

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

External exam – Dec-Jan 2017-18
ARTIFICIAL INTELLIGENCE (10CS562) Solution by Prasad B S

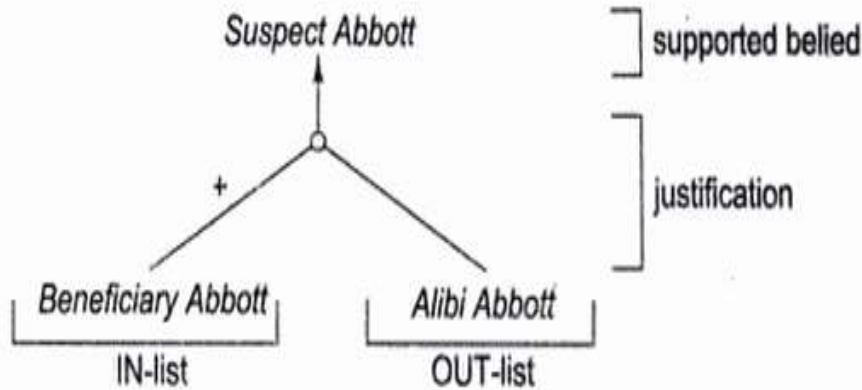
Module -1		MARK S
1 a.	State the algorithm for steepest ascent hill climbing along with its disadvantages.	[6]
Ans:	<p>Algorithm for Steepest Hill Climbing:</p> <ol style="list-style-type: none"> 1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as current state. 2. Loop until a solution is found or until a complete iteration produces no changes to current state. <ol style="list-style-type: none"> a. Let SUCC be a state such that any possible successor of the current state will be better than SUCC. b. For each operator that applies to the current state do: <ol style="list-style-type: none"> i. Apply the operator and generate a new state. ii. Evaluate the new state, If it is a goal state, then return it and quit. If not compare it to SUCC. If it is better, then set SUCC to this state. iii. If SUCC is better than current state, set current state to SUCC. <p>Drawbacks:</p> <ol style="list-style-type: none"> 1. Local Maximum/minimum 2. Plateau 3. Ridge <p>Methods to overcome these drawbacks.</p>	
1 (b)	Solve the following cryptarithmic problem SEND+MORE=MONEY	[10]
Ans.	<p>Constraint propagation</p> <ol style="list-style-type: none"> 1.M=1 since two single digit numbers plus a carry cannot total more than 19. 2.S=8 or 9 since $S+M(1)+C3(0 \text{ or } 1)>9$ 3.o=0 $S+M+C3$ can be either 10 or 11(contradicts M) 4.N=E or E+1 depends on C2, since N=E is not possible N=E+1 and C2=1. 5.For C2 to be 1 $N+R+C1 >9$, $N+R>8$, N+R cannot be greater than 18, and hence E cannot be 9. <p>Guess value of E as 2 (Appears Three times and highly interactive)</p> <ol style="list-style-type: none"> 6.N=3 since $N=E+1$ 7.R=8 or 9 since $R+N(3)+C1(1 \text{ or } 0)=2 \text{ or } 12$, since N is 3 is must be 12. 8.2+D=Y or $2+D=10+Y$ <p>Guess for C1=0</p> <ol style="list-style-type: none"> 9.2+D=Y 10.N+R=10+E 11.R=9 12.S=8 	

	$9567 + 1085 = 10652$	
2 (a)	Define Artificial Intelligence and list the task domain of artificial intelligence.	[4]
Ans:	The study of technique for solving exponentially hard problems in polynomial time by exploiting knowledge about the problem domain. Task Domains: Perception, Natural Language, Reasoning, Games, Mathematics, Engineering, Scientific Analysis, etc elaborate on few of the above	
(b)	Explain four categories of production system.	[4]
Ans:	Structure of AI programs that facilitates search process. A production system consists of : <ul style="list-style-type: none"> • A set of rules • One or more knowledge/databases that contain the appropriate information. • A control strategy • A rule applier. Types of production system: <ul style="list-style-type: none"> • Monotonic Production system: • Non-monotonic production system: • Partially commutative production system • Commutative production system example for each	
(c)	Explain problem characteristics with respect to heuristic search?	[8]
	<ol style="list-style-type: none"> 1. Is the problem decomposable into a set of independent smaller or easier problems? 2. Can solution step be ignored or at least undone if they prove unwise? 3. Is the problem's universe predictable? 4. Is a good solution the problem obvious without comparison to all other possible solutions? 5. Is the desired solution a state or a path to state. 6. Is a large amount of knowledge absolutely required to solve the problem, or is knowledge important only to constrain the search? 7. Is the solution to the problem require interaction between the computer and a person? 	
3 (a)	Explain the 'Frame problem'.	[6]
Ans.	The problem of representing the facts that change and that do not change is known as the frame problem. Any suitable example: Moving of a table on place to another.	
3 (b)	Write the algorithm for conversion to clause normal form.	[10]
Ans.	CNF: Every sentence in Propositional Logic is logically equivalent to a conjunction of disjunctions of literals. A sentence expressed as a conjunction of disjunctions of literals is said to be in Conjunctive normal Form or CNF. (AND of ORs). 1. Eliminate \rightarrow : $a \rightarrow b$ is equivalent to $\neg a \vee b$	

	<p>2. Reduce the scope of each \neg to a single term using:</p> <ol style="list-style-type: none"> 1. $\neg(\neg p) = p$; 2. $\neg(a \wedge b) = \neg a \vee \neg b$ 3. $\neg(a \vee b) = \neg a \wedge \neg b$ 4. $\neg \forall x: p(x) = \exists x: \neg p(x)$ 5. $\neg \exists x: p(x) = \forall x: \neg p(x)$ <p>3. Standardize variables so that each quantifier binds a unique variable. $\forall x: p(x) \vee \forall x: q(x) = \forall x: p(x) \vee \forall y: q(y)$</p> <p>4. Move all quantifiers to the left of the formula: $\forall x: p(x) \vee \forall y: q(y)$ can be written as $\forall x: \forall y: p(x) \vee q(y)$ (prenex normal form)</p> <p>5. Eliminate existential qualifiers: Eliminate the existential qualifier by substituting for the variable a reference to a function that produces the desired value. $\exists y: \text{principal}(y) = \text{principal}(s1(y))$ $S1$ is known as skolem function.</p> <p>6. Drop all the prefix: all remaining variables are universal quantifiers hence need not be specified. $\forall x: \forall y: p(x) \vee q(y) = p(x) \vee q(y)$</p> <p>7. Convert the matrix into a conjunction of disjuncts by using associative and distributive property: $(a \vee b) \vee c = a \vee (b \vee c)$ $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$</p> <p>8. Create a separate clause for each disjunct.</p>											
4 (a)	Define Horn Clause and give the syntactic difference between PROLOG and logic:	[4]										
	<p>Horn clause: A horn clause is a clause that has at most one positive literal.</p> <ul style="list-style-type: none"> • Ex: $p, \neg p \vee q, p \rightarrow q$ <table border="1" data-bbox="290 1192 1247 1577"> <thead> <tr> <th>LOGICAL</th> <th>PROCEDURAL</th> </tr> </thead> <tbody> <tr> <td>Variables are explicitly quantified. $\forall x, \exists x$</td> <td>Implicitly quantified. Uppercase: Variables Lowercase: constants.</td> </tr> <tr> <td>And \wedge</td> <td>,</td> </tr> <tr> <td>OR \vee</td> <td>Written as list of separate statements.</td> </tr> <tr> <td>$p \rightarrow q$</td> <td>Written backward: $q :- p$</td> </tr> </tbody> </table>	LOGICAL	PROCEDURAL	Variables are explicitly quantified. $\forall x, \exists x$	Implicitly quantified. Uppercase: Variables Lowercase: constants.	And \wedge	,	OR \vee	Written as list of separate statements.	$p \rightarrow q$	Written backward: $q :- p$	
LOGICAL	PROCEDURAL											
Variables are explicitly quantified. $\forall x, \exists x$	Implicitly quantified. Uppercase: Variables Lowercase: constants.											
And \wedge	,											
OR \vee	Written as list of separate statements.											
$p \rightarrow q$	Written backward: $q :- p$											
(b)	Write the algorithm to unify (L1,L2)	[6]										
	<p>Algorithm: Unify(L1, L2)</p> <ol style="list-style-type: none"> 1. If $L1$ or $L2$ are both variables or constants, then: <ol style="list-style-type: none"> (a) If $L1$ and $L2$ are identical, then return NIL. (b) Else if $L1$ is a variable, then if $L1$ occurs in $L2$ then return {FAIL}, else return $(L2/L1)$. (c) Else if $L2$ is a variable then if $L2$ occurs in $L1$ then return {FAIL}, else return $(L1/L2)$. (d) Else return {FAIL}. 											

	<ol style="list-style-type: none"> 2. If the initial predicate symbols in $L1$ and $L2$ are not identical, then return {FAIL}. 3. If $L1$ and $L2$ have a different number of arguments, then return {FAIL}. 4. Set $SUBST$ to NIL. (At the end of this procedure, $SUBST$ will contain all the substitutions used to unify $L1$ and $L2$.) 5. For $i \leftarrow 1$ to number of arguments in $L1$: <ol style="list-style-type: none"> (a) Call Unify with the ith argument of $L1$ and the ith argument of $L2$, putting result in S. (b) If S contains FAIL then return {FAIL}. (c) If S is not equal to NIL then: <ol style="list-style-type: none"> (i) Apply S to the remainder of both $L1$ and $L2$. (ii) $SUBST := APPEND(S, SUBST)$. 6. Return $SUBST$. 	
(c)	Write a note on conflict resolution.	[6]
	<p>The result of the matching process is a list of rules whose antecedents have matched the current state description along with whatever variable bindings were generated by the matching process. It is the job of the search method to decide on the order in which rules will be applied. But sometimes it is useful to incorporate some of that decision making into the matching process. This phase of the matching process is then called <i>conflict resolution</i>.</p> <p>There are three basic approaches to the problem of conflict resolution in a production system:</p> <ul style="list-style-type: none"> • Assign a preference based on the rule that matched. • Assign a preference based on the objects that matched. • Assign a preference based on the action that the matched rule would perform. 	
5 (a)	Define Frame. State the Bayes Theorem and explain the notations used.	[6]
	<ul style="list-style-type: none"> ▪ A frame is a collection of attributes (slots) and associated values that describe some entity in the world. ▪ Uses set theory as its basis. ▪ A frame system is collection of frames that are connected to each other by virtue of the fact that value of attribute of one frame may be another frame. ▪ Each frame represents either a class or an instance. <p>Baye's Theorem</p> <ul style="list-style-type: none"> ▪ To compute conditional probability, we need to take into account the prior probability of H (the probability that we would assign to H if we had no evidence) & the extent to which E provides evidence of H. ▪ To do this we need to define a universe that contains an exhaustive, mutually exclusive set of H_i's , among which we are trying to discriminate. ▪ $P(H_i E)$ – The probability that hypothesis H_i is true given evidence E ▪ $P(E H_i)$ – The probability that we will observe evidence E given that hypothesis i is true. ▪ $P(H_i)$ – The a priori probability that hypothesis I is true in the absence of any specific evidence. These probabilities are called prior probabilities or priors. ▪ K = The number of possible hypothesis. $P(H_i E) = \frac{P(E H_i) P(H_i)}{\sum_{k=1}^n P(E H_k)P(H_k)}$	
(b)	Write a note on Justification Based Truth Maintenance System(JTMS)	[10]
	<ul style="list-style-type: none"> • Also known as Reason maintenance system(RMS). • Allows assertion to be connected via a spreadsheet-like network of dependencies serving as a bookkeeper. 	

- Purely syntactic , domain independent way to represent beliefs and change it consistently.



6 (a) Write a note on closed world assumption.

[6]

There are many fewer true statements than false ones. If something is true and relevant it makes sense to make it part of the knowledge base. Therefore assume that the only true statements are those that necessarily must be true in order to maintain the consistency of the knowledge base

CWA: Presumption that a statement that is true is also known to be true. The closed-world assumption (CWA), in a formal system of logic used for knowledge representation, is the presumption that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false.

(b) Explain Bayesian network.

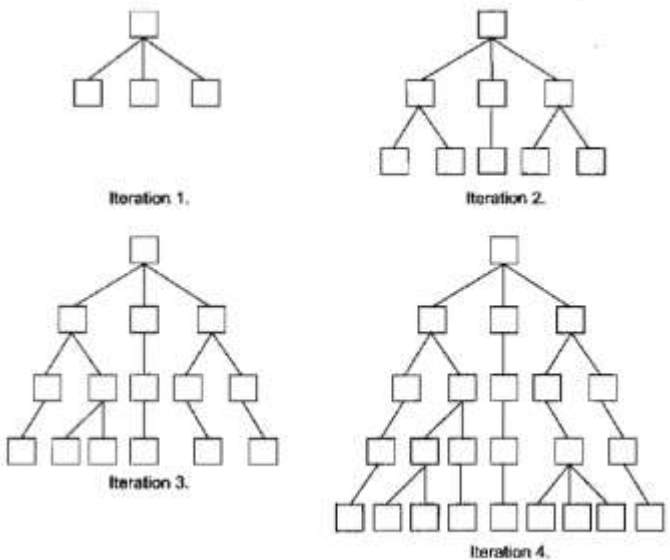
[10]

- Reduces the complexity of a Bayesian reasoning by relying on modularity of the world, it tries to describe cluster of events that interact.
- Directed acyclic graph (DAG) represents causality relationship among variables.

Attribute	Probability
$p(\text{Wet} \setminus \text{Sprinkler}, \text{Rain})$	0.95
$P(\text{Wet} \setminus \text{Sprinkler}, \neg \text{Rain})$	0.9
$p(\text{Wet} \setminus \neg \text{Sprinkler}, \text{Rain})$	0.8
$p(\text{Wet} \setminus \neg \text{Sprinkler}, \neg \text{Rain})$	0.1
$p(\text{Sprinkler} \setminus \text{RainySeason})$	0.0
$p(\text{Sprinkler} \setminus \neg \text{RainySeason})$	1.0
$p(\text{Rain} \setminus \text{RainySeason})$	0.9
$p(\text{Rain} \setminus \neg \text{RainySeason})$	0.1
$p(\text{RainySeason})$	0.5

Fig. 8.3 Conditional Probabilities for a Bayesian Network

7 (a)	Define conceptual dependency, mention its goals along with its representation.	[6]
	<p>It is a theory of how to represent the kind of knowledge about events that is usually contained in Natural language sentences.</p> <ol style="list-style-type: none"> 1. The goal is to represent the knowledge in a way that 2. Facilitates drawing inferences from the sentences. <p>Is independent of the language in which the sentences were originally stated. CD representation of a sentence is built not out of primitives corresponding to the words used in the sentence, but rather out of conceptual primitives that can be combined to form the meanings of words in any particular language.</p> <p style="text-align: center;"> $I \overset{p}{\leftrightarrow} \text{ATRANS} \overset{o}{\leftarrow} \text{book} \overset{R}{\leftarrow} \begin{array}{l} \text{to} \rightarrow \text{man} \\ \text{from} \leftarrow I \end{array}$ </p> <p>where the symbols have the following meanings:</p> <ul style="list-style-type: none"> • Arrows indicate direction of dependency. • Double arrow indicates two way link between actor and action. • p indicates past tense. • ATRANS is one of the primitive acts used by the theory. It indicates transfer of possession. • o indicates the object case relation. • R indicates the recipient case relation. <p style="text-align: center;">Fig. 10.1 A Simple Conceptual Dependency Representation</p>	
(b)	Write the algorithm for minmax (position, depth,players) and explain.	[10]

	<p>Algorithm: MINIMAX(Position, Depth, Player)</p> <ol style="list-style-type: none"> If DEEP-ENOUGH(Position, Depth), then return the structure VALUE = STATIC(Position, Player); PATH = nil This indicates that there is no path from this node and that its value is that determined by the static evaluation function. Otherwise, generate one more ply of the tree by calling the function MOVE-GEN(Position Player) and setting SUCCESSORS to the list it returns. If SUCCESSORS is empty, then there are no moves to be made, so return the same structure that would have been returned if DEEP-ENOUGH had returned true. If SUCCESSORS is not empty, then examine each element in turn and keep track of the best one. This is done as follows. Initialize BEST-SCORE to the minimum value that STATIC can return. It will be updated to reflect the best score that can be achieved by an element of SUCCESSORS. For each element SUCC of SUCCESSORS, do the following: <ol style="list-style-type: none"> Set RESULT-SUCC to MINIMAX(SUCC, Depth + 1, OPPOSITE(Player)) This recursive call to MINIMAX will actually carry out the exploration of SUCC. Set NEW-VALUE to - VALUE(RESULT-SUCC). This will cause it to reflect the merits of the position from the opposite perspective from that of the next lower level. If NEW-VALUE > BEST-SCORE, then we have found a successor that is better than any that have been examined so far. Record this by doing the following: <ol style="list-style-type: none"> Set BEST-SCORE to NEW-VALUE. The best known path is now from CURRENT to SUCC and then 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). Now that all the successors have been examined, we know the value of Position as well as which path to take from it. So return the structure VALUE = BEST-SCORE PATH = BEST-PATH 	
8 (a)	<p>Write the algorithm for</p> <ol style="list-style-type: none"> Depth first iterative deepening Iterative deepening A* <p>Iterative deepening</p>  <p style="text-align: center;">Fig. 12.10 Iterative Deepening</p> <p>Algorithm: Iterative-Deepening-A*</p> <ol style="list-style-type: none"> Set THRESHOLD = the heuristic evaluation of the start state. Conduct a depth-first search, pruning any branch when its total cost function ($g + h'$) exceeds THRESHOLD. If a solution path is found during the search, return it. Otherwise, increment THRESHOLD by the minimum amount it was exceeded during the previous step, and then go to Step 2. 	[6]
(b)	Write a note on global ontology.	[10]

	<ol style="list-style-type: none"> 1. Ontology: Philosophical study of what exists. In AI it is concerned with which categories can usefully quantify over and how those categories relate to each other. 2. Global ontology: Specifies at a very high level what kind of things exist and what are their general properties are. 3. Highest level concept in CYC is called 'Thing'. Everything is an instance of Thing. Distinction below this level are: 4. IndividualObject Vs Collection: <ul style="list-style-type: none"> ◦ Corresponds to Class CLASS. Instances of Collection are: Nations, Person, Cars ◦ Instance of IndividualObject are India, Ram , BMW 5. Intangible, Tangible and composite: <ul style="list-style-type: none"> ◦ Intangible are things without mass. Ex: sets, numbers, laws ◦ Tangible objects with mass. Ex: An orange, a table, Chalk ◦ CompositeObject: has two slots physicalExtent (tangible aspect) , intangible extent. Ex: Body and Mind 6. Substance: Substance is a subclass of IndividualObject. Any subclass of Substance is something that retains its properties when it is cut up into smaller pieces. Ex: wood 7. Intrinsic vs Extrinsic properties: A property is intrinsic if when an object has that property all parts of the object also have that property. (property of substance). 8. Events and process: An event is anything with temporal extent. Process is a subclass of event. 9. Slots: Slots is a subclass of Intangible. There are many types of slots. 10. Time: Events can have temporal properties such as duration and startsBefore. 11. Agents: An important subset of CompositeObject is Agent, collection intelligent beings. Agents can be collective(corporation, religion) or own individual. 	
9 (a)	Explain classification of spell checking techniques.	[6]
	<p>Classification of spelling errors:</p> <ol style="list-style-type: none"> 1. Typographic errors: Errors caused due to mistakes during typing. Ex: Netwrkinstead of network 2. Orthographicherrors: Error due to lack of comprehension of the concerned language on part of the user. Ex: wellcome, accommodation 3. Phonetic errors: Poor cognition on part of the listener. 'Rough' spelled as 'Ruff', 'Piece' spelt as 'Peace' or 'peas' 	
(b)	Explain knowledge aquisitions.	[10]
	<ul style="list-style-type: none"> ▪ Knowledge acquisition -- one can learn by experience and by storing the experience in a knowledge base. One basic example of this type is rote learning. ▪ Rote Learning is basically <i>memorisation or Caching..</i> <ul style="list-style-type: none"> ▪ Saving knowledge so it can be used again. ▪ Retrieval is the only problem. ▪ Store computed values (or large piece of data) ▪ Recall this information when required by computation. No repeated computation, inference or query is necessary. ▪ Significant time savings can be achieved. 	

	<ul style="list-style-type: none"> ▪ Many AI programs (as well as more general ones) have used caching very effectively. ▪ Capabilities of Rote learning: <ul style="list-style-type: none"> ▪ Organisation -- access of the stored value must be faster than it would be to recompute it. Methods such as hashing, indexing and sorting can be employed to enable this. ▪ Generalisation -- The number of potentially stored objects can be very large. May need to generalise some information to make the problem manageable. 	
10 (a)	Briefly explain four ways of handling sentences.	[4]
(b)	Write a note on decision trees.	[6]
	<ul style="list-style-type: none"> ▪ Basically each leaf of a <i>decision tree</i> asserts a positive or negative concept. To classify a particular input we start at the top and follow assertions down until we reach an answer. ▪ Quinlan in his ID3 system (1986) introduced the idea of decision trees. ▪ Decision tree advantages: <ul style="list-style-type: none"> ▪ Quicker than version spaces when concept space is large. ▪ Disjunction easier. ▪ Disadvantages: <ul style="list-style-type: none"> ▪ Representation not natural to humans -- a decision tree may find it hard to explain its classification. <div style="text-align: center; margin-top: 20px;"> <pre> graph TD Root([Income range of applicant?]) -- "< \$30K" --> Node1([Criminal record?]) Root -- "\$30-70K" --> Node2([Years in present job?]) Root -- "> \$70K" --> Node3([Criminal record?]) Node1 -- yes --> Leaf1([loan]) Node1 -- no --> Leaf2([no loan]) Node2 -- "< 1" --> Leaf3([no loan]) Node2 -- "1-5" --> Leaf4([loan]) Node2 -- "> 5" --> Node4([Makes credit card payments?]) Node4 -- yes --> Leaf5([loan]) Node4 -- no --> Leaf6([no loan]) Node3 -- no --> Leaf7([loan]) Node3 -- yes --> Leaf8([no loan]) </pre> </div>	
(c)	Write the algorithm for candidate elimination.	[6]

	<p>Algorithm: Candidate Elimination</p> <p>Given: A representation language and a set of positive and negative examples expressed in that language. Compute: A concept description that is consistent with all the positive examples and none of the negative examples.</p> <ol style="list-style-type: none">1. Initialize G to contain one element: the null description (all features are variables).2. Initialize S to contain one element: the first positive example.3. Accept a new training example. If it is a <i>positive example</i>, first remove from G any descriptions that do not cover the example. Then, update the S set to contain the most specific set of descriptions in the version space that cover the example and the current elements of the S set. That is, generalize the elements of S as little as possible so that they cover the new training example. If it is a <i>negative example</i>, first remove from S any descriptions that cover the example. Then, update the G set to contain the most general set of descriptions in the version space that <i>do not</i> cover the example. That is, specialize the elements of G as little as possible so that the negative example is no longer covered by any of the elements of G.4. If S and G are both singleton sets, then if they are identical, output their value and halt. If they are both singleton sets but they are different, then the training cases were inconsistent. Output this result and halt. Otherwise, go to step 3.	
--	--	--