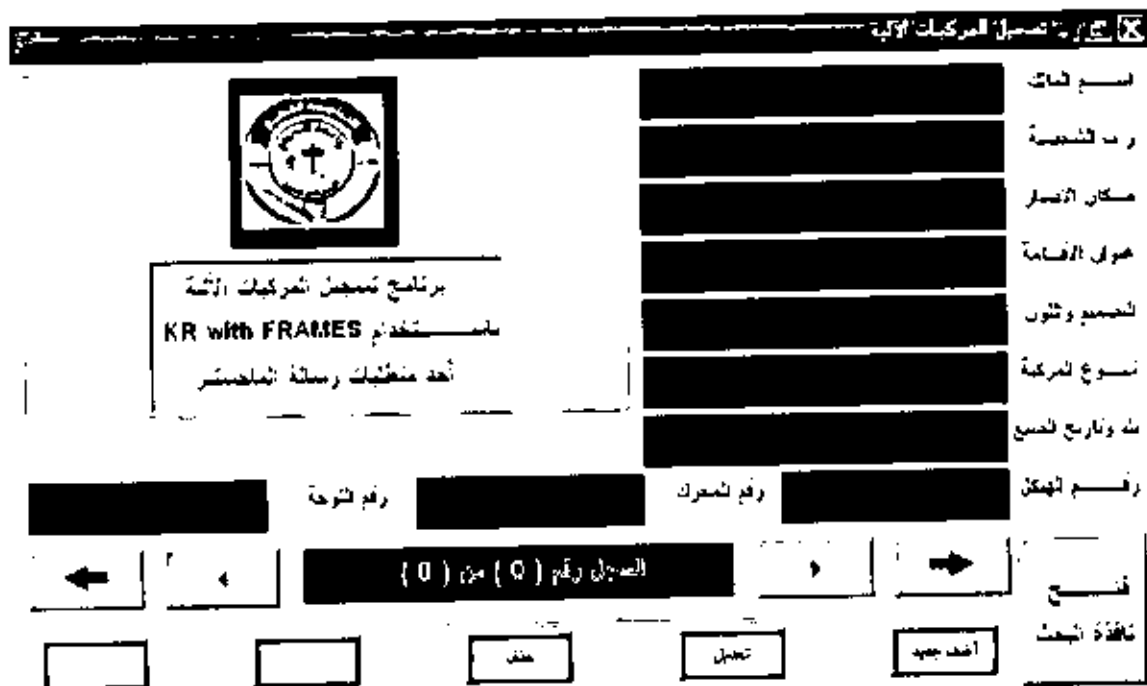
1 ?-
2 a c:/Documents and Settings/1/Desktop/case study lamb program/inheritance elephant by semantic
network/semantic1.pl compiled 0.00 sec, 4,764 bytes
1 ?- satisfies(hasname, e1, N).
N = winifand
Yes
2 ?- satisfies(legs, e1, N).
N = 3
Yes
3 ?- satisfies(isa, e1, elephant).
Yes
4 ?- satisfies(legs, elephant, N).
N = 4
Yes
5 ?- satisfies(skin, e1, Colour).
Colour = grey
Yes
6 ?- satisfies(lactates, e1, P).
P = yes
Yes
7 ?- satisfies(mammal, e1, yes).
Yes
8 ?-
  
```

Try the following queries:

- ?- satisfies(hasname, e1, N). % check an unambiguous fact about e1
- ?- satisfies(legs, e1, N). % check e1 has 3 legs, not 4
- ?- satisfies(isa, e1, elephant). % check e1 isa elephant
- ?- satisfies(legs, elephant, N). % normal elephants have 4 legs
- ?- satisfies(skin, e1, Colour). % 2 steps: e1 isa elephant has grey skin
- ?- satisfies(lactates, e1, Ans). % e1 isa elephant ako mammal lactates yes

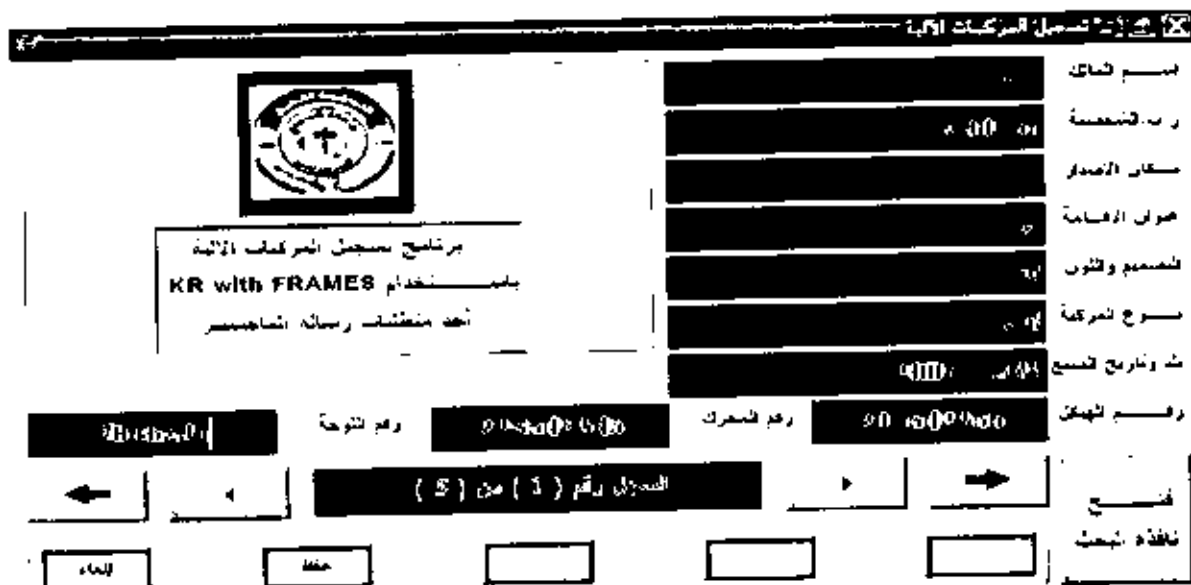
Car registration in Traffic Office

The complete screen shot for manipulate tasks shown below:-



هذا البرنامج أخذ منطقات رسالة ماجستير بعنوان 'Knowledge Representation Schemes'

- 1- To add slot press button of (add new one)



البرنامج أخذ منطقات رسالة ماجستير بعنوان 'Knowledge Representation Schemes'.

2- To edit slot press button of edit

هذا البرنامج أحد منظمات رسالة ماجستير بعنوان 'wledge Representation Scemes'

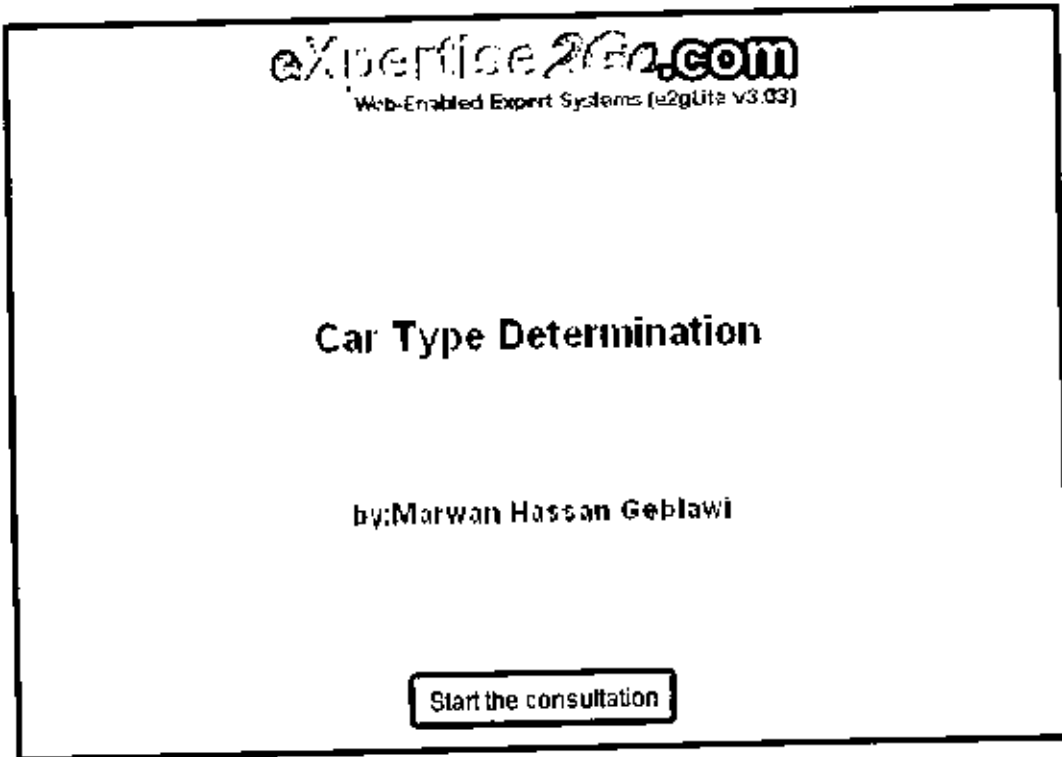
3- To delete slot press delete button

- select the slot to be deleted and press delete button

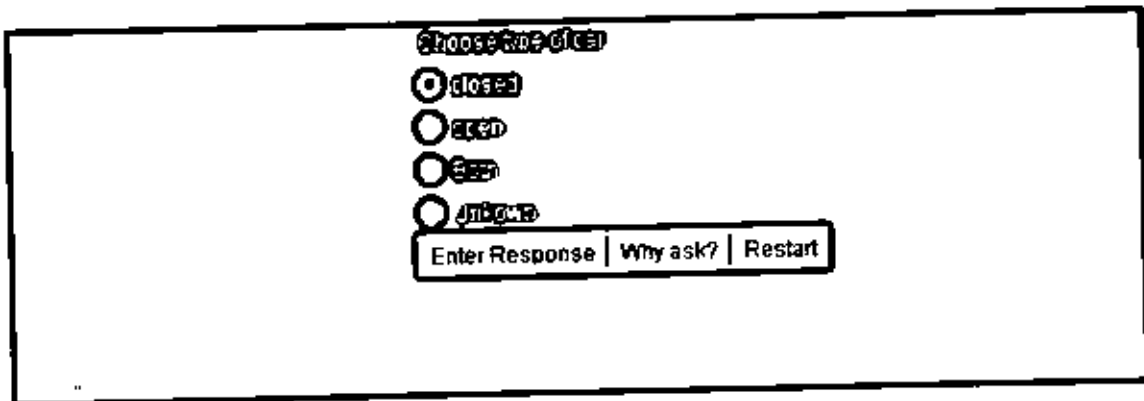
YES: means you are sure to delete, NO : means you are not sure to delete.

Car Type Determination

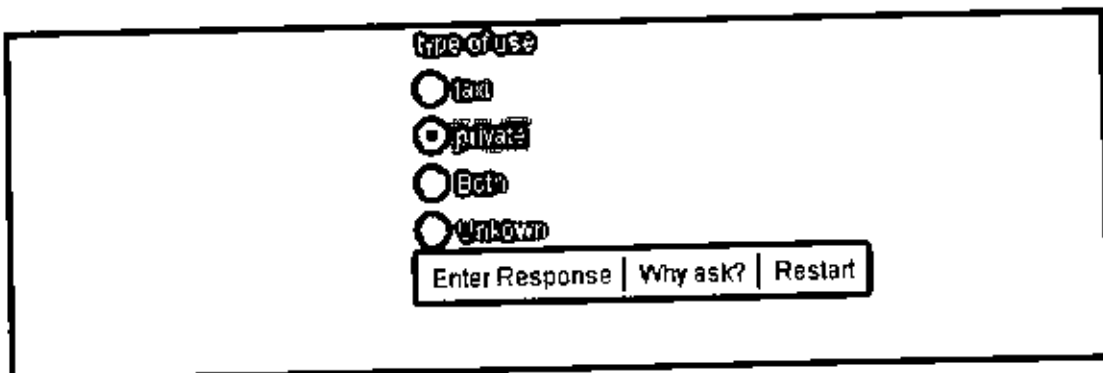
1- Startup E2glite



2- Make selection (such as: closed)



3- When select choice (Private)



4- When select choice (Ages of the driver)

select Age

12-14

16-17

18-

Unknown

Enter Response | Why ask? | Restart

5- When select choice tall of the driver

tall of the driver

Small

Medium

Tall

Unknown

Enter Response | Why ask? | Restart

6- When select choice(distance driver over set)

how much the distance driver over set

0-13

13-26

30-

Unknown

Enter Response | Why ask? | Restart

7- When select choice(speed of the car)

the speed of the car in km/hr

100+

60-100

0-60

Unknown

Enter Response | Why ask? | Restart

8- Press on Enter response (this is a final result)

FINAL RESULTS

- Value 1 of Car type is: lancer (69.75% confidence)
- Value 2 of Car type is: dawoo (39.5% confidence)
- Value 3 of Car type is: kawo (54.791378% confidence)
- Value 4 of Car type is: Sorry a Car cannot be recommended at this stage (100.0% confidence)
- Value 5 of Car type is: honda (45.0% confidence)

Explain | all conclusions | Restart

Recognition of Digits 0 to 9

In this application, each digit is presented by 5x7 bit map, as shown in figure 1.

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

1	2	3	4	5
6	7	8	9	1
				0
1	1	1	1	1
1	2	3	4	5
1	1	1	1	2
6	7	8	9	0
2	2	2	2	2
1	2	3	4	5
2	2	2	2	3
6	7	8	9	0
3	3	3	3	3
1	2	3	4	5

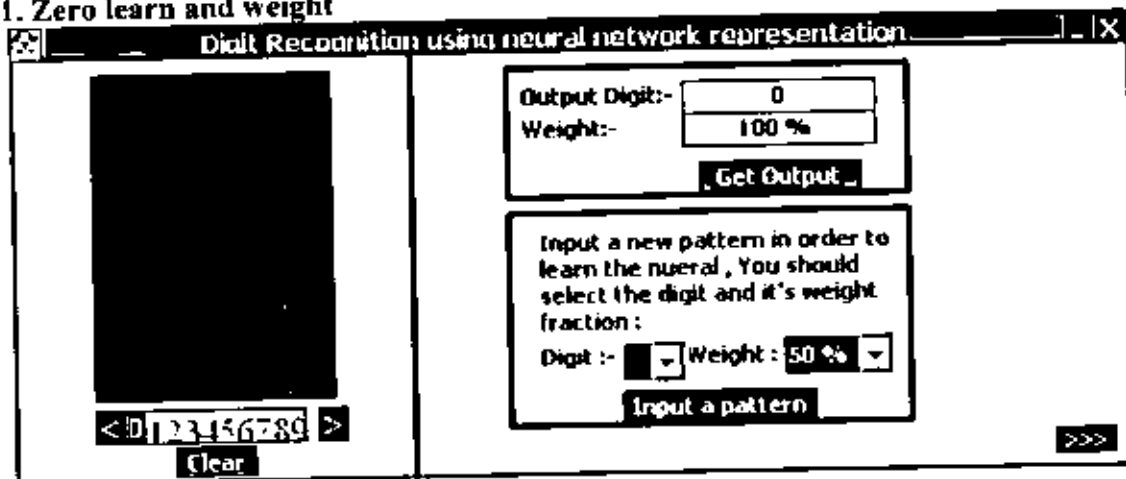
	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35

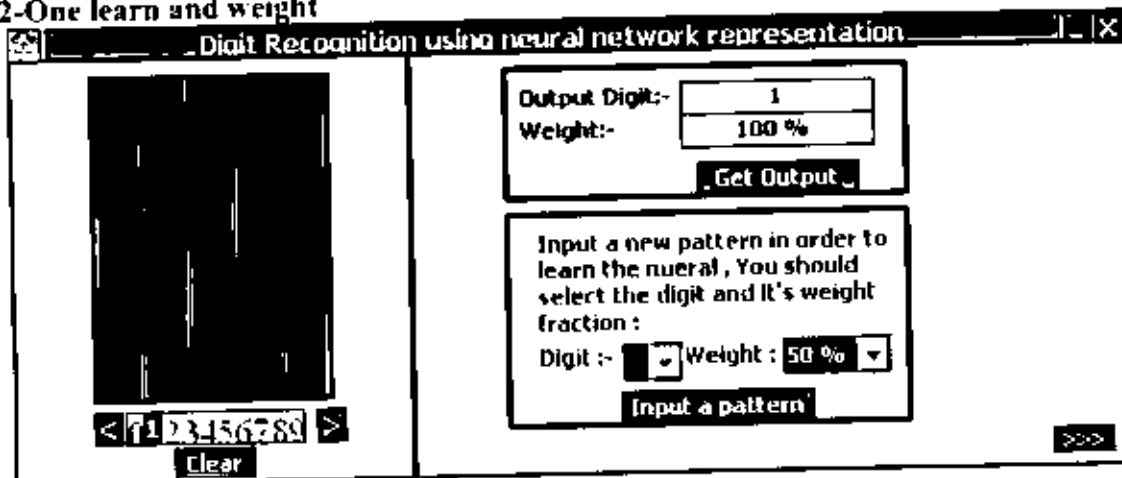
Figure 1 bit maps for digit recognition

Digit Recognition learn and weight

1. Zero learn and weight



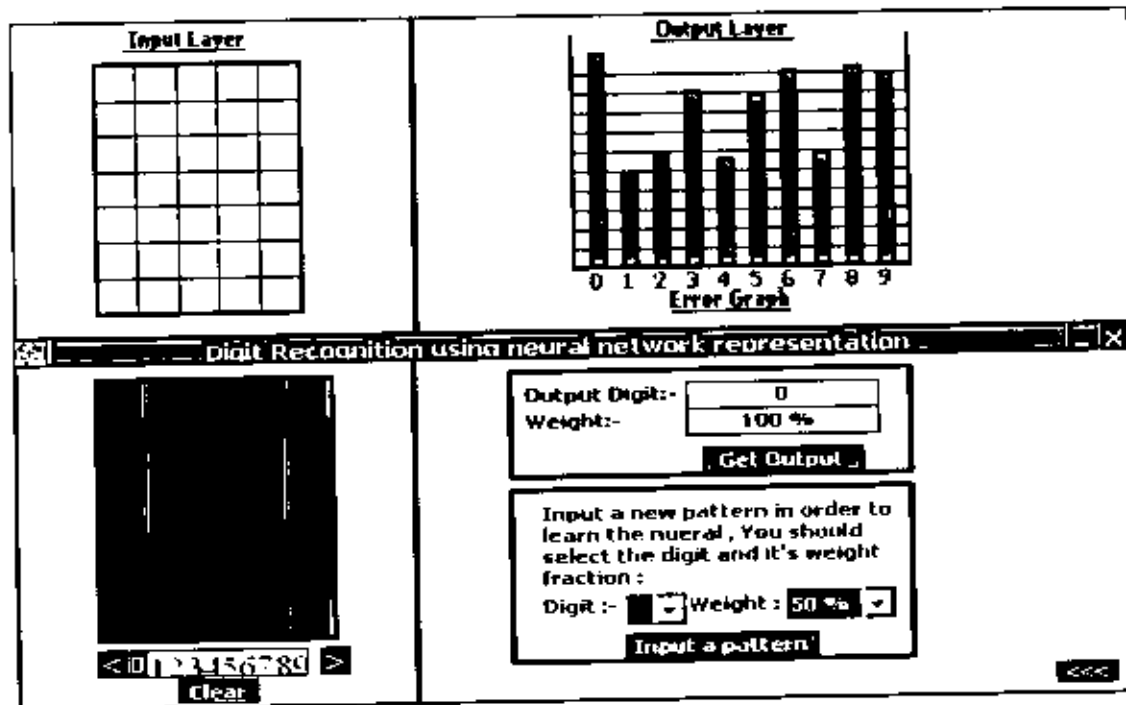
2-One learn and weight



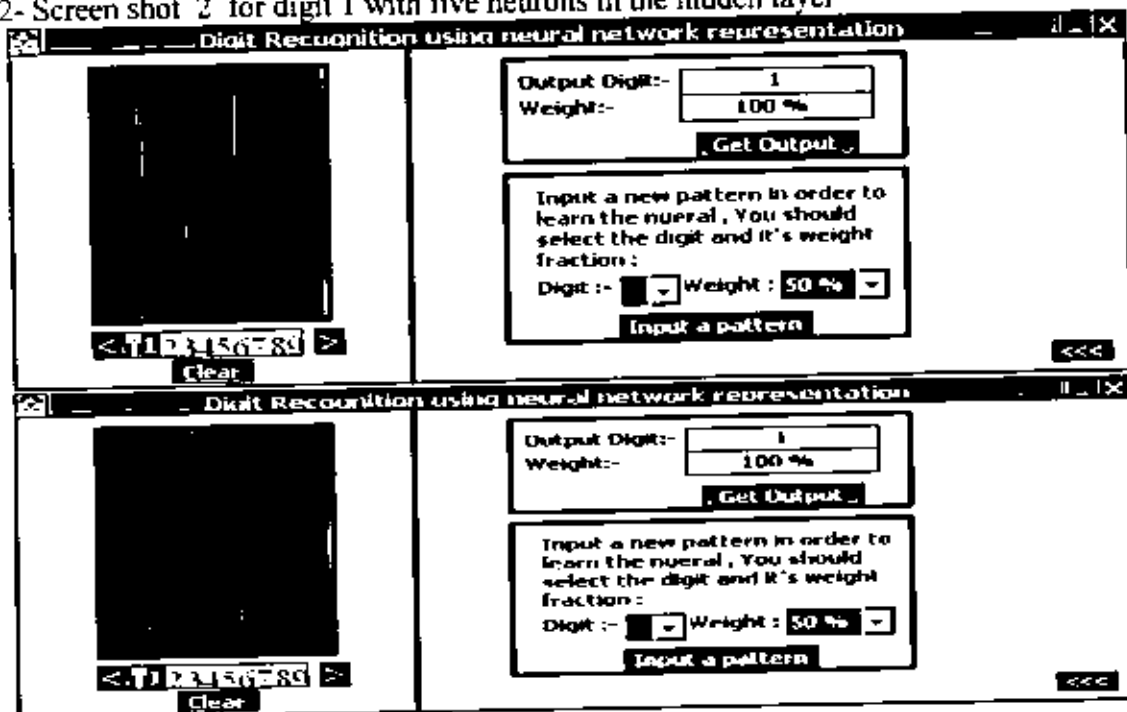
And so on for 2,3,4,5 6,7,8 and 9 digits.

Determine an optional number of hidden neurons

1- Screen shot 2 for digit 0 with five neurons in the hidden layer



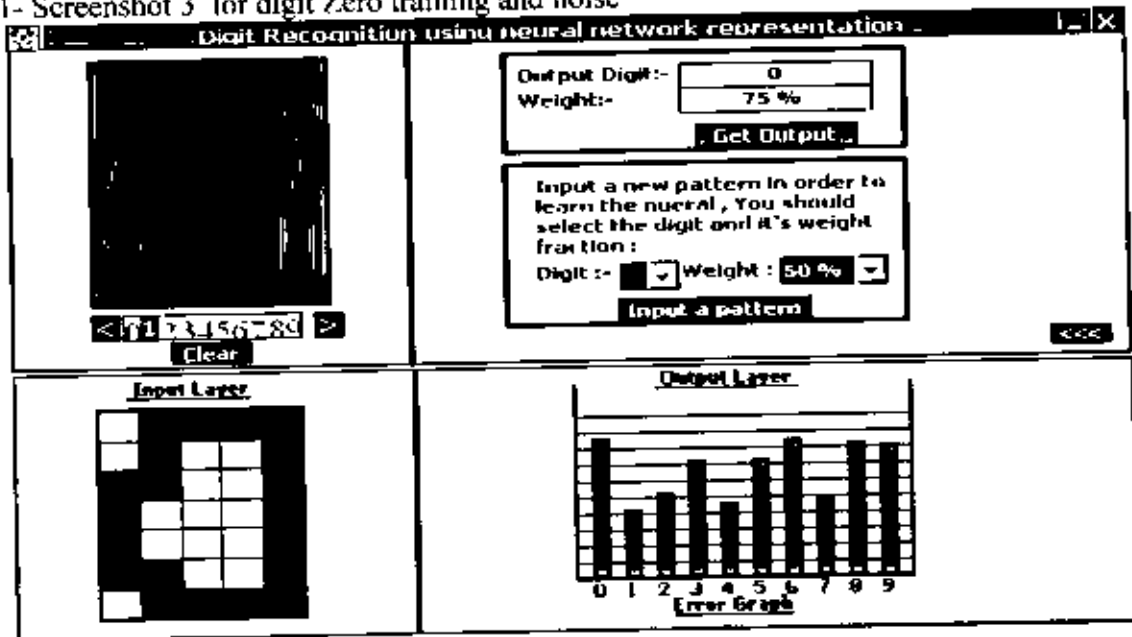
2- Screen shot 2 for digit 1 with five neurons in the hidden layer



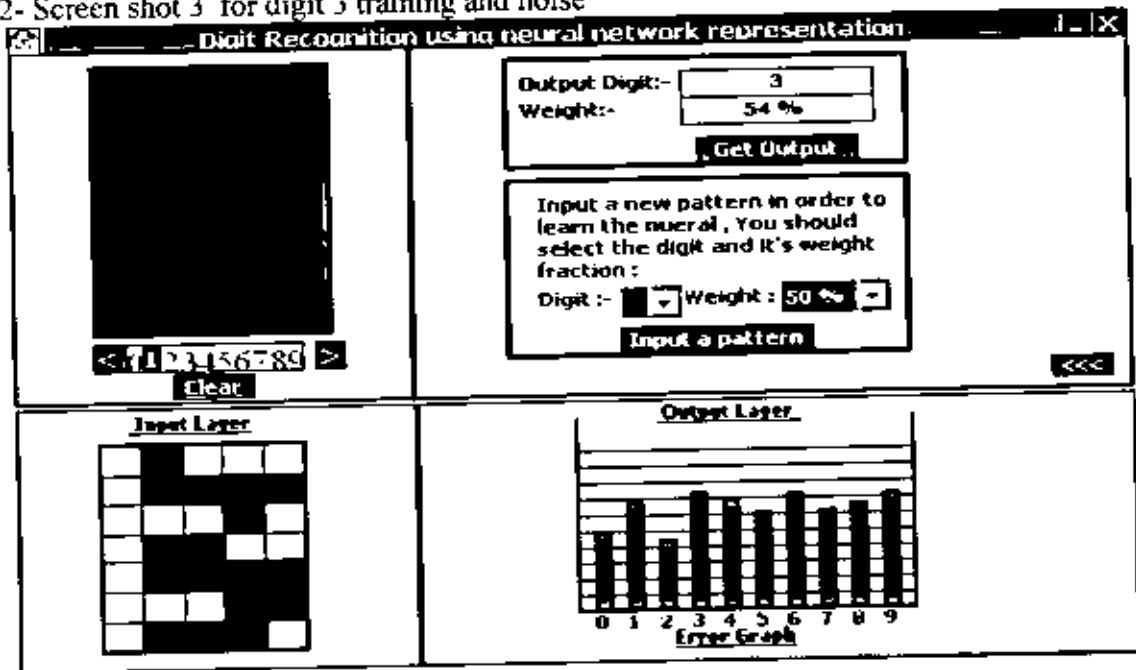
And so on for 2,3,4,5,6,7,8 and 9 digits.

Neural network training

1- Screenshot 3 for digit Zero training and noise

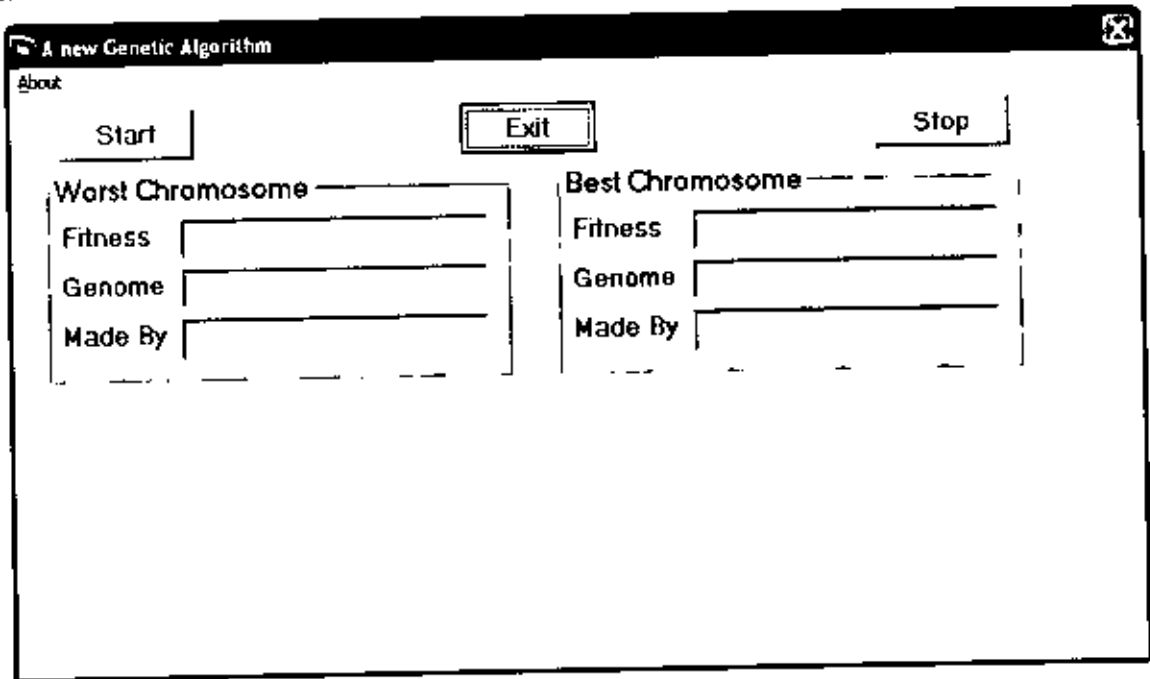


2- Screen shot 3 for digit 3 training and noise

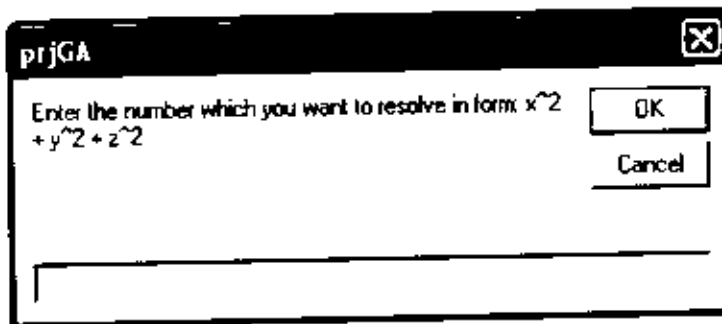


Resolve Equation

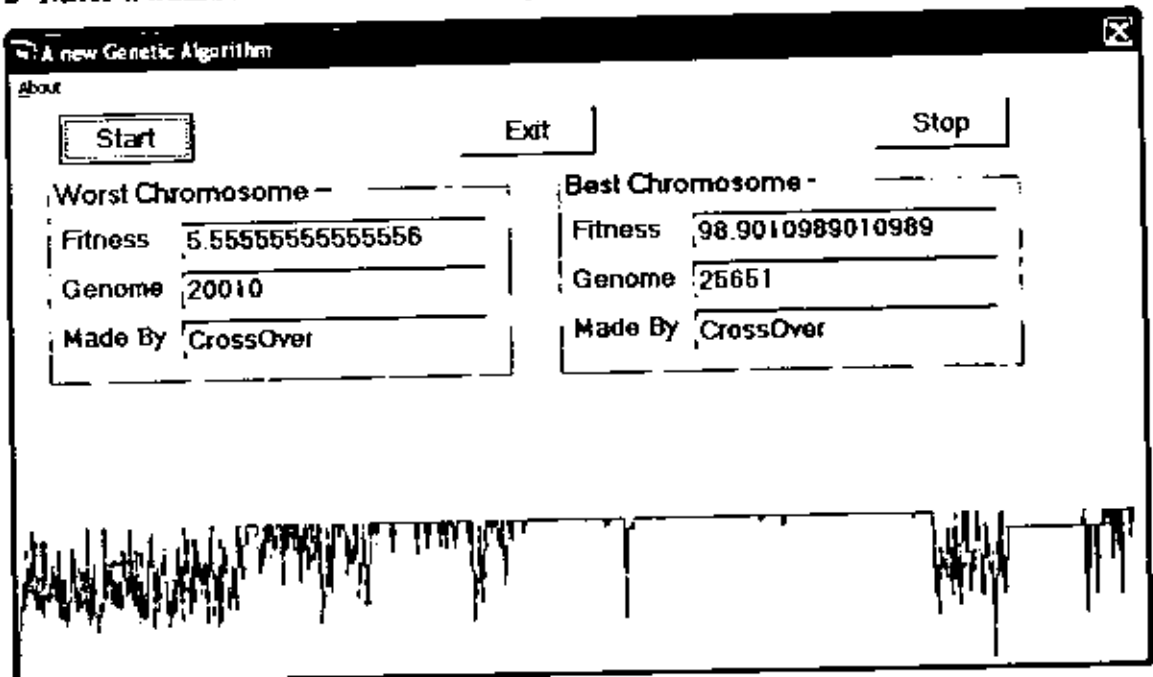
1- General menu



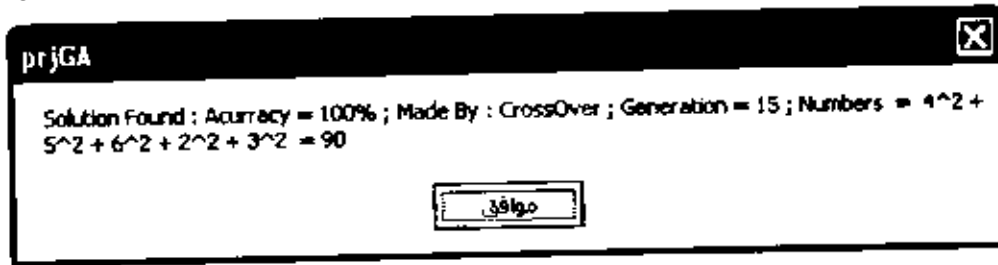
2- Press start button



2- Enter a number witch resolve the equation



3- The final result



conclusion from the result: the best values for the variable solving the equation when the number 90 entered is: $X=6, Y=2, Z=3$

ملخص الرسالة

تعتبر تقنيات تمثيل المعرفة بالإضافة للاستنتاج العمود الفقري لعلم الذكاء الصناعي ويعتبر تمثيل المعرفة حقل ثانوي كبير بين الذكاء الصناعي وعلم الإدراك الاصطناعي.

تهتم طرق تمثيل المعرفة بالطريقة التي تخزن بها المعلومات في دماغ الإنسان. إن المعايير الرئيسية لتقييم طرق تمثيل المعرفة هي: كفاية منطقية، قوة إرشادية بالإضافة لملائمة تدوين الملاحظات. وتعني الكفاية المنطقية بأنه يجب أن تكون طريقة تمثيل المعرفة قادرة على تمثيل كل التطبيقات بينما القوة الإرشادية تعني بأنه بالإضافة لامتلاك لغة تمثيل معبرة لابد أن يكون هناك بعض طرق التمثيل التي تم إنشاءها وتفسيرها ويمكن استخدامها لحل المسائل.

وطرق تمثيل المعرفة هي أحد المنهجيات المستخدمة بشكل واسع لتحليل وتمثيل المعرفة في الذكاء الصناعي، وبناء عليه تحليل ومعالجة أي مسألة بدون تمثيل المعرفة يعتبر مضيعة للوقت وخطأ فادح، وعلاوة على ذلك تحليل ومعالجة حل أي مسألة بدون اختيار الطريقة المناسبة لتمثيل المعرفة يعتبر مضيعة للوقت وخطأ فادح.

لحد الآن ليس هناك نظرية أو تقنية وحيدة لشرح وتنظيم المعرفة الإنسانية في الحاسوب التقليدي. ولا توجد تركيبة مثالية وحيدة من طرق تمثيل المعرفة صالحة لمعالجة جميع التطبيقات. وإحدى المسؤوليات المهمة بالنسبة لمهندس المعرفة هي اختيار الطريقة المناسبة من طرق تمثيل المعرفة لتمثيل المعرفة للتطبيق المراد معالجته.

تقارن هذه الرسالة بين بعض أنواع طرق تمثيل المعرفة مثل التمثيل بالقواعد، بالمنطق، بالشبكات الدلالية، بالأطر، بالمنطق الضبابي، بالشبكات العصبية، الخوارزميات الوراثية وذلك لبعض التطبيقات الحقيقية بالاعتماد على المنظورين النظري والعملية مستخدما المعايير: المتغيرات، العلاقات، البرمجة، تقنيات الاستدلال.

إن غرض هذه الرسالة هي مساعدة مهندس المعرفة على اختيار الطريقة المناسبة من طرق تمثيل المعرفة لتحليل ومعالجة التطبيق المراد حله. وقد تم تصميم 7 برامج بحيث يعالج كل برنامج مسألة ما بناء على مفهوم طريقة المعرفة المناسبة للتطبيق .

تحتوي هذه الرسالة على تسعة فصول وجزءاً خاصاً بالمراجع التي بلغت 29 مرجعاً بالإضافة لسبعة ملاحق. تضمنت فيها لقطات من شاشات تنفيذ البرامج التي قمت بتصميمها.

الفصل الأول: يحتوي على فكرة عامة عن الذكاء الصناعي وتعريفه بالإضافة لتمثيل المعرفة وأنواعها وأهم الأبحاث المتعلقة بموضوع تمثيل المعرفة بالإضافة لأهداف هذه الرسالة والدافع إليها.

الفصل الثاني: يوضح طريقة تمثيل المعرفة باستخدام القواعد .

الفصل الثالث: يوضح طريقة تمثيل المعرفة باستخدام المنطق.

الفصل الرابع: يوضح طريقة تمثيل المعرفة باستخدام الشبكات الدلالية

الفصل الخامس: يوضح طريقة تمثيل المعرفة باستخدام الأطر أو الهياكل.

الفصل السادس: يوضح طريقة تمثيل المعرفة باستخدام المنطق الضبابي.

الفصل السابع: يوضح طريقة تمثيل المعرفة باستخدام الشبكات العصبية .

الفصل الثامن: يوضح طريقة تمثيل المعرفة باستخدام الخوارزميات الوراثة.

الفصل التاسع: يتضمن مقارنة بين بعض أنواع طرق تمثيل المعرفة. وضمنت في نهايته

الاستنتاجات التي توصلت إليها من خلال المقارنة النظرية والعملية.



بن الدراسة ليست غاية في حد ذاتها
ولما الغاية هي خلق الإنسان المتواضع العبد

جامعة التحددي
سوت

التاريخ:

الموافق: 22 / 1 / 2001

الرقم الاشاري: ل.ع. / 103 / 2001

كلية العلوم
قسم الحاسوب
عنوان البحث

((طرق تمثيل المعرفة و مقارنتها من خلال بعض

التطبيقات الحقيقية))

مقدمة من الطالب
مروان حسين القبلاوي

لجنة المناقشة:
د. عبد الحميد محمد عبد الكافي
1400 (مشرف الرسالة)
الدكتور/د. الرئيس سامي الفقيه
(ممتحن داخلي)
الدكتور / د. ناجي أحمد بلزينة
(ممتحن خارجي)

1999

عند:
د. محمد علي سالم الفرجاني
أمين اللجنة الشعبية لكلية العلوم

www.althadadi.edu.ly



جامعة التحدّي - كلية العلوم

طرق تمثيل المعرفة ومقارنتها من خلال بعض التطبيقات الحقيقية

رسالة مقدمة لقسم الحاسوب

كمتطلب جزئي للحصول على درجة الماجستير في علوم الحاسوب

مقدمة من الطالب: مروان حسن القبلاوي

إشراف: د. عبدالحميد محمد عبدالكافي

العام الجامعي 2006/2005 ف