1. Define Artificial Intelligence. Explain various task domain of AI

:-

Artifical Intelligence (AI) is the study of how to make computer do things which, at the moment, people do better.

AI focused on formal tasks, such as playing games and theorem proving. Some games checker playing progeam that not only played games with opponents but also used its experience at those games to improve its later performance.

Game playing and theotem proving share the property that people who do them well at those task simply by being fast at exploring large amount of solution path and then and selecting the best one. AI focused on the sort of problem solving that we do every day when we decided how to get to work in the morning often called commonsense reasoning.

AI research progressed and techniques for handling larger amount of world knowledge were developed some progeam was made on the tasks just described and new tasks could reasonable be attempted. These include perception (vision and speech) natural language understanding and problem solving in specialized domain such as medical diagnosis and chemical analysis.

The ability to use language to communicate a wide variety of ideas is perhaps the most important thing that separats human from the other animals. We simplify the problem by restricting it to written languages, this problem usually refered to as natural language understanding.

In addition to these mundane task, many people can also perform one or more specialized task in which carefully acquired expertise is necessary. example of such task includes engineering design, Scientific discovery, medical diagnosis and financial planning.

Scanned by TapScanner

Mundane Tasks

- Perception
  - Vision
  - Speech
- Natural language
  - Understanding
  - Generation
  - Translation
- Commonsense reasoning
- Robot control

Formal Tasks

- Games
  - Chess
  - Backgammon
  - checkers - GO
- Mathematics
  - Geometry
  - Logic
  - Integral calculus
  - Proving properties of program

Scanned by TapScanner

Expert Tasks

- Engineering
  - Design
  - Fault Finding
  - Manufacturing planning
- Scientific analysis
- Medical diagnosis
- Financial analysis

Fig :- Some of the Task Domain of Artifical Intelligence

AI is also called as expert System. it is the attempt to Solve part or all of a partical significant problem that previously required human expertise.

* Artifical Intelligence Techniques

1) AI Technique is a Comman manner to organism and use the knowledge efficient it should be perceivable by people who provides it.

2) It should be easily modifyable to correct error, it should be used in many Situation.

Scanned by TapScanner

2. Write Breath first search Algorithm with example.

:- 1) Breath first search is the most common search strategy for traversing a tree or a graph. this algorithm searchs breathwise in a tree or a graph so it is called breath first search.

2) Breath First search algorithm start searching from the root node of tree and expands all successor node at the current label before moving the nodes of next level.

3) It is uninformed type of search techniques (lined or Brute force method).

4) Breath first search is implemented by using FIFO queue structure.

* Algorithm

Step 1 :- Create a variable called NODE-LIST and set it to the initial state.

Step 2 :- until a goal state is found or NODE-LIST is empty.

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 :

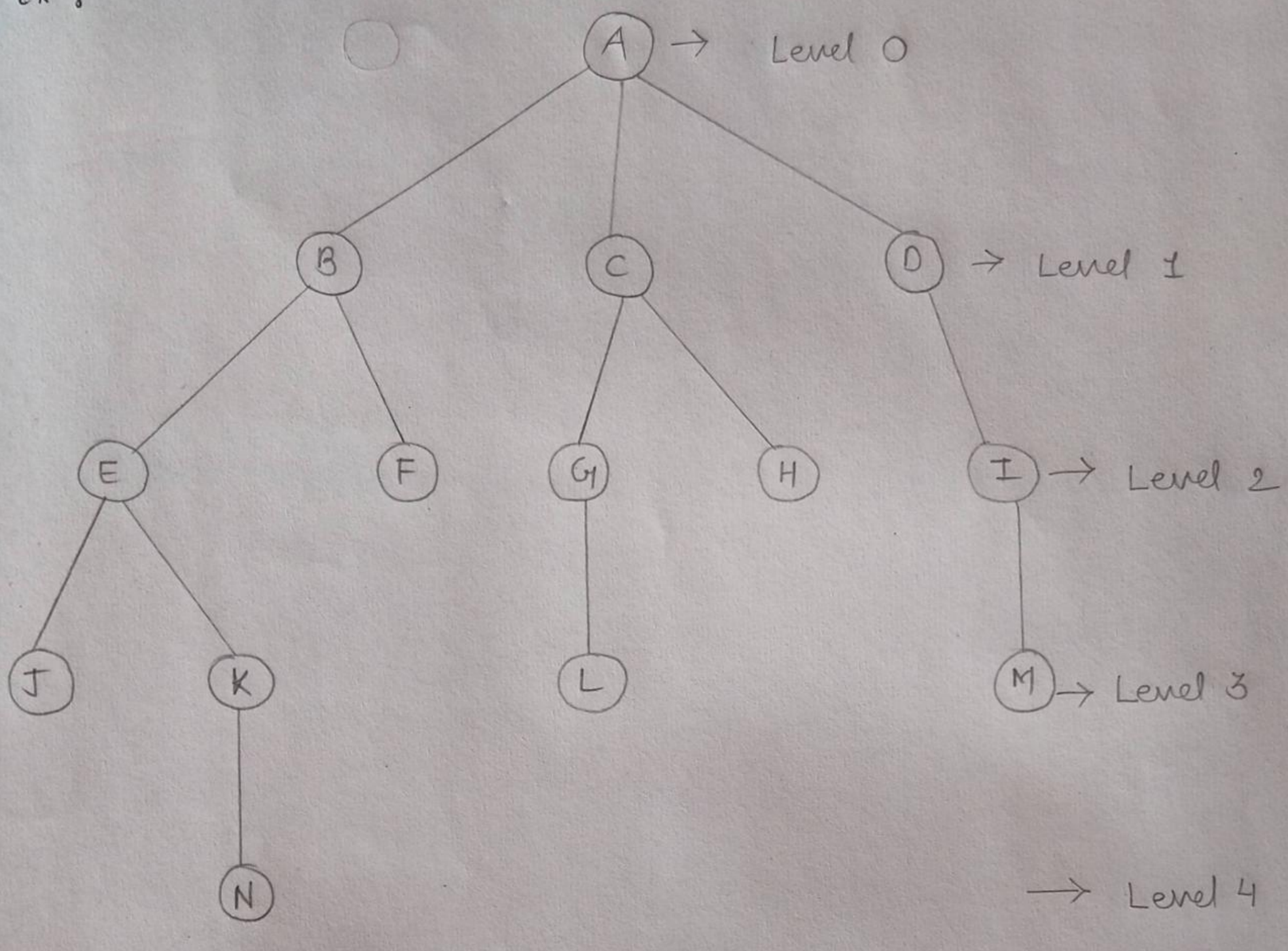
i) Apply the rule to genrate a new state

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

iii) otherwise, add the new state to the end of NODE-LIST.

Scanned by TapScanner

Ex :-



A			

B	C	D		

C	D	E	F	

D	E	F	G	H

E	F	G	H	I	

F	G	H	I	J	K

H	I	J	K	L	

I	K	L	M		

L	M	N			

A - B - C - D - E

Scanned by TapScanner

Breath First Search is optimal because it always gives shortest path as a result.
The time complexity in Breath First Search can be calculated as follow.

$$O(b^d)$$

where,

b = branch factors
d = depth levels.

* Advantages for Breath first Search

1) Breath first Search with Provids the Solution if any Solution exist.

2) If their are more than one Solution for a given Problem then BFS will Provids minimal Solution which requires least number of steps.

* Disadvantages for Breath first Search

1) It requires lot of memory since each level of the tree must be saved into the memory to expand the next levels.

2) Breath First Search needs a lots of times if the Solution is for aways from the root node.

Scanned by TapScanner

3. What is Production System in AI? Explain Production System characteristics.

:-

We have just examined a set of characteristics that distinguish various classes of problem. we have also argued that production system are a good way to describe the operation that can be performed in a search for a solution to a problem.

There are five classes of production system they are as followed.

1) Monotonic Production System
2) Non - Monotonic production System
3) A Partially commutative production System
4) A commutative production System
5) Not partially commutative production System

1) Monotonic Production System :-

Application of role prevents later application of another rule. (rules are independent of each other)

2) Non- Monotonic Production System :-

Non- Monotonic which is not true.

3) A Partially Commutative Production System :-

If is a production System with the property that if the application of a particular sequence of rule transforms state X to state Y then any Permitative of those rule that allowable.

Scanned by TapScanner

4) A Commutative production System :-

It is a production system that is both monotonic and partially commutative (means no change in application of rule and any sequence is allowable).

5) Not Partially Commutative Production System :-

This is not true.

* Production System Categories.

	Monotonic	Non-Monotonic
Partially Commutative	Theorem proving	Robot Navigation
Not-Partially Commutative	Chemical Analysis	Bridge

Fig :- Four categories of Production System

1) Partially Commutative Monotonic Production System :-

Are useful for solving ignorable problems.
ex :- (Theorem proving) problems that involve creating new things rather than changing old one are ignoreable.

2) Partially Commutative Non-Monotonic Production System :-

Are useful for problem in which changes occur but can it reversed and in which order of operation is not critical.

Scanned by TapScanner

This is the case in physical manipulation problem such as Robot Navigation on flat plane.

Suppose that the robot has following operator.

1) Go North (N)
2) Go South (S)
3) Go East (E)
4) Go West (W)

To reach its goal it does not matter wheather robot executes N-N-E or N-E-N depending on how the operator are selected.

3) Not Partially Commutative Monotonic Production System :-

Are useful for many problem in which irreversiable change occur the order in which they are perform can be very important for determining the final output you can not reverse the step and we cannot change the order.

4) Not Partially Commulative Non-Monotonic Production System :-

Are useful in which reversible changes occur and order does not matter.

Scanned by TapScanner

4. Explain the Approaches to knowledge Representation.

:- A good System for the representation of knowledge in a particular domain should posses the following four properties.

- Representational Adequacy :- The ability to represent all of the kinds of knowledge that are needed in that domain.
- Inferential Adequacy :- The ability to manipulate the representational structure in such a way as to derives new structure coresponding to new knowledge inferred from old.
- Inferential Efficiency :- The ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions.
- Acquisitional Efficiency :- The ability to acquire new information easily. The simplest case involves direct insertion, by a Person of new knowledge into the database. Ideally, the Program itself would be able to control knowledge acquisition.

unfortunately, no single System that can optimizes all of the capabilities for all kind of knowledge has yet been found. As a result, multiple techniques for knowledge representation exist. many Programs rely on more than one technique. They are as follows.

Scanned by TapScanner

1) Single Relational knowledge
2) Inheritable knowledge
3) Property Inheritance
4) Inferential knowledge
5) Procedural knowledge

1) Single Relational knowledge :- Simplest way to represent declarative facts is as a set of relations of the same sort used in database system. shown an example of such a relation system.

Player	Height	weight	Bats - Throws
Hank Aaron	6-0	180	Right - Right
Willie Mays	5-10	170	Right - Right
Babe Ruth	6-2	215	Left - Left
Ted Williams	6-3	205	Left - Right
player_info ('hank aaron', '6-0', 180, right).			

Fig :- Simple Relational knowledge and a Sample fact in prolog.

The reason that this representation is simple is that standing alone it provids very weak inferential capabilities but knowledge represented in this form may serve as the input to more powerful inference engines.

Providing support for relational knowledge is what database system to do. Thus we do not need to discuss this kind of knowledge representation structure.

Scanned by TapScanner

2) Inheritable knowledge :- In order to support property inheritance object must be organized into classes and classes must be arranged in a generalization hierarchy. Shows some additional baseball knowledge inserted into a structure that is so arranged.

Line represent attributes. Boxed nodes represent object and values of attributes of object. These can also be viewed as objects with attributes and values, and so on. The arrow on the line point from an object to its value along the corresponding attribute line, is a slot-and-filler structure. it may also be a semantic network or a collection of frames.

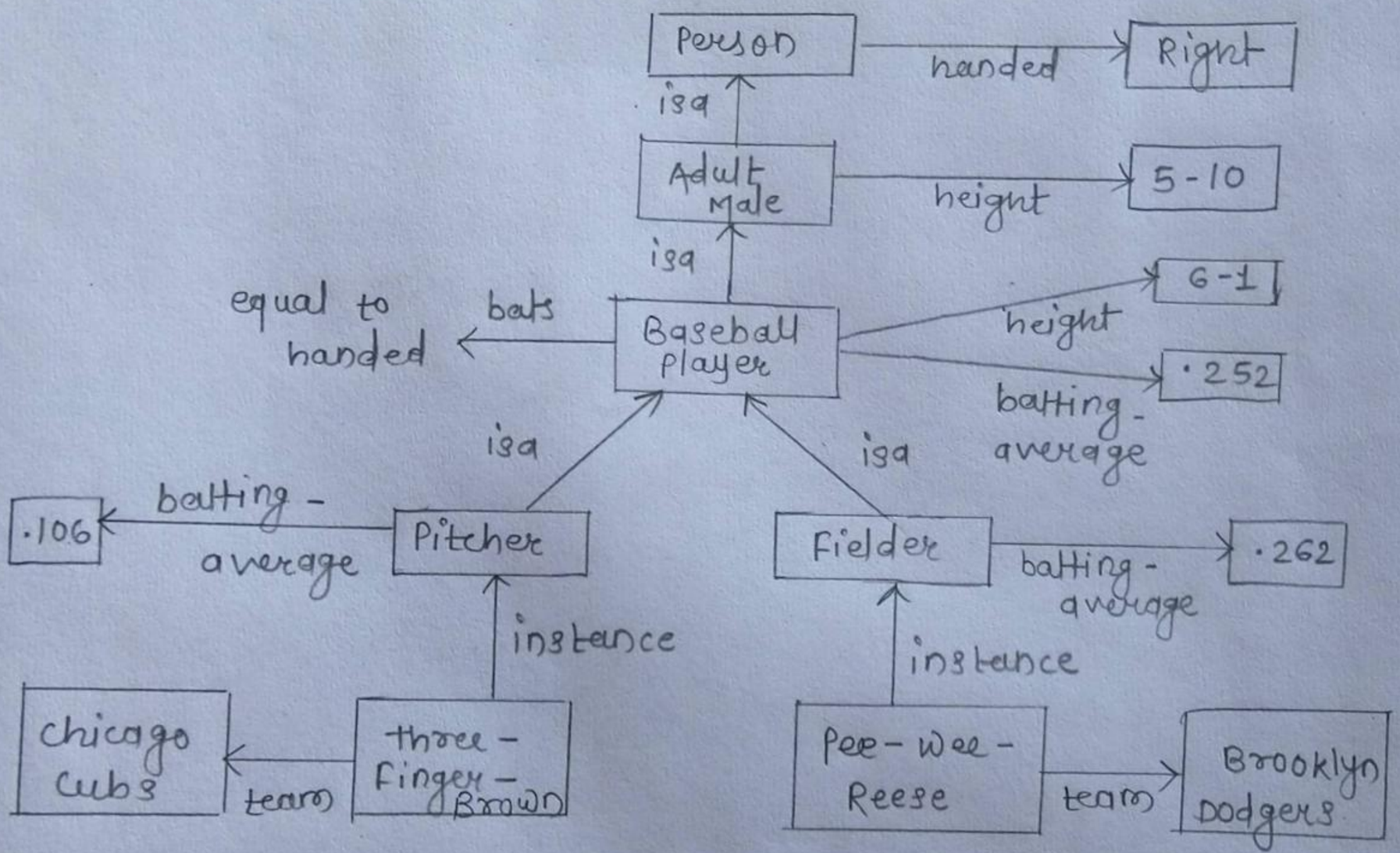


fig :- Inheritable knowldge

Each individual frame represents the collection of attributes and values associated with a particular node.

Scanned by TapScanner

3) Property Inheritance :- To retrieve a value V for attributes A of an instance object O:

1. Find O in the knowledge base
2. If there is a value there for the attributes A, report that value
3. Otherwise, see if there is a value for the attribute instance if not, then fail.
4. Otherwise, move to the node corresponding to that value and look for a value for the attributes A. if one is found, report it.
5. Otherwise, do until there is no value for the isa attribute or until an answer is found:
   a) Get the value of the isa attribute and move to that node.
   b) See if there is a value for the attributes A. if there is report it.

4) Inferential knowledge :- Property inheritance is a powerful form of inference, but it is not the only useful form. Sometimes all the power of traditional logic (and sometimes even more than that) is necessary to describe the inferences that are needed. Show two examples of the use of first-order predicate logic to represent additional knowledge about baseball.

Of course, this knowledge is useless unless there is also an inference procedure that can exploit (just as the default knowledge in the previous example would have useless without our algorithm for moving through the knowledge structure).

Scanned by TapScanner

$\forall x$: Ball $(x) \wedge$ Fly $(x) \wedge$ Fair $(x) \wedge$ Infield - catchable $(x) \wedge$
Occupied - Base (First) $\wedge$ Occupied - Base (Second) $\wedge$ (outs < 2) $\wedge$
$\neg$ [Line - Drive $(x) \vee$ Attempted - Bt, (x)]
$\rightarrow$ Infield - fly (x)

$\forall x, y$: Batter $(x) \wedge$ batted $(x, y) \wedge$ Infield - fly $(y) \rightarrow$ out (x)

Fig :- Inferential knowledge

5) Procedural knowledge :- Production rules, particularly ones that are augement with information on how they are to be used are more procedural than are the other representation methods discussed in this chapter. But making a clean distinction between declarative and procedural knowledge is difficult. Although at an intuitive level such a distinction makes some sense, at a formal level it disappears, as a discussed in section, infact as you can see the structure of the declarative knowledge of fig. The important difference is in how the knowledge is used by the procedures that knowledge.

IF : ninth inning, and
score is close, and
less than 2 outs, and
first base is vacant and
batter is better hitter than next batter,
Then : walk the batter

Fig :- Procedural knowledge as Rules

Scanned by TapScanner

5. Explain the Concept of isa hierarchy with an example.

:-

There are classes of object and specilized subset of those classes, there are attributes and specialization of attributes. Consider, for example the attributes height. it is actually a specialization of the more general attributes physical-size which is in turn, a specialization of physical-attributes. These generalization specialization relationship are important for attributes for the same reason that they are important for other concepts - they support inheritance.

- Techniques for Reasoning about value

values of attributes are specified explicitly when a knowledge base is created. We saw several examples of that in the baseball example.

* Information about the type of the value. for example, the value of height must be a number measured is a unit of length.

* Constraints on the value, often stated in term of related entities for example, the age of a person cannot be greeater than age of either of that person's parent.

* Rules for computing the value when it is needed. We showed an example of such a rule in the fig bat attributes. These rules are called backward rule. Such rules have also been called if-needed rule.

* Rules that describes action that should be taken if a value ever become known, as forward rules, or sometimes if-added rules.

Scanned by TapScanner

- Single-Valued Attributes

A specific but very useful kind of attributes is one that is guaranteed to take a unique value. for example, a baseball player can, at any one time, have only a single height and be a member of only one team. If there is already a value present for one of these attributes and a different value is asserted, then one of two things has happend.

* Introduce an explicit notation for temporal interval. If two different value are ever asserted for the same temporal interval, signal a contradiction automatically.

* Assume that the only temporal interval that is of interest is now, so if a new value is asserted, replace the old value.

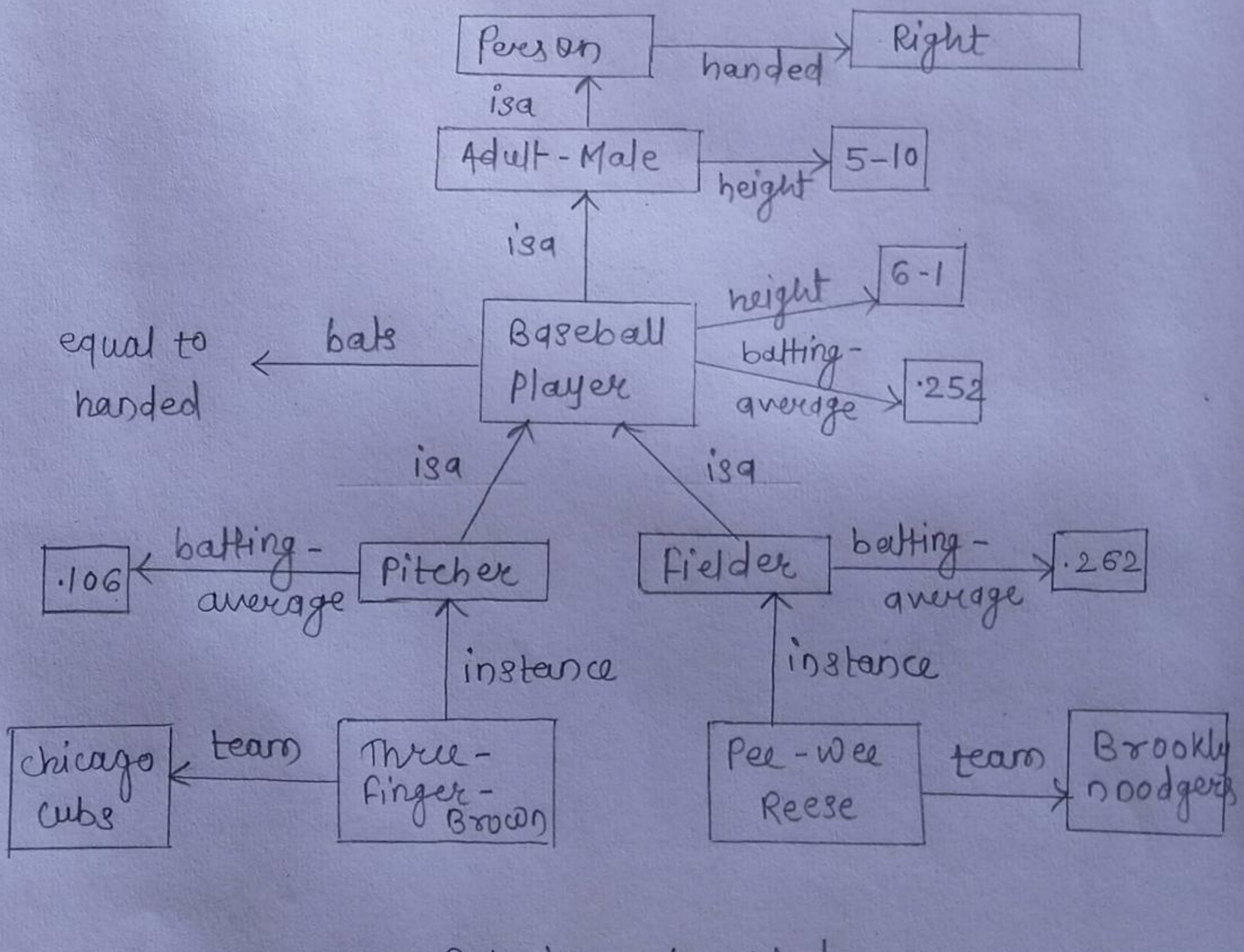


Fig :- Inheritable Knowledge

Scanned by TapScanner

6. Write a note on Contcol knowledge

:- In large knowledge bases that contain thousands of rules, the intractability of search is an overriding concern. when there are many possible path of reasoning it is critical that fruitless ones not be pursued knowledge about which paths are most likely to lead quickly to a goal state is often called search contcol knowledge. it take many forms.

1) knowledge about which states are more preferable to other.
2) knowledge about which rule to apply in a given situation.
3) knowledge about the order in which to pursue subgoal.
4) knowledge about the useful sequence of rules to apply.

The problem-solver can then use probabilistic decision analysis to choose a cost effective alternative at each point in the search. By now it should be clear that we are discussing how to represent knowledge about knowledge.

A number of AI system represent their contcol knowledge with rules. we look briefly at two such systems.
SOAR and PRODIGY.

- SOAR :- SOAR is a general architecture for building intelligent system, SOAR is based on set of specific, motivated hypothese about the structure of human problem solving. The hypothese are derived from what we know about short-term memory, Practice effect, etc. SOAR.

Scanned by TapScanner

1) Long-term memory is stored as a set of Production (or rules)
2) Short-term memory (also called working memory) is a buffer that is affected by perception and serves as a storage area for fact deduced by rules in long-term memory. Working memory is analogous to the state description in problem solving.
3) All problem-solving activities take place as state space traversal. There are several classes of problem solving activities, including reasoning about which states to explore, which rules to apply in a given situation and what effects those rules will have.
4) All intermediate and final result of problem solving are remembered (or chunked) for future reference.

The third feature is of most intrest to us here. When SOAR is given a staret state and a goal state, it sets up an initial problem space. in order to take the first step in that space, it must choose a rule form the set of application one.

SOAR also has rules for expressing a preference for applying a whole sequence of rules in a given situation. in learning mode, Soar can take useful sequence and build from them more complex production that it can apply in the future.

- PRODIGY :- PRODIGY is a genral-purpose problem solving system that incoreporate several different learning mechanism. A good deal of the learning in PRODIGY is directed at automatically constructing a set of control Rules to improve search in a Particular domain.

Scanned by TapScanner

we return to PRODIGY's learning method, but we mention here a few facts that bear on the issue of search control rule.
PRODIGY can acquire control rule in number of ways:

1) Through hand coding by programming.
2) Through a static analysis of the domain's Operator
3) Through looking at traces of its own problem-solving behavior.

PRODIGY learn controls rules from its experience, the program will then be forced to plan a repairing step or else backtrack and try working on another subgoal first. Proper search control knowledge can prevent this wasted computational effort. Rules we might consider includes.

i) If a Problem's Subgoal includes sanding and painting, the we should solve the Sanding Subgoal first.

ii) If Subgoal include sealing and painting then consider what the object is made of. If the object is made of wood, then we should seal it before painting it.

Scanned by TapScanner

7. Explain Minimax Search Procedure.

:-

The minimax search procedure is a depth-limited search procedure. the starting position is exactly as good for us the position generated by the best move we can make next.

An example of this operation is shown in fig. it assumes a static evaluation function that returns values from -10 to 10, with 10 indicating a win for us, -10 a win for the opponent and 0 an even match. Since our goal is to maximize the value of the heuristic function, we choose to move to B. Backing B's value up to A, we can conclude that A's value is 8, since we know we can move to a position with a value of 8.

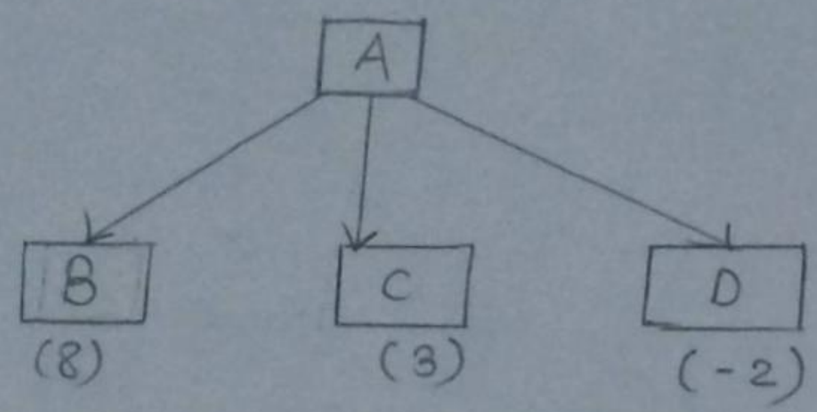


Fig :- One-Ply Search

We know that the static evaluation function is not completly accurate. we would like to carry the search farther ahead than one ply.

for example, After our move, the situation would appear to be very good, but if we look one move ahead, we will see that one of our pieces also get captured and so this situation is not as favorable as it seemed. new game positions - at the next move which will be made by the opponent.

Scanned by TapScanner

Instead of applying the static evaluation function to each of the positions that we just generated, we apply the static evaluation function to each of the positions that we just generated.

We apply the plausible-move generator, generating a set of successor position for each position. if we wanted to stop here, at two-ply lookahead. we could apply the static evaluation function to each of these positions.

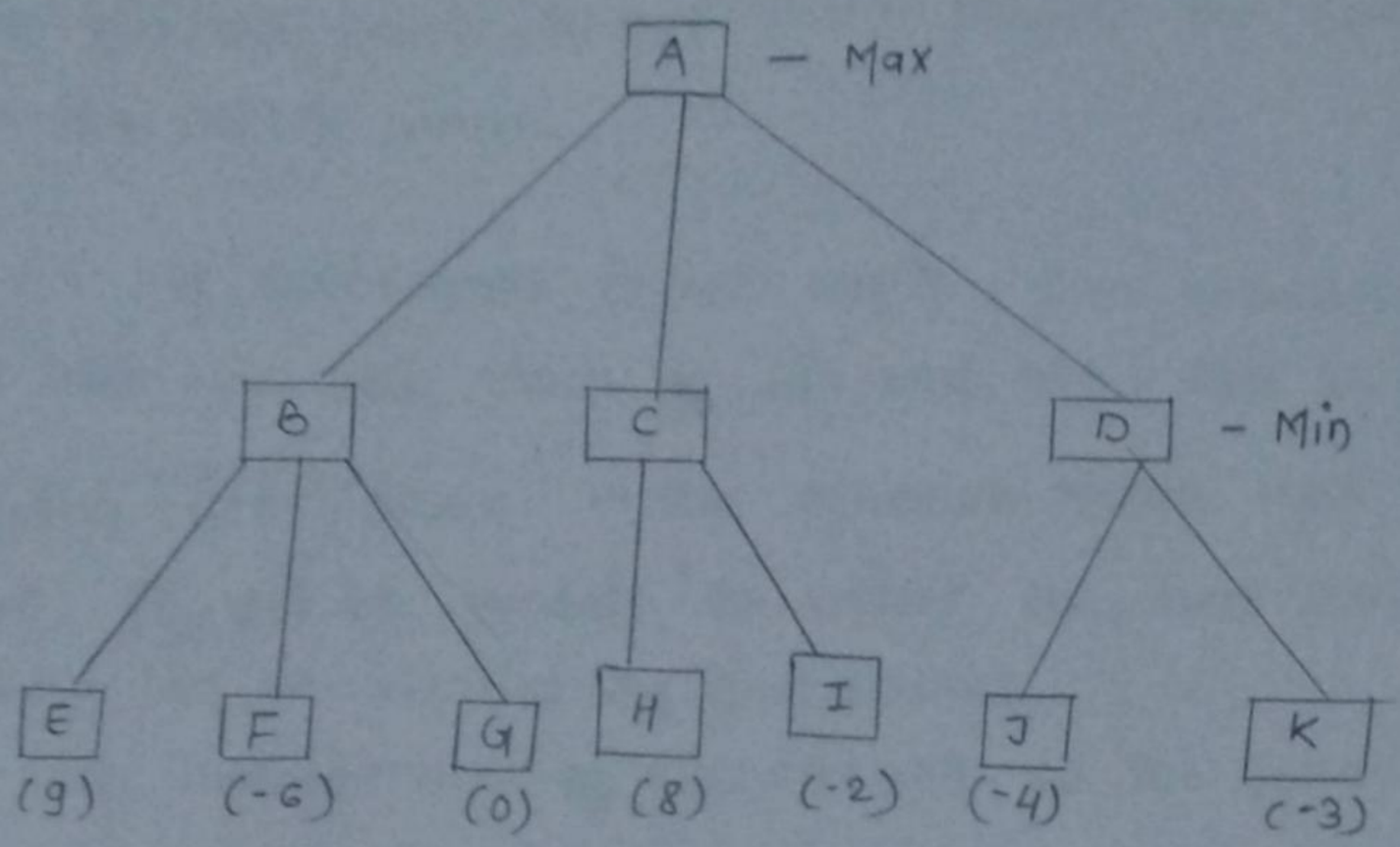


Fig :- Two-Ply Search

But now we must take into account that the opponent gets to choose which successor moves to make and thus which terminal values should be backed up to the next level.

The algorithm for MINIMAX is as follow.

Scanned by TapScanner

* Algorithm for MINIMAX

Step 1 :- If DEEP-ENOUGHE (Position, Depth), then return the structure

Value = STATIC (Position, Player)
PATH = nil

Step 2 :- Otherwise, generate one more ply of the tree by calling the function MOVE-GEN (Position player) and setting SUCCESSORS to the list it return.

Step 3 :- If SUCCESSORS is not empty, then examine each element in turn and keep track of the best one. This is done as follow

Initialize BEST-SCORE to the minimum value that STATIC can return. It will be update to reflect the best score that can be achieved by an element of SUCCESSORS.
For each element SUCC of SUCCESSORS do the following.

(a) Set RESULT-SUCC to

MINIMAX (SUCC, Depth + 1, OPPOSITE (Player))

(b) Set NEW-VALUE to -VALUE (RESULT-SUCC).

(c) If NEW-VALUE > BEST-SCORE, then we have found a successor that is better than any that have been examined so far.

i) Set BEST-SCORE to NEW-VALUE
ii) The best know path is now foem CURRENT to SUCC and then

Scanned by TapScanner

on to the appropriate path down from succ as determined by the recursive call to MINIMAX. So set BEST-PATH to the result of attaching succ to the front of PATH (RESULT-succ).

Step 5 :- Now that all the successors have been examined, we know the value of position as well as which path to take from it.

VALUE = BEST-SCORE
PATH = BEST-PATH

Scanned by TapScanner

8. What are the different components of planning System?

:- The Problem-solving System based on the elementary techniques discussed in chapter, it was necessary to perform each of the following function.

1) Choose the best rule to apply next based on the best available heuristic information.
2) Apply the chosen rule to compute the new Problem State that arises from its application.
3) Detect when a solution has been found.
4) Detect dead ends so that they can be abandoned and the System's effort directed in more fruitful directions.
5) Detect when an almost correct solution has been found and employ special techniques to make it totally correct.

* Choosing Rules to Apply

The most widely used technique for selecting appropriate rule to apply is first to isolate a set of difference between the desired goal state and the current state and then to identify thods rules that are relevant to reducing those difference. if Several rules are found, a variety of other heuristic information can be exploited to choose among them. This techniques is based on the means-end analysis method.

Scanned by TapScanner

* Applying Rules

One way to describes, for each action, each of the changes it make to the state description. In addition some statement that everything else remains unchanged is also neccessary. An example of this approach is described in Green (1969). In this system a given state was described by a set of predicates representating the fact that were true in that state.

For example, in fig shows how a state, called S0, of a simple blocks world problem could be represented.

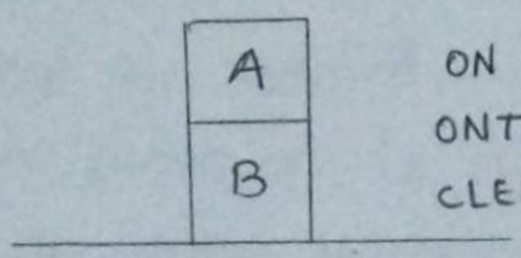


fig :- A simple Blocks world Description

* Detecting a Solution

A Planning System has succeeded in finding a solution to a problem when it has found a sequence of operator that performing the initial problem state into the goal state.

The way it can be solved depends on the way that state description are represented. for any representation scheme that is used, it must be possible to reason with representation to discover whether one matches another. Then the corresponding reasoning mechanism could be used to discover when a solution

Scanned by TapScanner

had been found.

* Detecting Deads Ends

As a Planning System is searching a sequence of operation to solve a particular problem, it must be able to detect when it is exploring a path can never leads to a solution (or at least appear unlikely to lead to one). The same reason mechanism that can be used to detect a solution can often be used for detecting a dead end.

If the search process is reasoning forword from the initial state, it can prune any path that leads to a state from which the goal state cannot be reached.

If the search process is reasoning backward from the goal state, it can also terminate a path either because it is sure that the initial state cannot be reached or because little progress is being made. in reasoning backward each goal is decomposed into subgoals.

* Repairing an Almost correct solution

The kind of techniques we are discussing are often useful in solving nearly decomposable problems. one good way of solving such problem is to assume that they are completely decomposable proceed to solve the subproblem separately and then checks that when the subsolution are combined, they do in fact yield a solution to the original problem. of course, if they do. The simplest is just to throw out the solution, look for another one and hope that it is better.

Scanned by TapScanner

A still better way to patch up incomplete solution is not really to patch them up at all but rather to leave them incomplete specified until the last possible moment. Then when as much information as possible is available, complete the specification in such a way that no conflict arise. This approach can be though of as a least-commitment strategy.

Scanned by TapScanner

9. Explain Hierarchical Planning.

:- A Problem Solver may have to genrate long plans. In order to do that efficiently, it is important to be able to eliminate some of the details of the problem until a solution that address the main issues is found. early attempts to do this involved the use of macro-operator. in which larger operator were built from smaller ones.

A better approach was developed in the ABSTRIPS system. which actually planned in a hierarchy of abstraction spaces, in each of which preconditions at a lower level of abstraction were ignored.

```
( OPERATOR
    ( PRECONDITIONS
      ( and ( ... )
        ( forall (w...) ...)
        ( not
          ( exists ...)
          ( or ... ))
    ( POSTCONDITIONS
      ( ADD ( ... ))
      ( DELETE ( ... ))
      ( if ( and ( ... )( ... ))
        ( ADD ( ... )( ... ))
        ( DELETE ( ... )( ... )))))))
```

fig :- A complex Operator

Scanned by TapScanner

The ABSTRIPS approaches to Problem-Solving is as follow:

First solve the Problem Completely, considering only Precondition whose criticality value is the highest possible. These values reflect the expected difficulty of satisfying the Precondition. once this is done. use the constructed plan as the outline of a complete plan and consider precondition at the next-lowest criticality level. continue this process of considering less and less critical Precondition until all of the precondition of the original rules have been considering. Because this process explores entire plans at one level of detail before it looks at the lower-level detail of any one of them, it has been called length-first search.

Clearly, the assignment of appropriate criticality values is crucial to the success of this hierarchical planning method. Those Preconditions that no operators can satisfy are clearly the most critical.

In order for a hierarchical planning system to work with STRIPS-like rules, it must be told, in addition to the rules themselves the appropriate criticality value for each term that may occur in a precondition. gives these values, the basic Process can process can function in very much the same way that nonhierarchical planning does.

Scanned by TapScanner

10. Explain Syntatic Processing with Grammers and Parser.

:- Syntactic Processing is the step in which a flat input sentence is converted into a hierarchical structure that corresponding to the unit of meaning in the sentence. This Process is called Parsing. Although there are natural language understanding system that skip this step, it plays an importantant role in many natural languages understanding System for two reasons.

- Semantic Processing must operate on sentence constituents. if there is no syntactic parsing step, then the semantics system must decide on its own constituents. that semantics can consider. Syntactic Parsing is computationally less expensive than is semantic Processing.

- Although it is often Possible to extract the meaning of a sentence without using grammatical fact, it is not always possible to do so. Consider for example, the sentences.

1) The Satellite Orbited Mars
2) Mars Orbited the Satellite

In second sentence, Syntactic facts demands an interpretation in which a planet (mars) resolves around a satellite, despite the apparent improbability of such a scenario.

Although there are many ways to Produce a Parse, almost all the System that are actually used too main components.

1) A declarative representation, called a grammer, of the Syntactic fact about the language.
2) A ~~Prod~~ Procedure, called a Parse, that Compare the grammar

Scanned by TapScanner

against input sentences to Produce Parsed Structure.

* Grammars and Parsers

The most common way to represent grammer is as a set of Production rule. Although details of the forms that are allowed in the rules vary the basic idea remains the same and is illustrated in fig. which shows a simple context-free, phrase structure grammer, the vertical bar should be read as "or" The ε denotes the empty string. Symbols that are further expanded by rules are called nonterminal symbol. Symbols that corresponding directly to string that must be found in an input sentence are called terminals symbols.

S → NP VP
NP → the NP1
NP → PRO
NP → NP1
NP → NP
NP1 → ADJS N
ADJS → ε | ADJ ADJS
VP → V
VP → V NP
N → File | Printer
PN → Bill
PRO → I
ADJ → short | long | fast
V → Printed | Created | Want

fig :- A Simple Grammer for a Fragment of English

Scanned by TapScanner

Grammer formalism such as this one underlie many linguistic theories, which in turn provides the basis for many natural language understanding system. modern linguistic theories includes: the government binding, theory of chomsky and categorial grammer.

The theoretical basis of the grammer, the parsing process takes the rules of the grammar and compare them against the input sentences. each rules that matchs add something to the complete structure that is being built for the sentence. The simplest structure to build is a parse tree, which simply records the rules and how they are matched in fig show the parse tree that would be produced for the sentences "Bill Printed the file" using this grammer.

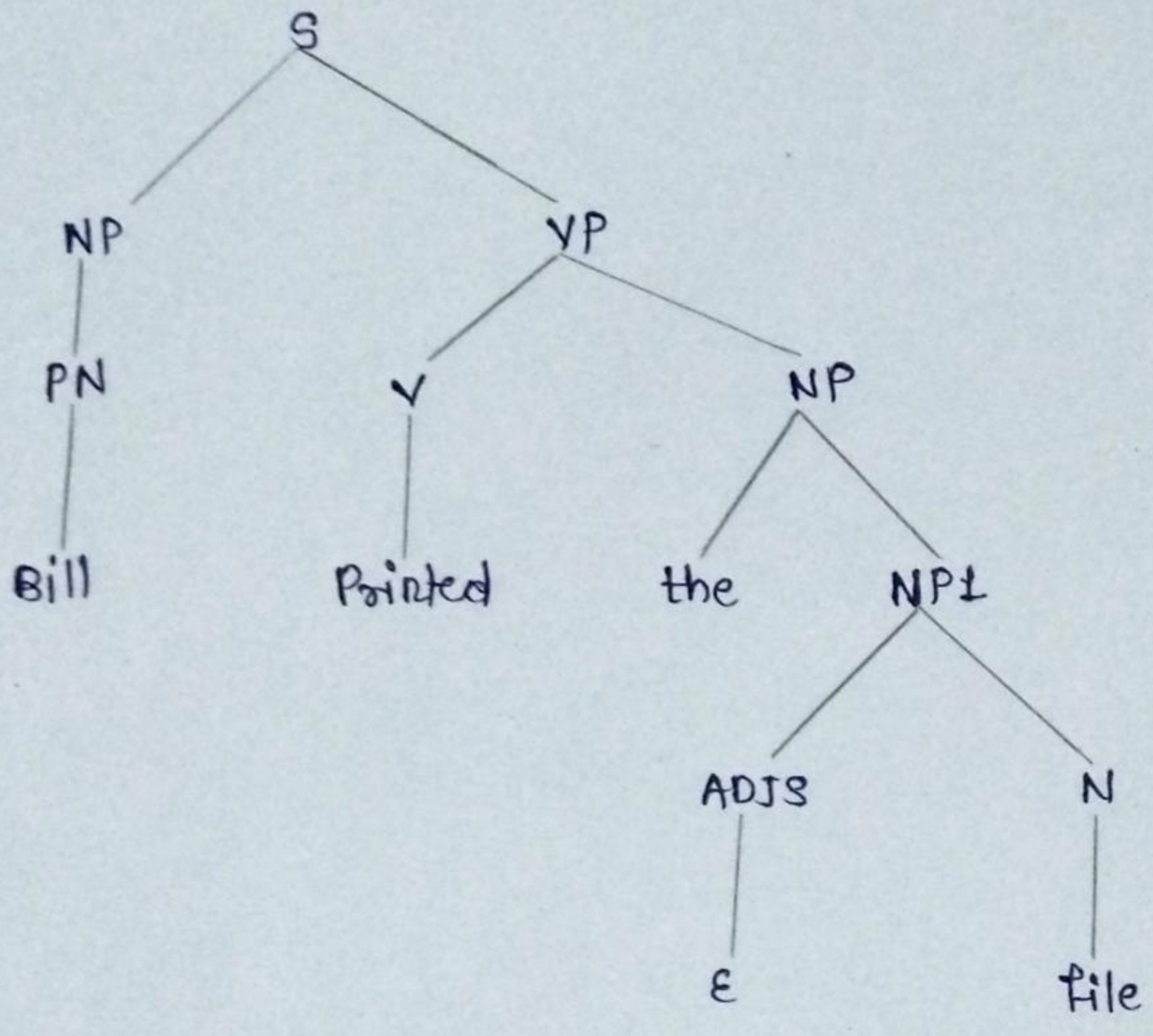


fig :- A Parse Tree for a sentence

Scanned by TapScanner

Notice that every node of the parse tree corresponds either to an input word or to a nonterminal in our grammar. each level in the parse tree corresponds to the application of one grammar rule.

Scanned by TapScanner

11. what is Distributed Reasoning System ? And what are the advantages of distributed reasoning System over Large monolithic System.

:- A distributed reasoning System to be one that is composed of a set of separate modules (often called agents since each module is usually expected to act as a problem solving entity in its own right) and a set of communication path between them. This defination is intentionally very vague. It admits System everywhere along a spectrum that range from tightly coupled System in which there is a Completely centralized control mechanism and a shared knowledge base to ones in which both control and knowledge are fully distributed.

* Distributed reasoning System have significant advantages over Large monolithic System.

1) System Modularity :- It is easier to build and maintain a collection of quasi - independent modules than one huge one.

2) Efficiency :- Not all knowledge is needed for all tasks. By modularizing it we gain the ability to focus the problem - solving system's efforts in way that are most likely to pay off.

3) Fast Computer Architecture :- As Problem - solver get more complex they need more and more cycles. Although machines continue to get faster, the real speed ups are beginning to come not from a single Processor with a huge associated memory.

4) Heterogeneous Reasoning :- The Problem solving technique and knowledge representation formalisms that are best for working on one part of a problem may not be the best for working on another part.

Scanned by TapScanner

5) Multiple Perspectives :- The knowledge required to solve a problem may not reside in the head of a single person. It is very difficult to get many diverse people to build a sigle, coherent knowledge base, and sometimes it is impossible because their models of the domain are actually inconsistent.

6) Distributed Problems :- Some problems are inherently distributed. for example, there may be different data available in each of several distinct physical locations.

7) Reliability :- If a problem is distributed across agents on different system. Problem-solving can continue even if one system fails.

Scanned by TapScanner

12. Explain Psychological Modeling.

:- The Production system was originally proposed as a model of human information processing and it continues to play a role in Phycholo-gical modeling. Some Production System model stress the sequential nature of Production System. i.e the manner in which short-term memory is modified over time by the rules.
other model stress the parallel aspect in which all production match and fire simultaneously. no matter how many there are both type of models have been used to explain timing data from experiments on human problem solving.

In the elaboration phase of the processing cycle, production fire in parallel. in the decision phase, operator and states are choosen and working memory is modified, thus setting the stage for another elaboration phase. By tying this phase to Particular timing SOAR accounts for a number of psychological phenomena.

Another approch to psychological modeling draws its inspiration from the physical organization of the human brain itself.
while individual neurons are quite slow compared to digital computer circuits, there are vast number of these richly interconnected components and they all operates concurrently.

Scanned by TapScanner