USN | | | | | | | | | | |

Internal Assessment Test 2 – Dec 2021

Sub:	Introduction to Artificial Intelligence					Sub Code:	18CS753	Branch:	ECE/EEE/ME/CIV
Date:	21/12/21	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VII		OBE

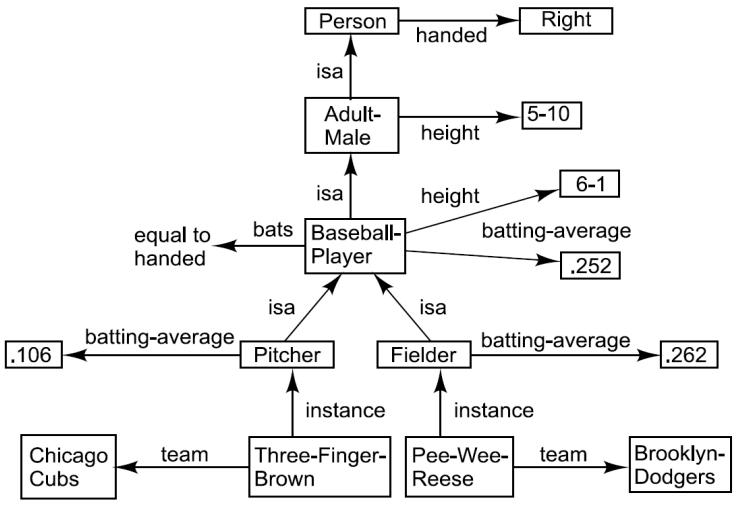
Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	Explain knowledge representation using inheritance.	[10]	CO2	L1
2.	State and explain unification algorithm Unify(L1,L2).	[10]	CO2	L2
3.	Write the algorithm for conversion of a first order logic formula to clause form.	[10]	CO2	L3
4a.	Explain Resolution Algorithm for First Order Logic.	[06]	CO2	L2
4b.	Given the following axioms: P (PandQ)->R (SorT)->Q T Where P,Q,R,S,T are propositions, Prove that R is true using Resolution.	[04]	C02	L3
5 (a)	Explain the limitations of close world assumption.	[10]	CO2	L1
(b)	Explain Bayes Theorem.	[04]	CO2	L1
6	Explain justification based truth maintenance system.	[10]	CO2	L1

Scheme and Solutions

Sub:	Introduction to Artificial Intelligence	Sub Code:	18CS753	Branch:	ECE/EEE/ME/CIV

		MARKS	CO	RBT
1.	Explain knowledge representation using inheritance.	[10]		



1. Find *O* in the knowledge base.
2. If there is a value there for the attribute 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 attribute *A.* If one is found, report it.
5. Otherwise, do until there is no value for the *is a* attribute or until an answer is found:

(a) Get the value of the *is a* attribute and move to that node.

(b) See if there is a value for the attribute *A.* If there is, report it.

2. State and explain unification algorithm Unify(L1,L2). [10]

If *L*1 or *L*2 are both variables or constants, then:

(a) If *L*1 and *L*2 are identical, then return NIL.

(b) Else if *L*1 is a variable, then if *L*1 occurs in *L*2 then return {FAIL}, else return (*L*2/*L*1).

(c) Else if *L*2 is a variable then if *L*2 occurs in *L*1 then return {FAIL}, else return (*L*1/*L*2).

(d) Else return {FAIL}.

2. If the initial predicate symbols in *L*1 and *L*2 are not identical, then return {FAIL).

3. If *L*I and *L*2 have a different number of arguments, then return {FAIL}.

4. Set *SUBST* to NIL.

5. For $i \leftarrow 1$ to number of arguments in *L*1:

(a) Call Unify with the /th argument of *L*1 and the *i*th argument of *L*2, putting result in *S.*

(b) If *S* contains FAIL then return {FAIL}.

(c) If *S* is not equal to NIL then:

(i) Apply *S* to the remainder of both *L*1 and *L*2.

(ii) *SUBST :* = APPEND(S, *SUBST*).

6. Return *SUBST.*

3. Write the algorithm for conversion of a first order logic formula to clause form. [10]

Algorithm : Convert to Clause Form

1. Eliminate $\rightarrow$, using: $a \rightarrow b = \neg a \vee b$.
2. Reduce the scope of each $\neg$ to a single term, using:
   - $\neg(\neg p) = p$
   - deMorgan's laws: $\neg(a \wedge b) = \neg a \vee \neg b$, $\neg(a \vee b) = \neg a \wedge \neg b$
   - $\neg\forall x : P(x) = \exists x : \neg P(x)$
   - $\neg\exists x : P(x) = \forall x : \neg P(x)$
3. Standardize variables.
4. Move all quantifiers to the left of the formula without changing their relative order.
5. Eliminate existential quantifiers by inserting Skolem functions.
6. Drop the prefix.
7. Convert the matrix into a conjunction of disjuncts, using associativity and distributivity.
8. Create a separate clause for each conjunct.
9. Standardize apart the variables in the set of clauses generated in step 8, using the fact that
   $(\forall x : P(x) \wedge Q(x)) = \forall x : P(x) \wedge \forall x : Q(x)$

4a. Explain Resolution Algorithm for First Order Logic. [06]

Convert all the statements of *F* to clause form.

2. Negate *P* and convert the result to clause form. Add it to the set of clauses obtained in 1.

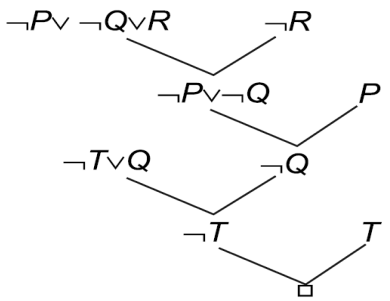
3. Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.

Select two clauses. Call these the parent clauses.

Resolve them together. The resolvent will be the disjunction of all the literals of both parent clauses with appropriate substitutions performed and with the following exception: If there is one pair of literals *T* 1 and ¬*T* 2 such that one of the parent clauses contains *T* 2 and the other contains *T* 1 and if *T* 1 and *T* 2 are unifiable, then neither *T* 1 nor *T* 2 should appear in the resolvent. If there is more than one pair of complimentary literals, only one pair shold be omitted from the resolvent.

If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

4b. Given the following axioms: [04]
P
(PandQ)->R
(SorT)->Q
T
Where P,Q,R,S,T are propositions, Prove that R is true using Resolution.



5 (a) Explain the limitations of close world assumption [05]
.

The CWA says that only objects that satisfy any predicate are those that must.

CWA can fail to produce an appropriate answer for either of the two reasons:

a) The assumptions are not true in the real world. CWA will yeild appropriate results exactly to the extent that the assumptions that all the relevant positive facts are present in the knowledge base is true.

b) CWA is purely syntactic reasoning process.

A(Joe) OR B(Joe)
We can include NOT A(Joe)
We cannot include NOT B(Joe), as that would lead to a contradiction.

5 (b) Explain Bayes Theorem. [05]

$P(H_i \backslash E)$ = the probability that hypothesis H_i is true given evidence E
$P(E \backslash H_i)$ = 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 *prlors*.
k = the number of possible hypotheses

Bayes' theorem then states that

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

6 Explain justification based truth maintenance system. [10]

If we want to derive a fact F, which cannot be derived monotonically from what we know, then we derive F on the basis of an assumption A.

Later a new fact comes, which invalidates A.

In non-monotonic reasoning, it is necessary to have dependency based backtracking

Dependency based backtracking is implemented by adding justifications at each node.

Each justifiication corresponds to the process that led to the node.

A TMS provides a problem solver system with both assertions and dependencies among assertions.

Each justification consists of two parts: IN list and OUT list

Assertions in IN list are connected to justifications by a +
Assertions in OUT list are connected to justifications by a -

Assertions in a TMS network are believed, when they have valid justification.
A justification is valid, if every assertion in the IN list is believed and none in the OUT list is.