B.E.

Sixth Semester Examination, May-2007

Intelligent Systems (CSE-304)

Note : Attempt any five questions.

Q. 1. (a) Discuss the features of 'LISP' and 'Prolog' Programming Languages. To which generation of Programming Languages these two Languages belong ? 10

Ans. During the 1970s Prolog became popular in Europe for artificial intelligence applications. In the United State, however Prolog remained a relatively minor computer language. In the United States researchers preferred the LISP language for artificial intelligence applications. LISP was considered a more powerful language for these applications, though it was more difficult to learn & use than Prolog.

During these early years both Prolog & LISP were very slow in execution & consumed large amounts of memory. In addition, users needed considerable programming expertise to use a Prolog or LISP program.

The name Prolog was taken from the phrase "programming logic". The language was originally developed in 1972 Alaire Colmerquer & P. Roussel at the university of Marseilles in France.

Prolog uses only data about object & their relationships. Prolog also emphasizes symbolic processing. LISP and Prolog are the best-known object oriented languages.

A prolog program is a collection of data or facts & the relationships among these facts. In other words, the program is a database.

Prolog and LISP both are object-oriented languages. They use heuristics functions to solve problems. Both are most efficient at formal reasoning.

Q. 1. (b) Write the procedure in Prolog to concatenate two lists. 5

Ans. One of Prolog's most powerful structures the list. Prolog's ability to use lists in solving problems in one of the language's major advantages over languages such as BASIC and FORTRAN, which do not support list structures.

A list is an ordered sequence of terms. Lists terms can be variables, simple objects, compound objects or other lists. A list can contain an unlimited number of terms.

Each list is set off in brackets, with the components of the list separated by commas, as follows :

[alfred, bill, frank, mary, susan, tom]

Procedure in Prolog to Concatenate Two Lists :

To combine two lists, we can create an append predicate. Its format is as follows :

```
append ([ ], List B, List B).
append ([ X: List 1], List 2, [X : List 3]):–
append (List 1, List 2, List 3 ).
```

Example : Using the append predicate.

```
Goal : append ([a, b, c], [d, e], x )
        x = ["a" , "b" , "c" , "d" , "e" ]
        1 solution.
```

Q. 1. (c) What do you mean by a list in the context of LISP ? How it is represented in memory ? 5

Ans. During the 1970s Prolog became popular in Europe for artificial intelligence applications. In the United State, Prolog remained a relatively minor computer language. In the United States researchers preferred the LISP language for artificial intelligence applications. LISP was considered a more powerful language for these applications, though it was more difficult to learn & use than Prolog.

During these early years LISP were very slow in execution & consumed large amounts of memory. In addition, users needed considerable programming expertise to use a LISP program.

Q. 2. Discuss the searching technique "Best First Search" and "Breadth First Search" with example. When would best first search be worse than simple breadth first search ? 20

Ans. Breadth First Search :

(i) Create a variable called NODE-LIST and set it to the initial state.

(ii) Until a goal state is found or NODE-LIST is empty do:

(a) Remove the first element from NODE-LIST and call it E.

If NODE-LIST was empty, quit.

(b) For each way that each rule can match the state described in E do :

(i) Apply the rule to generate a new state.

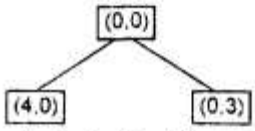
(ii) If the new state is a goal state, quit and return this state.

(iii) Otherwise & the new state to the end of NODE-LIST.

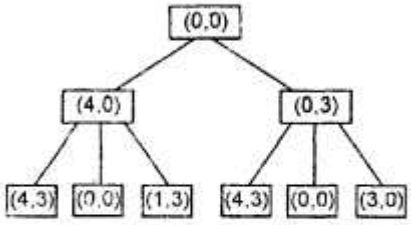
Advantages of Breath-First Search

(i) Breadth-first search will not get trapped exploring a blind all.

(ii) If there is a solution, then breadth-first search is guaranteed to find it. Furthermore, if there are multiple solutions, then a minimal solution (i.e., one that requires the minimum number of steps) will be found. This is guaranteed by the fact that longer paths are never explored until all shorter ones have already been examined.



One level of a Breadth-First Search Tree



Two levels of a Breadth-First Search Tree

Best First Search : A method, best-first search which is a way of combining the advantages of both depth-first and breadth-first search into a single method.

Algorithm : Best-First Search

(i) Start with OPEN containing just the initial state.

(ii) Until a goal is found or there are no nodes left on OPEN do :

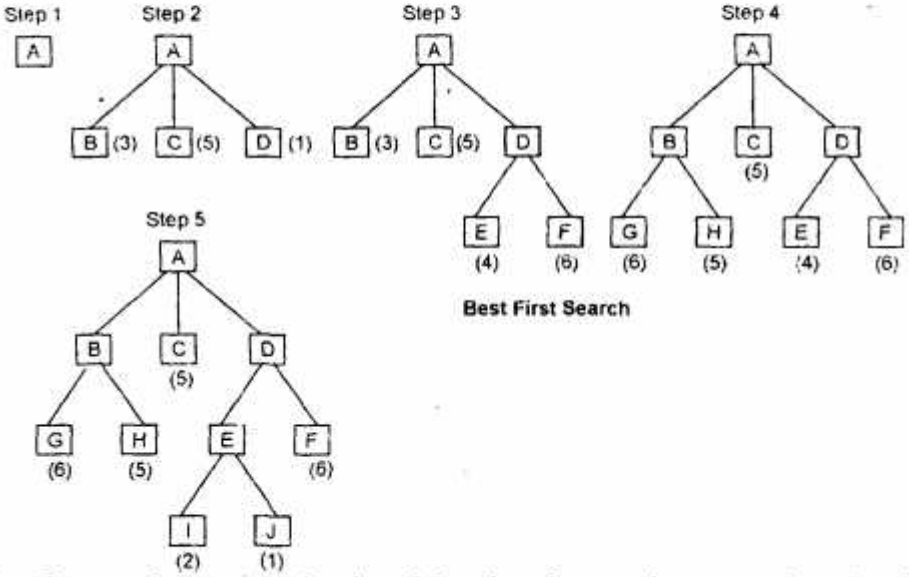
(a) Pick the best node on OPEN.

(b) Generate its successors.

(c) For each successor do :

(i) If it has not been generated before, evaluate it add it to OPEN and record its parent.

(ii) If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node & to any successors that this node may already have.



Best First Search

In best-first search, one move is selected, but the others are kept around, so that they can be revisited later if the selected path becomes less promising.

Q. 3. (a) What do you understand by backward and forward reasoning ? Explain with example.

Ans. The object of a search procedure is to discover a path through a problem space from an initial configuration to a goal state. While PROLOG only searches from a goal state. There are actually two directions in which such a search could proceed :

- Forward, from the start states.
- Backward, from the goal states.

Reason Forward from the Initial States : Begin building a tree of move sequences that might be solutions by starting with the initial configurations at the root of the tree. Generate the next level of the tree by finding all the rules whose left sides match the root node and using their right sides to create the new configurations. Generate the next level by taking each node generated at the previous level & applying to it all of the rules whose left sids match it. Continue until a configuration that matches the goal state is generated.

Reason Backward from the Goal State : Begin building a tree of move sequences that might be solutions by starting with the goal configurations at the root of the tree. Generate the next level of the tree by finding all the rules whose right sides match the root node. These are all the rules that ; if only we could

apply them, would generate the state we want. Use the left sides of the rules to generate the nodes at this second level of the tree. Generate the next level of the tree by taking each node at the previous level and finding all the rules whose right sides match it. Then use the corresponding left sides to generate the new nodes continue until a node that matches, the initial state is generated. This method of reasoning backward from the desired final state is often called goal-directed reasoning.

The production system model of the search process provides an easy way of viewing forward & backward reasoning as symmetric processes.

Consider the problem of solving a particular insstance of the 8-puzzle.

1	2	3
4	5	6
7	8	9

Assume the areas of the tray are numbered.

Square 1 empty and square 2 contains tile n →

Square 2 empty and square 1 contains tile n

Square 1 empty and square 4 contains tile n →

Square 4 empty and square 1 contains tile n

Square 2 empty and square 1 contains tile n →

Square 1 empty and square 2 contains tile n

Q. 3. (b) If a problem solving search program were to be written to solve each of the following types of problems, determine whether the search should proceed forward or backward. Also justify your answer :

(a) Water jug problem

(b) Fake coin problem. **8**

Ans. (a) A Water Jug Problem : You are given two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug ?

The state space for this problem can be described as the set of ordered pairs of integers (x, y), such that x = 0, 1, 2, 3 or 4 and y = 0, 1, 2 or 3; x represents the number of gallons of water in the 3-gallon jug and y represents the quantity of water in the 3-gallon jug. The start state is (0, 0), the goal state is (2 ,n) for any value of n.

	Current State	New State	
1.	(x, y) if x < 4	→ (4, y)	Fill the 4-gallon jug
2.	(x, y) if y < 3	→ (x, 3)	Fill the 3-gallon jug

3.	(x, y) if x>0	→ (x − d, y)	Pour some water out of the 4-gallon jug
4.	(x, y) if y>0	→ (x, y − d)	Pour some water out of the 3-gallon jug
5.	(x, y) if x>0	→ (0, y)	Empty 4-gallon jug on the ground
6.	(x, y) if y>0	→ (x, 0)	Empty the 3 gallon jug on the ground
7.	(x, y) if x + y ≥ 4 & y > 0	→ (4, y − (4 − x))	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8.	(x, y) if x + y ≥ 3 & x > 0	→ (x − (3 − y), 3)	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is uill
9.	(x, y) if x + y ≤ 4 & y > 0	→ (x + y ,0)	Pour all the water from the 3-gallon jug into the 4-gallon jug
10.	(x, y) if x + y ≤ 3 & x > 0	→ (0, x +y)	Pour all the water from the 4-gallon jug into the3-gallon jug.
11.	(0, 2)	→ (2, 0)	Pour the 2 gallons from the 3-gallon jug into the 4-gallon jug.
12.	(2, y)	→ (0, y)	Empty the 2 gallon is th 4-gallon jug on the ground

Production Rules for the water jug Problem

Gallons in the 4-gallon jug	Gallon in the 3-Gallon jug		Rule Applied
0	0	⟶	2
0	3	⟶	9
3	0	⟶	2
3	3	⟶	7
4	2	⟶	5 or 12
0	2	⟶	9 or 11
2	0		

Solution to the water jug problem.

(b) Fake Coin Problem : You have a balance scale and 12 coins, which is counterfeit. The counterfeit weigh less or more than the other coins. Can you determine the counter feit in 3 weighting, 4 till if it is heavier or lighter ?

Problem : For some given n>1, there are $(\hat{3}n - 3)/2$ coins, 1 of which is counterfeit. The counterfeit weigh less or more than the other coins. Can you state on for hand n, weighting experiments with a balance scale, with which you determine the counter feit coins & till if it is heavier or lighter.

The Solution for 3 Coins : For n = 2 there are 3 coins, weighing are

1 against 2

1 against 3

Both of three can have three outcomes : fall to the left (l), fall to the right (r), or balance (b) the following table gives the answer for each of these outcomes :

Outcomes		Fake Coin	Solution for 12 Coins
l	l	1 too heavy	n = 3, we replace
l	b	2 too light	coins 'c' by coins
l	r	(not possible)	3c – 2, 3c – 1, & 3c
b	l	3 too light	
b	b	no fake coin	
b	r	3 too heavy	1 2 3 against 4 5 6
r	l	(not possible)	1 2 3 against 7 8 9
r	b	2 too heavy	
r	r	1 too light	

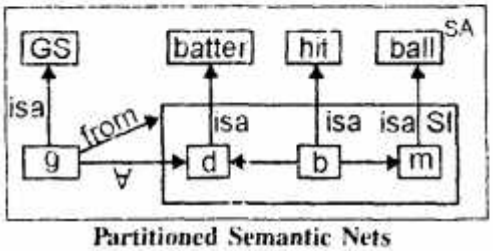
Q. 4. (a) What are semantic Nets ? Construct partitioned semantic Net representation for the following : **12**

(i) Every batter hit a ball

(ii) All the batters like the pitches

Ans. Semantic Nets : The main idea behind semantic nets is that the meaning of a concept comes, from the ways in which it is connected to other concepts. In a semantic net, information is represented as a set of nodes connected to each other by a set of labeled arcs, which represent relationship among the node.

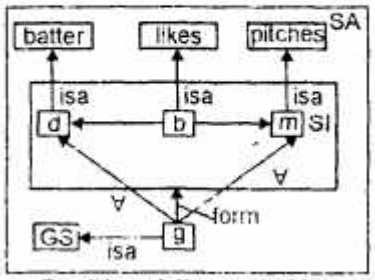
(i) Every batter hit a ball : $\forall x : \text{batter}(x) \longrightarrow \exists_{(y)} ; \text{ball}(y) \wedge \text{Bite}(x, y)$



Partitioned Semantic Nets

Node g stands for the assertion Node g is an instance of the special class GS of general statement about the world.

(ii) All the batters like the pitches :



Partitioned Semantic Nets

g has two ∀ links, one pointing to d, which represents any batter, and one pointing to m, representing any pitches.

Q. 4. (b) Consider the following sentences :

(i) John likes all kind of food.

(ii) Apples are food

(iii) Chicken is food

(iv) Anything any one eats and isn't killed by is food

(v) Bill eats peanuts and is still alive

(vi) Sue eats everything Bill eats.

Translate these sentences into formulas in predicate logic. 8

Ans. (i) ∀x : likes (John, food (x))

(ii) Food (Apples)

(iii) food (chicken)

(iv) ∀x, ∀y : eat s(x, y) A alive (x) → food (y)

(v) eats (Bill, peanuts) A alive (Bill)

(v) ∀x : eats (Bill, x) → eats (Sue, x)

Q. 5. Discuss about following in brief : 20

(a) Fuzzy logic

(b) Dempster-Shafer theory

(c) Non Monotic Reasoning

(d) Statistical Reasoning

Ans. (a) Fuzzy logic : In the representation techniques, we have not modified the mathematical underpinnings provided by set theory and logic. We have augmented those ideas with additional constructs provided by probability theory. In this, we take a different approach and briefly consider what happens if we make fundamental changes to our idea of set membership and corresponding changes to our definitions of logical operations.

The motivation for fuzzy sets is provided by the need to represent such propositions as ;

John is very tall.

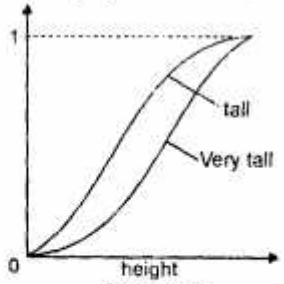
Mary is slightly ill.

Sue and Linda are close friends.

Exceptions to the rule are nearly impossible.

Most Frenchmen are not very tall.

While traditional set theory defines set membership as a boolean predicate, fuzzy set, theory allows us to represent set, membership as possibly distribution, such as shown in fig. for the set of tall people and the set of very tall people.



fuzzy logic

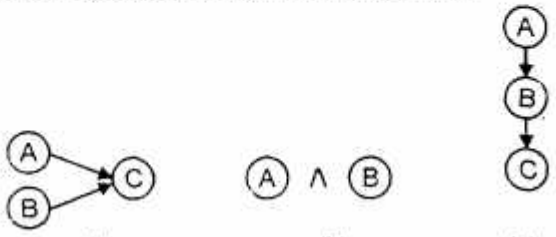
(b) Dempster-Shafer Theory : There are several techniques, all of which consider individual propositions and assign to each of thems a point estimate (i.e., a single number) of the degree of belief that is warranted given the evidence. But Dempster Shafer theory is new approach which considers sets of

propositions and assigns to each of them of interval [Belief, Plausibility] in which the degree of belief must lie. Belief measures the strength of the evidence in favour of set of propositions. It ranges from 0 to 1.

Plausibility (Pl) is defined to be

$$Pl(S) = 1 - Bel(\neg S)$$

It also ranges from 0 to 1 & measures the extent to which evidence in favour of $\neg S$ leaves room for belief in S. In certainty factors, there are three combination scenarios.



(a) (b) (c)

When we use Dempster-Shafer theory, we do not need an explicit combining function for the scenario in fig. (b). Since we have that capability already in our ability to assign a value of m to a set of hypotheses. But we do need a mechanism for performing the combinations of scenarios (a) & (c). Dempster's rule of combination's serves both these functions. It allows us to combine any two belief functions.

Suppose we given two belief functions m_1 and m_2 Let X be the set of subsets of $(-)$ to which m_1 assigns a non-zero value and let Y be the correspondings set for m_2. We define the combination m_3 of m_1 and m_2 to be

$$m_3(z) = \frac{\Sigma_{X \cap Y = Z}\, m_1(X) \cdot m_2(Y)}{1 - \Sigma_{X \cap Y = \phi}\, m_1(X) \cdot m_2(Y)}$$

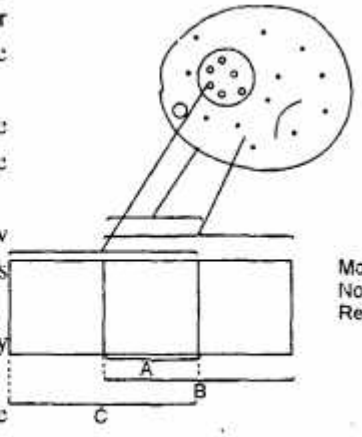
(c) Non-Monotic Reasoning : Non-monotic reasoning, in which the axioms and/or the rules of inference are extended to make it possible to reason within complete information. These systems preserve however, the property, that, at any given moment, a statement is either believed to be true, believed to be false or not believed to be either.

Non-monotonic reasoning systems, are designed to be able to solve problems in which all of these properties may be missing :

(1) How can the knowledge base be extended to allow inferences to be made on the basis of lack of knowledge as well as on the presence of it ?

(2) How can the knowledge base be updated property when a new fact is added to the system ?

(3) How can knowledge be used to help resolve conflicts when there are several inconsistent non-monotonic inferences that could be drawn ?



Models, Wff's and Non-monotonic Reasoning

Figure shows one way of visualizing how non-monotonic reasoning works in all of them. The box labeled A corresponds to an original set of wff's. The large circle contains all the models of A.

Non-monotonic Logic : One system that provides a basis for default reasoning is Non-monotonic Logic (NML), in which the language of first-order predicate logic is augmented with a modal operator M, which can be read as "is consistent." For example, the formula,

$$\forall x, y : \text{Related}(x, y) \wedge M\,\text{Get A long}(x, y) \rightarrow \text{Will Defend}(x, y)$$

(d) Statistical Reasoning : An important goal for many problem-solving systems is to collect evidence as the system goes along and to modify its behaviour on the basis of the evidence. To model this behaviour, we need a statistical theory of evidence. Bayesian statistics is such a theory. The fundamental notion of Bayesian statistics is that of conditional probability :

$$P(H/E)$$

The probability of hypothesis H given that evidence E. To compute this, we need to take into account the prior probability of H and the extent to which E provides evidence of H.

$P(Hi/E)$ = The probability that hypothesis H_i is true given evidence E.

$P(E/Hi)$ = The probability that will observe evidence E given that hypothesis i is true.

$P(H_i)$ = the a priori probability that hypothesis i true in the absence of any specific evidence.

These probabilities are called prior probabilities or priors. Bayes's theorem states that

$$P(Hi/E) = \frac{P(E/Hi) \cdot P(H_i)}{\sum_{n=1}^{k} P(E/H_n) \cdot P(H_n)}$$

Statistical techniques, such as multivariate analysis, provide an alternative approach to building expert-level systems. Statistical methods do not provide explicit rules that humans can understand. Therefore, it is difficult for them to explain their decisions.

Q. 6. (a) What do you understand by planning ? Justify its significance.

Ans. The word planning refers to the process of computing several steps of a problem-solving procedure before executing any of them. When we describe computer problem-solving behaviour, the distinction between planning and doing facts a bit since rarely can the computer actually do much of anything besides plan. In solving the 8-puzzle, for example, it cannot actually push any tiles around. So when we discussed the computer. Solution of the 8-puzzle problem, what we were really doing was outlining the way the computer might generate a plan for solving it. For problems such as the 8 puzzle, the distinction between planning and doing it unimportant.

The process of planning a complete solution can proceed just as would an attempt to find a solution by actually trying particular actions. If a dead-end path is detected, then a new one can be explored by backtracking to the last choice point. So, for example, in solving the 8-puzzle, a computer could look at a solution plan in the same way as a person who was actually trying to solve the problem by moving tiles on a board. If solution steps in the real world cannot be ignored or undone, though, planning becomes extremely important. Although real world steps may be irrevocable, computer simulation of those steps is not. So we can circumvent the constraints of the real world by looking for a complete solution in a simulated world in which backtracking is allowed. After we find a solution, we can execute it in the real world.

There are different Planning Techniques :

1. Triangle Tables : Provide a way of recording the goals that each operator is expected to satisfy as well as the goals that must be true for it to execute correctly. If something unexpected happens during the execution of a plan, the table provides the information required to patch the plan.

2. Metaplanning : A technique for reasoning not just about the problem being solved but all about the planning process itself.

3. Macro-operators : Allow a planner to build new operators that represent commonly used sequance of operators.

4. Case-Based Planning : Re-uses old plans to make new ones.

Q. 6. (b) What is Learning ? Explain. Also discuss various types of learnings. **14**

Ans. One of the most often criticisms of AI is that machines cannot be called intelligent until they ae able to do new things and do adapt to new situations, rather than simply doing as they are told to do.

According to Simnon, learning denotes

"..... changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

Learning covers a wide range of phenomena. At one end of the spectrum is skill refinement. People get better at many tasks simply by practising. At the other end of the spectrum lies knowledge acquisition.

Knowledge acquisition itself includes many different activities. Simple storing of computed information, or rote learning, is the most basic learning activity.

When a computer stores a piece of data, it is performing a rudimentary form of learning. In the case of data caching, we store computed values. So that we do not have to recompute them later. Caching has been used in A^I programs to produre. Some surprising performance improvements. Such caching is known as rote learnning.

There are different Types of Learning :

Winston's Learning Program : Winston describes a structural concept learning program. This program operated in a simple blocks world domains. Its goal was to construct representation of the definitions of concepts in the blocks domain. For example, it learned the concepts House, Tent and Arch shown in fig. The figure also shows an example of a near miss for each concept. A near miss is an object that is not are instance of the concept in question but that is very similar to such instances.

Version Spaces : Mitchell describes a approach to concept learning called version spaces. The goal is the same : to produce a description that is consistent with all positive examples but no negative examples in the training set.

Unsupervised Learning : What if a neural network is given no feedback for its outputs, not even a real-valued reinforcement ? Can the network learn anything useful ? The unintuitive answer is Yes. This form of learning is called unsupervised learning. Given a set of input data, the network is allowed to play with it to try to discover regularities and relationships between the different parts of the input.

Reinforcement Learning : What if we train our networks not with punishment and reward instead ? This process is certainly sufficient to train animals to perform relatively interesting task S. Barto discribes a network which learns as follows : (i) The network is presented with a sample input from the training set,

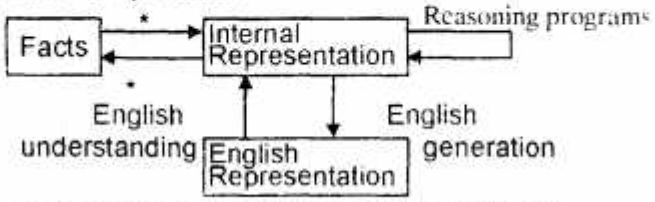
(ii) The network computes what it thinks should be the sample output. (iii) The network is supplied with a real valued judgment. (iv) The network adjusts its weights and the process repeats.

Q. 7. Discuss about expert system and neural network. Also contrast these two in terms of knowledge representation, knowledge acquisition and explanation. **20**

Ans. Expert systems solve problems that are normally solved by human "experts". To solve expert level problems, expert systems need access to a substantial domain knowledge base, which must be built as efficiently as possible. They also need to exploit one or more reasoning mechanisms to apply their knowledge to the problems they are given. Then they need a mechanism for explaining. What they have done to the users who rely on them. The most widely used way ofrepresenting domain knowledge in expert systems is as a set of production rules, which are often coupled with a frame system that defines the objects that occur in the rules.

Neural Networks : Research in neural networks came to virtual halt in the 1970, however, when the networks under study were shown to be very weak computationally. There are several reasons for including the appearance of faster digital computer on which to simulate larger networks, the interest in building massively parallel computers to and most important, the discovery of new neural network architectures and powerful learning algorithms.

Knowledge Representation : In order to solve the complex problems encountered in artificial intelligence, one needs both a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to new problems.



Mapping between Facts & Representation

(i) The knowledge level, at which facts (including each agent's behaviours and current goals) are described.

(ii) The symbol level, at which representation of objects at the knowledge level are defined in terms of symbols that can be manipulated by programs.

Knowledge Acquisition : Knowledge acquisition systems exist, many programs that interact with domain experts to extract expert knowledge efficiently. These programs provide support for the following activities :

(i) Entering knowledge

(ii) Maintaining knowledge base consistency

(iii) Ensuring knowledge base completeness.

The most useful knowledge acquisition programs are those that are restricted to a particular problem-solving paradigm, e.g., diagnosis or design.

Explanation : In order for an expert-system to be an effective tool, people must be able to interact with it easily. To facilitate this interaction, the expert system must have the following two capabilities :

(i) Explain its reasoning : In many of the domains in witch expert systems operate, people will not accept results unless they have been convinced of the accuracy of the reasoning process that produced those results.

(ii) Acquire new knowledge and modifications of old knowledge.

TEIRESIAS was the first program to support explanation and knowledge acquisition.

Q. 8. (a) Discuss all the steps involved in Natural language processing. 12

Ans. Language is meant for communicating about the world. We can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems.

There are number of steps involved in Natural Language processing :

(i) Morphological Analysis : Individual words are analyzed into their components and non-word tokens, such as punctuation, are separated from the words.

(ii) Syntactic Analysis : Linear sequences of words are transformed into structures that show how the words relate to each other. Some word sequences may be rejected if they violate the language's rules for how words may be combined.

(iii) Semantic Analysis : The structures created by the syntactic analyzer are assigned meanings. In other words, a mapping is made between the syntactic structures and objects in the task domains. Structures for which no such mapping is possible may be rejected.

(iv) Discourse Integration : The meaning of an individual sentence may depend on the sentences that precede it and may influence the meanings of the sentences that follow it.

(v) Pragmatic Analysis : The structure representing what was said is reinterpreted to determine what was actually meant.

Q. 8. (b) AI applications to robotics. 8

Ans. AI applications to Robotics :

(i) The input to an AI program is symbolic in form e.g., an 8-puzzle configuration or a typed english sentence. The input to a robot is typically an analog signal. Such as a two-dimensional video image or a speech waveform.

(ii) Robots require special hardware for perceiving and affecting the world, while AI programs require only general-purpose computers.

(iii) Robot sensors are inaccurate and thier effectors are limited in precision. There is always some degree of uncertainty about exactly where the robot is located and where objects and obstacles stand in relation to it. Robot effectors are also limited in precision.

(iv) Many robots must react in real time. A robot fighter plane, for example, cannot affod to search optimally or to stop monitoring the world during a LISP garbage collection.

(v) The real world is unpredictable, dynamic and uncertain. A robot cannot hope to maintain a correct and complete description of the world. This means that a robot must consdier the trade-off between devising and executing plans.

(vi) Because robots must operate in the real world, searching and backtracking can be costly. Consider the problem of moving furniture into a room. Operating in a simulated world with full information, an AI program can come up with an optimal plan by best-first search. Preconditions of operators can be checked quickly and if an operator facts to apply, another can be tried. Checking precondition in the real world, however, can be time-consuming if the robot does not have full information.