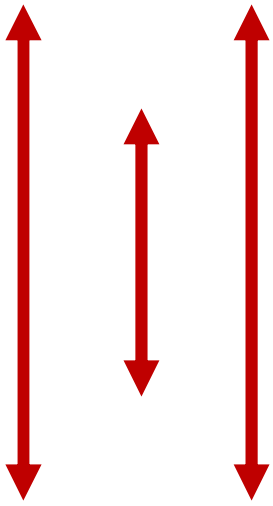
POKHARA UNIVERSITY

EXAM PAPER SOLUTION

ARTIFICIAL INTELEGENCE

CMP 421

Bachelor of Computer Engineering



By: Mr. Deepak Bhatta (Lecturer)

National Academy of Science and Technology
DHANGADHI ENGINEERING COLLEGE
Uttar Behadi, Dhangadhi-4, Kailali
(NEPAL)

[Fall 2012]

Q.4.a) The logical operator "←→" is read "if and only if". P←→Q is defined as being equivalent to (P→Q) ^ (Q→P). Based on the definition, show that P←→Q is logically equivalent to (P v Q)→(P ^ Q) using truth table.

Solⁿ:

Before Solving First of all Refresh your knowledge what you have studied.

1. Conditional Statement:

Let P and Q be propositions.

- The conditional statement P → Q, is the proposition

  "if P, then Q."

- The truth value of P → Q is **false** if P is **true** and Q is **false**. Else, it is **true**.
- In the proposition P → Q,
  - P is called the hypothesis, or the premise
  - Q is called the conclusion
- Many arguments in mathematical reasoning involve conditional statements, and there are many equivalent ways to express P → Q
  - "P implies Q" "P only if Q" "Q if P" "Q follows from P"
  - "P is sufficient for Q" "Q is necessary for P"
  - "a sufficient condition for Q is P"
  - "a necessary condition for P is Q"
- **Truth Table:**

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

  - **Example:** If it is raining outside, then the road is wet.

Note: It is not very natural, from English point of view

- Three special propositions are related to the conditional statement P → Q

  1. The **converse** of P → Q: Q ← P
  2. The **contrapositive** of P → Q: ~Q → ~P
  3. The **inverse** of P → Q: ~P → ~Q

2. Bi-Conditional Statement:

Let p and q be propositions.

- The bi-conditional statement **P←→Q**, is the proposition

  "P if and only if Q."

- The truth value of **P←→Q** is **true** if P and Q have the **same truth value**. Else, **false**.
- The biconditional statement is equivalent to $(\boldsymbol{p} \rightarrow \boldsymbol{q}) \wedge (\boldsymbol{q} \rightarrow \boldsymbol{p})$.
- **Truth Table:**

P	Q	P←→Q
T	T	T
T	F	F
F	T	T
F	F	T

  - **Example:** If it is raining outside if and only if the road is wet.

P	Q	P←→Q	(P v Q)	(P ^ Q)	(P v Q)→(P ^ Q)
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	F	F
T	T	T	T	T	T

≡

Q.4.b) Represent the following using First Order Logic.

i. Prasant likes easy courses.
ii. Science courses are hard.
iii. All the courses in the computer departments are easy.
iv. AI is a computer course.

Also, use **resolution** to answer the question. "What course would Prasant like?"

Solⁿ:

i) Prasant likes easy courses.

FOL: $\forall X: Easy(X) \rightarrow Likes(prasant, X)$

CNF: $\neg Easy(X) \vee Likes(prasant, X)$

ii) Science courses are hard.

FOL: $Hard\ (science) \equiv \neg Easy(science)$

CNF: $\neg Easy(science)$

iii) All the courses in the computer departments are easy.

FOL: $\forall Y: Computer(Y) \rightarrow Easy(Y)$

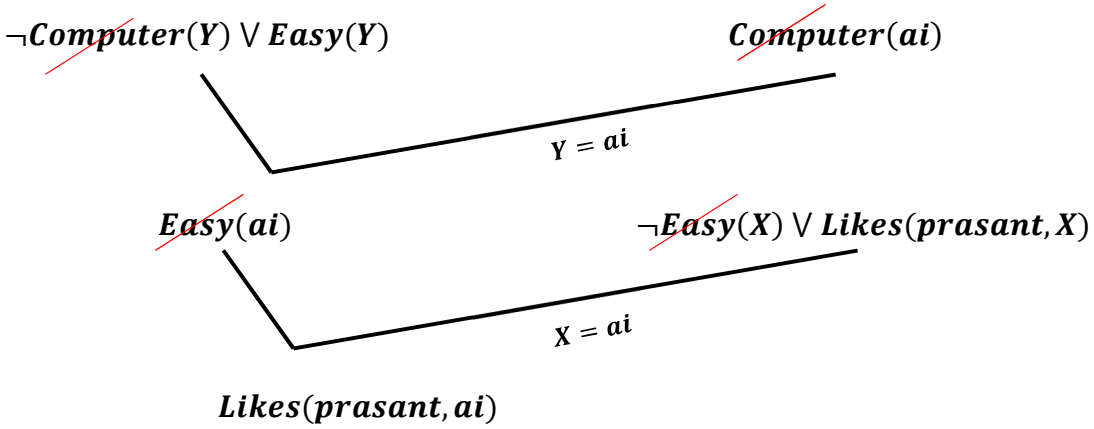
CNF: $\neg Computer(Y) \vee Easy(Y)$

iv) AI is computer course.

FOL: $Computer(ai)$

CNF: $Computer(ai)$

The Resolution procedure has been illustrated below:



At last we got $Likes(prasant, ai)$

Thus, **Prasant likes AI** is the final result.

OR

Q.4.b) Prove that **"jeevan is happy"** with the help of following facts expressed in CNF.

i) $\neg pass(X, history) \vee \neg win(X, lottery) \vee happy(X)$

ii) $\neg study(X) \vee pass(Y, Z)$

iii) $\neg lucky(W) \vee pass(W, U)$

iv) $\neg study(jeevan)$

$lucky(jeevan)$

v) $\neg lucky(U) \vee win(U, lottery)$

Sol^n: To prove that **"jeevan is happy"** we first go with the contradiction means it is **FALSE.**

Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"jeevan is not happy":** $\neg Happy(jeevan)$

We have given facts in CNF:

i) $\neg Pass(X, history) \vee \neg Win(X, lottery) \vee Happy(X)$

ii) $\neg Study(X) \vee Pass(Y, Z)$

iii) $\neg Lucky(W) \vee Pass(W, U)$

iv) $\neg Study(jeevan)$

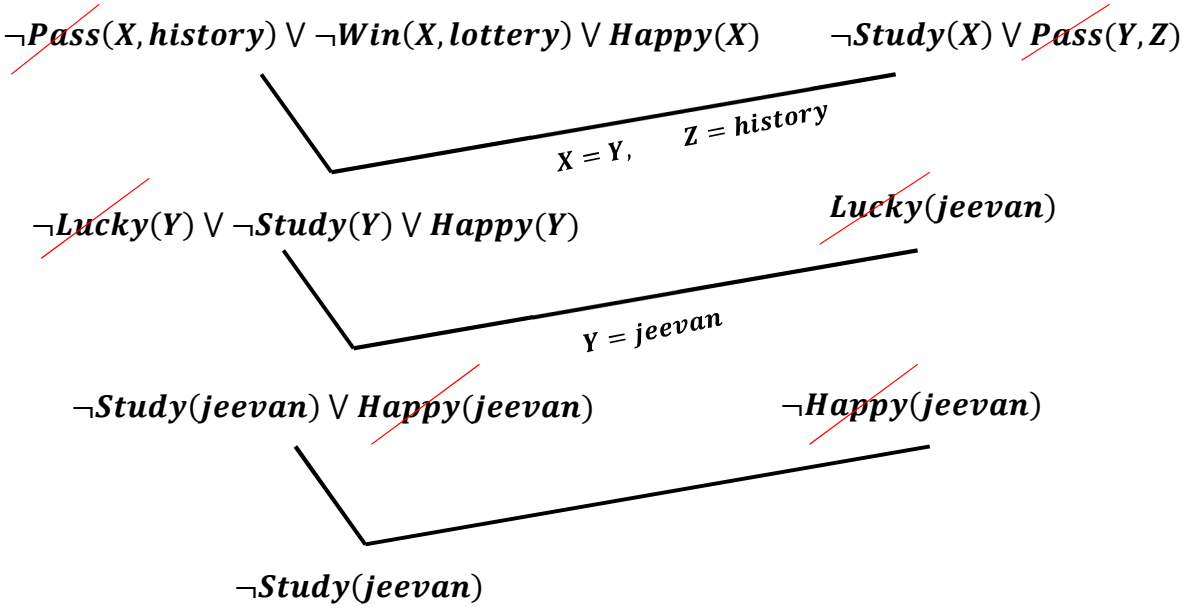
v) $Lucky(jeevan)$

vi) $\neg Lucky(U) \vee Win(U, lottery)$

Negation of Conclusion:

vii) $\neg Happy(jeevan)$

The Resolution procedure has been illustrated below:



Here, we couldn't get the **EMPTY** clause, means it shows that the $\neg Happy(jeevan)$ is true. This will not produce a contradiction. So, we can say that **"jeevan is not happy".**

Thus, the given statement **"jeevan is happy"** is not proved as per the given facts.

Slightly changes on above question to get it proved

Q.4.b) Prove that **"jeevan is happy"** with the help of following facts expressed in CNF.

i) $\neg Pass(X, history) \vee \neg Win(X, lottery) \vee Happy(X)$

ii) $\neg Study(X) \vee Pass(Y, Z)$

iii) $\neg Lucky(W) \vee Pass(W, U)$

iv) $Study(jeevan)$ // **I have removed the negation** $\neg Lucky(jeevan)$

v) $\neg Lucky(U) \vee Win(U, lottery)$

Sol^{n}: To prove that **"jeevan is happy"** we first go with the contradiction means it is **FALSE.** Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"jeevan is not happy":** $\neg Happy(jeevan)$

We have given facts in CNF:

i) $\neg Pass(X, history) \vee \neg win(X, lottery) \vee happy(X)$

ii) $\neg Study(X) \vee pass(Y, Z)$

iii) $\neg Lucky(W) \vee pass(W, U)$

iv) $Study(jeevan)$

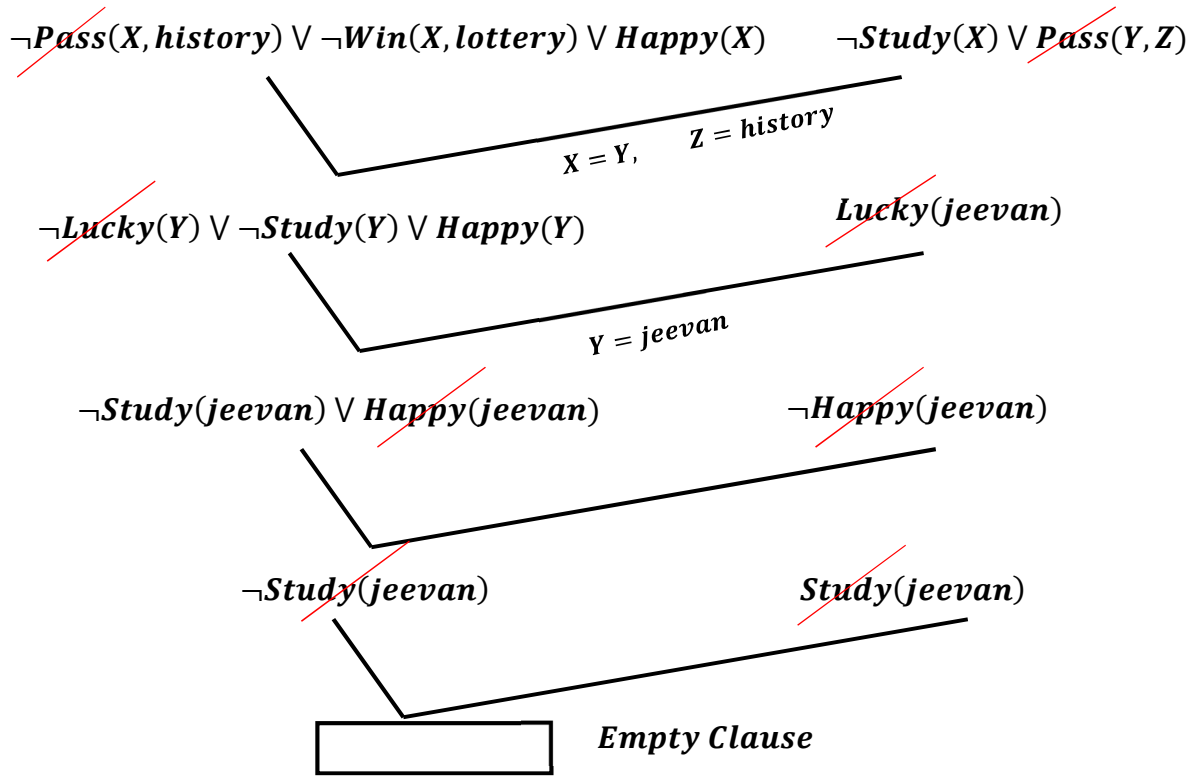
v) $Lucky(jeevan)$

vi) $\neg Lucky(U) \vee Win(U, lottery)$

Negation of Conclusion:

vii) $\neg Happy(jeevan)$

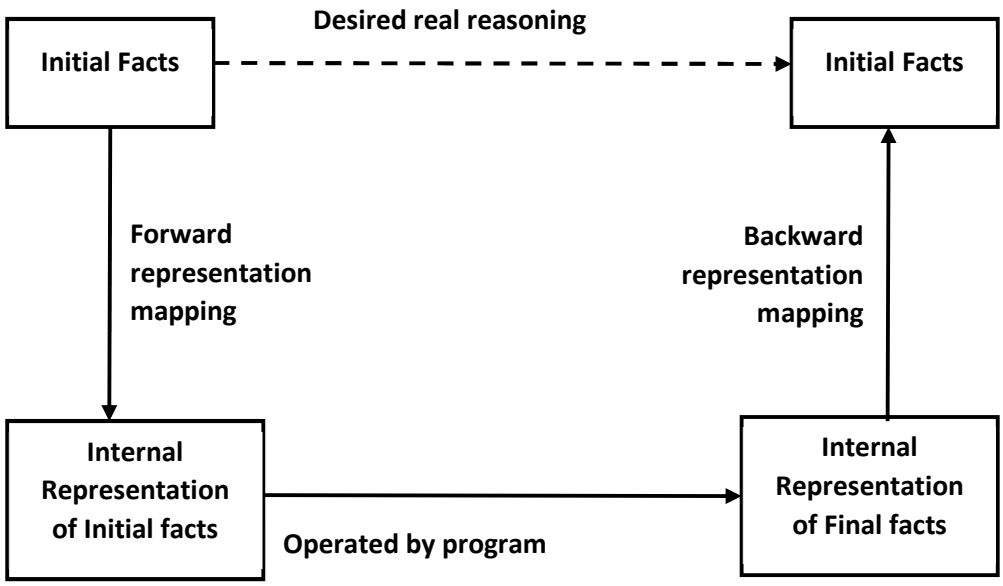
The Resolution procedure has been illustrated below:



Here, The **EMPTY** clause shows that the $\neg happy(jeevan)$ is false. This produce a contradiction or $happy(jeevan)$ will not produce contradiction with the known statement. So, we can say that **"jeevan is happy".** Thus, the given statement **"jeevan is happy"** is proved.

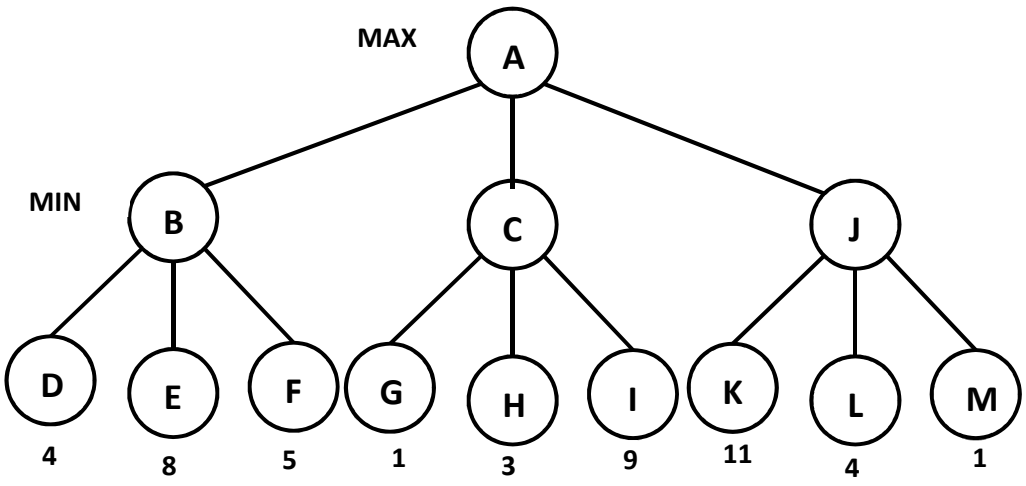
Q.5.b) Explain the forward and backward representation mapping with diagram. **[5]**

Ans: AI is the area of computer science focusing on creating machine that can engage on behaviors that humans consider intelligent. Knowledge can be managed by knowledge engineering which includes knowledge acquisition, knowledge representation and knowledge manipulation.

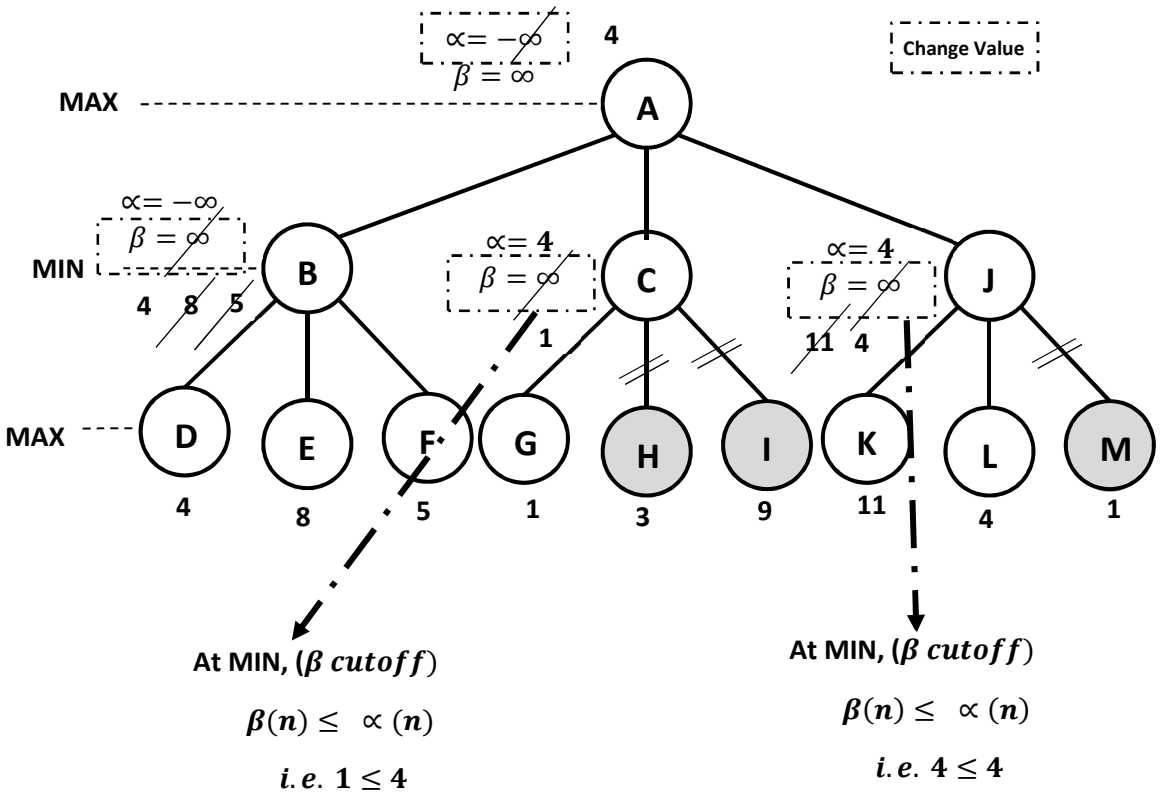


[Spring 2012]

Q.3.a) In the figure below, use minimax search with alpha-beta pruning to decide the next move (node) by MAX player from node A. The numbers indicate utility values. **[8]**



Soln:



Q.3.b) Solve the following crypto arithmetic problem. [7]

```
    S  E  N  D
    M  O  R  E
-------------------
M   O  N  E  Y
```

Soln:

```
    9  5  6  7
    1  0  8  5
-------------------
1   0  6  5  2
```

Q.4.a) Assume the following facts:

- **i)** Bhaskar is a physician.
- **ii)** All physician knows surgery.
- **iii)** All MBBS are physician

Prove that **"Bhaskar knows surgery"** using resolution.

Sol^{n}: Convert these clauses/facts into FOL then CNF

i) Bhaskar is a physician

FOL: $Physician(bhaskar)$

CNF: $Physician(bhaskar)$

ii) All physician knows surgery.

FOL: $\forall X: Physician(X) \rightarrow Surgery(X)$

CNF: $\neg Physician(X) \vee Surgery(X)$

iii) All MBBS are physician.

FOL: $\forall Y: MBBS(Y) \rightarrow Physician(Y)$

CNF: $\neg MBBS(Y) \vee Physician(Y)$

To prove that **"Bhaskar knows surgery"** we first go with the contradiction means it is **FALSE.**

Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"Bhaskar does not know surgery":** $\neg Surgery(bhaskar)$

We have given facts in CNF:

i) $Physician(bhaskar)$

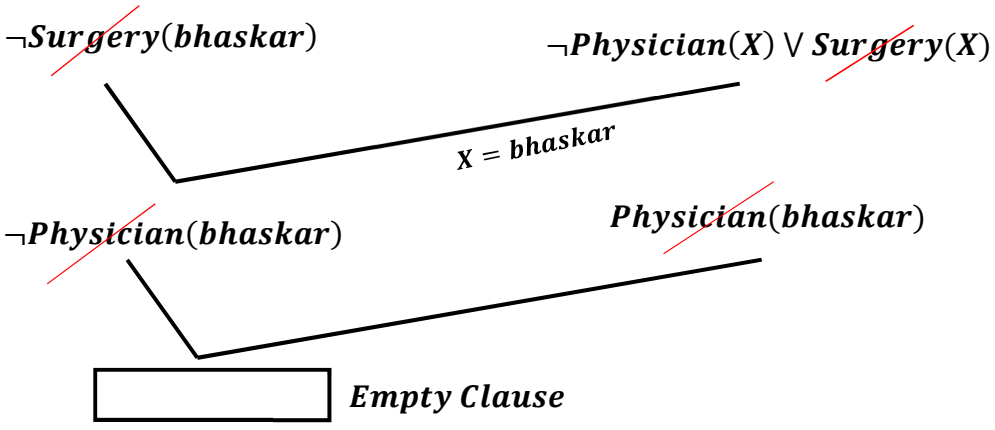
ii) $\neg Physician(X) \vee Surgery(X)$

iii) $\neg MBBS(Y) \vee physician(Y)$

Negation of Conclusion:

iv) $\neg Surgery(bhaskar)$

The Resolution procedure has been illustrated below:



Here, The **EMPTY** clause shows that the $\neg Surgery(bhaskar)$ is false. This produce a contradiction or $Surgery(bhaskar)$ will not produce contradiction with the known statement. So, we can say that **"Bhaskar knows surgery".**

Thus, the given statement **"Bhaskar knows surgery"** is proved.

What is Wff?

Wff stands for Well Formed Formula

Well Formed Formula (Wff) is a predicate holding any of the following −

1. All propositional constants and propositional variables are Wffs.
2. If **x** is a variable and **Y** is a Wff, **∀xY** and **∃xY** are also Wff.
3. Truth value and false values are Wffs.
4. Each atomic formula is a Wff.
5. All connectives connecting Wffs are Wffs.
   *Conjunction **[^]**, Disjunction **[v]**, implication **[→ ←→]**, Negation **[~].***

One way to check whether or not an expression is a wff is to try to state it in English. If you can translate it into a correct English sentence, then it is a wff.

More examples: To express the fact that Tom is taller than John, we can use the atomic formula Taller(tom, john), which is a wff. This wff can also be part of some compound statement such as Taller(tom, john) ^ ~ Taller(john, tom), which is also a wff.

If x is a variable representing people in the world, then Taller(X,tom), ∀X Taller(X,tom), ∃X Taller(X,tom), ∃Y ∀X Taller(X,Y) are all wffs among others.

However, Taller(X,john) and Taller(tom ^ mary, jim), for example, are NOT wffs.

Q.4.b) Convert the following into **Wffx**:

i) Everyone is liked by someone.

ii) It is 2012.

iii) Ravi's father is Rani's father.

iv) All cats have tail and whiskers.

Sol^{n}: Convert these clauses/facts into Wffx

i) Everyone is liked by someone.

Wff: $\forall X : \exists Y : \text{LikedBy}(X, Y)$

ii) It is 2012.

Wff: $now = 2012$

iii) Ravi's father is Rani's father.

Wff: $Father(father(ravi), rani))$

iv) All cats have tail and whiskers.

Wff: $\forall X : \text{Cat}(X) \rightarrow \text{Hastail}(X) \wedge Haswisker(X)$

[Fall 2013]

Q.3.a) Using a suitable example, illustrate steps of A* search. Why A* search is better than Best first search. **[7]**

Ans:

For your extra knowledge. Here, I've shown a different approach to solve the A* star search problem.

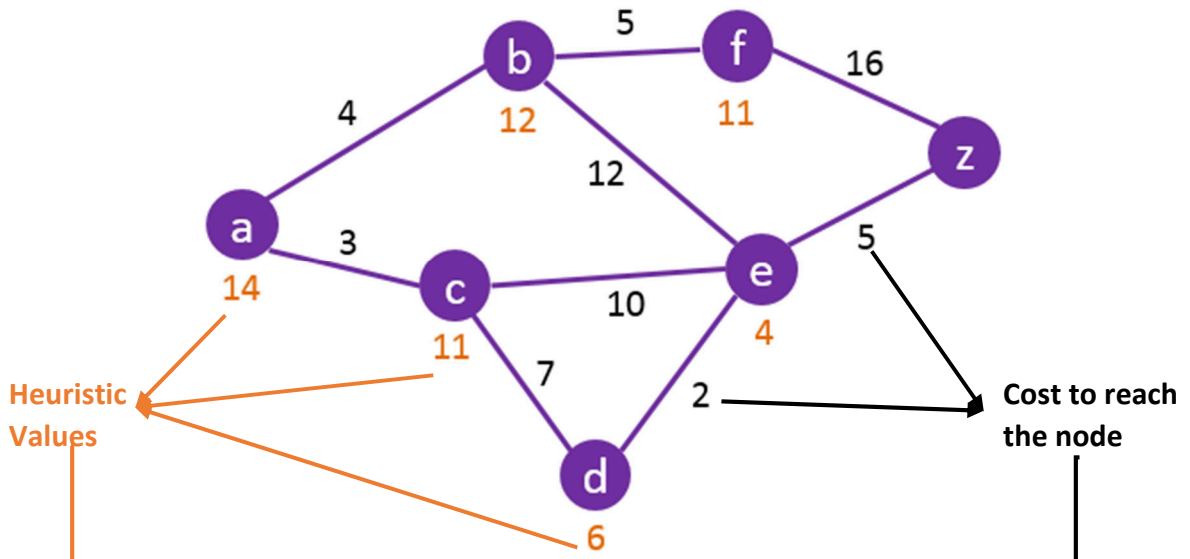
If you feel difficulty then skip it and see the easy method to solve that I have given you in Note or also you can get it down below there I have solve in easy method.

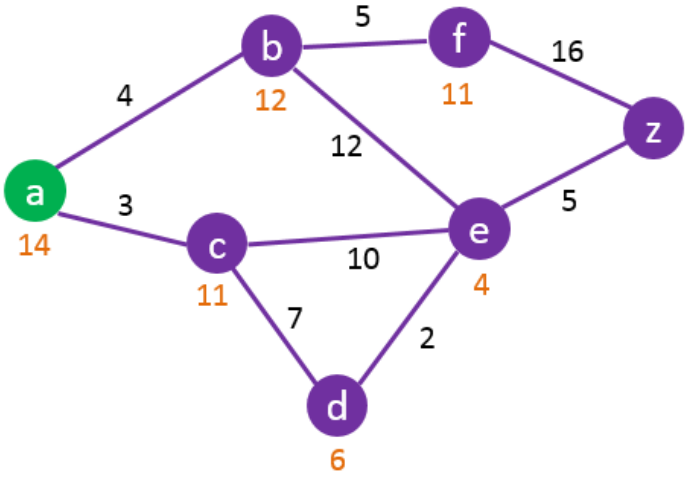
A* algorithm – pseudo code

- For each node n in the graph
  - n.f = Infinity, n.g = Infinity
- Create an empty list.
- start.g = 0, start.f = H(start) add start to list.
- While list not empty
  - Let current = node in the list with the smallest f value, remove current from list
  - If (current == goal node) report success
  - For each node, n that is adjacent to current

```
If (n.g > (current.g + cost of edge from n to current))
  n.g = current.g + cost of edge from n to current
  n.f = n.g + H(n)
  n.parent = current
  add n to list if it isn't there already
```

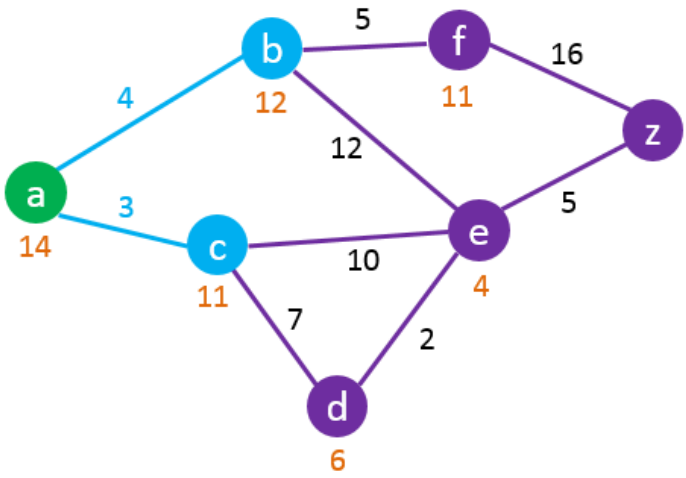
A suitable example, illustrate steps of A* search:





Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Current	0	14	14	
B		∞	12		
C		∞	11		
D		∞	6		
E		∞	4		
F		∞	11		
Z		∞	0		
* Total Distance = Shortest Distance from A + Heuristic Distance to Z					

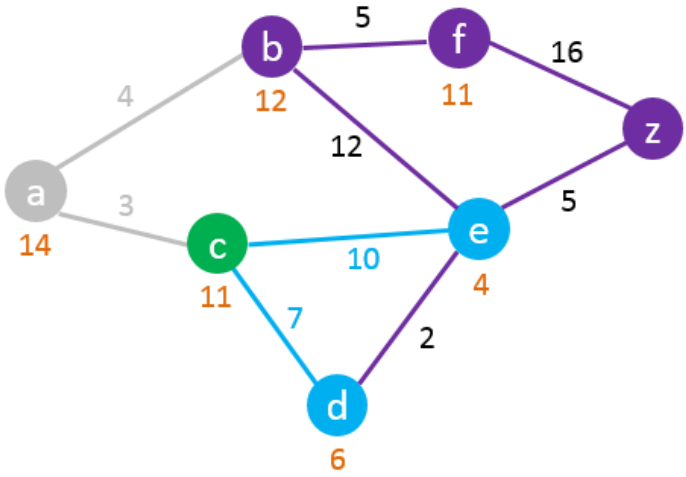
Start by setting the starting node (A) as the current node.



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Current	0	14	14	
B		∞ 4	12	16	A
C		∞ 3	11	14	A
D		∞	6		
E		∞	4		
F		∞	11		
Z		∞	0		
* Total Distance = Shortest Distance from A + Heuristic Distance to Z					

Check all the nodes connected to A and update their **"Shortest Distance from A"** and set their **"previous node"** to "A".

Update their total distance by adding the shortest distance from A and the heuristic distance to Z.

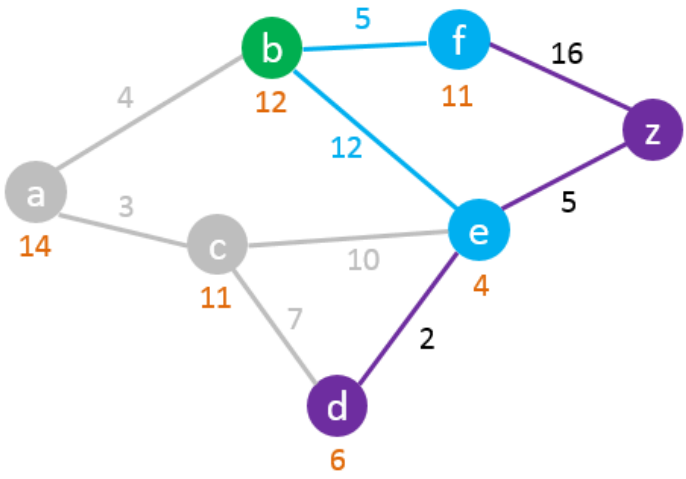


Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B		4	12	16	A
C	Current	3	11	14	A
D		∞ 3+7=10	6	16	C
E		∞ 3+10=13	4	17	C
F		∞	11		
Z		∞	0		
* Total Distance = Shortest Distance from A + Heuristic Distance to Z					

Set the current node (A) to **"visited"** and use the unvisited node with the smallest total distance as the **current node** (e.g. in this case: Node C).
Check all unvisited nodes connected to the current node and add the distance from A to C to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

C → D: 3 + 7 = 10 < ∞ – *Change Node D*
C → E: 3 + 10 = 13 < ∞ – *Change Node E*

The next current node (unvisited node with the shortest total distance) could be either node B or node D. Let's use node B.

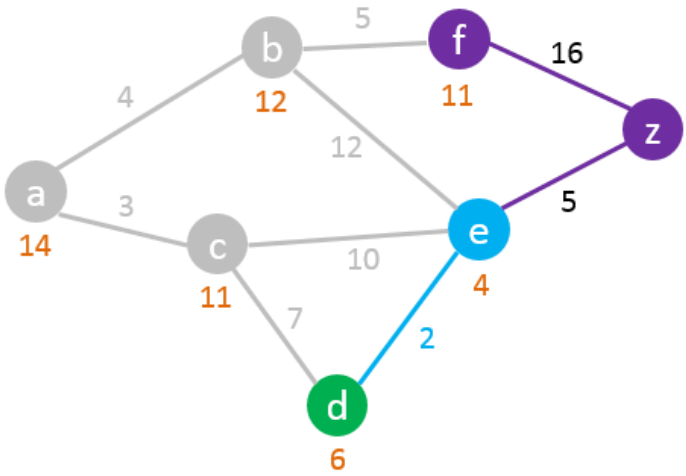


Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Current	4	12	16	A
C	Visited	3	11	14	A
D		10	6	16	C
E		13 4+12=~~16~~	4	17	C
F		~~∞~~ 4+5=9	11	20	B
Z		∞	0		
* Total Distance = Shortest Distance from A + Heuristic Distance to Z					

Check all unvisited nodes connected to the current node (B) and add the distance from A to B to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

B → E: 4 + 12 = 16 > 13 – *Do not change Node E*
B → F: 4 + 5 = 9 < ∞ – *Change Node F*

The next current node (unvisited node with the shortest total distance) is D.



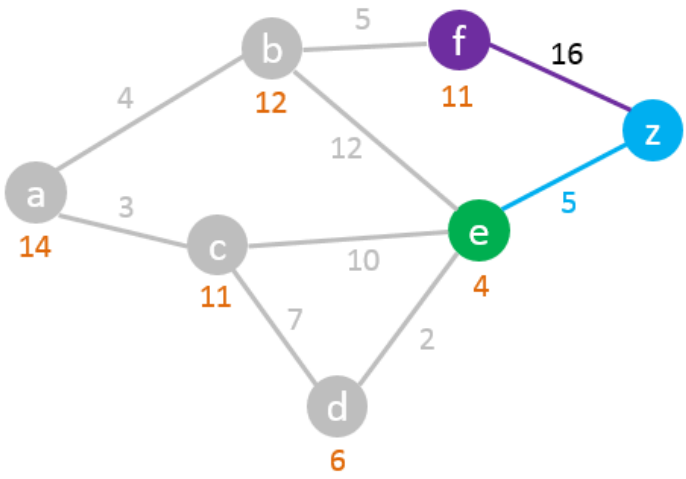
Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Current	10	6	16	C
E		~~13~~ 10+2=12	4	16	D
F		9	11	20	B
Z		∞	0		

* Total Distance = Shortest Distance from A + Heuristic Distance to Z

Check all unvisited nodes connected to the current node (D) and add the distance from A to D to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

D → E: 10 + 2 = 12 < 13 – *Change Node E*

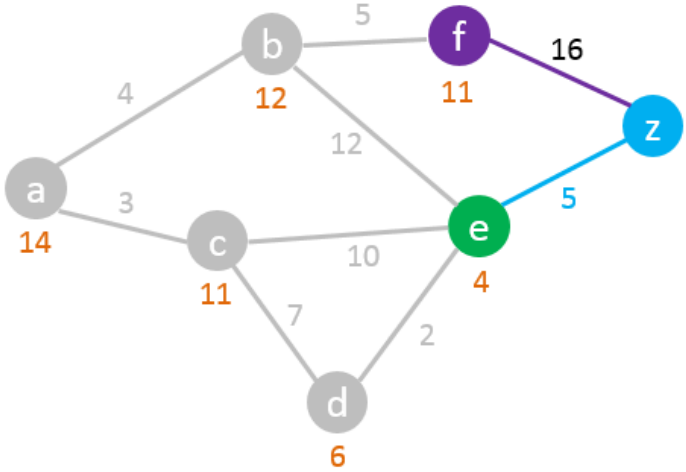
The next current node (unvisited node with the shortest total distance) is E.



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Visited	10	6	16	C
E	Current	12	4	16	D
F		9	11	20	B
Z		∞ 12+5=17	0	17	E
* Total Distance = Shortest Distance from A + Heuristic Distance to Z					

Check all unvisited nodes connected to the current node (E) and add the distance from A to E to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

E → Z: 12 + 5 = 17 < ∞ – *Change Node Z*



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B	Visited	4	12	16	A
C	Visited	3	11	14	A
D	Visited	10	6	16	C
E	Visited	12	4	16	D
F		9	11	20	B
Z	Current	17	0	17	E

* Total Distance = Shortest Distance from A + Heuristic Distance to Z

We found a path from A to Z, but is it the shortest one?

Check all unvisited nodes. In this example, there is only one unvisited node (F). However, its total distance (20) is already greater than the distance we have from A to Z (17) so there is no need to visit node F as it will not lead to a shorter path.

We found the shortest path from A to Z.
Read the path from Z to A using the previous node column:
Z > E > D > C > A
So the Shortest Path is:
A – C – D – E – Z with a length of 17

Best-first search algorithm visits next state based on heuristics function f(n) = h with lowest heuristic value (often called greedy). It doesn't consider cost of the path to that particular state. All it cares about is that which next state from the current state has lowest heuristics.

A* search algorithm visits next state based on heuristics f(n) = h + g where h component is same heuristics applied as in Best-first search but g component is path from the initial state to the particular state. Therefore, it doesn't choose next state only with lowest heuristics value but one that gives lowest value when considering its heuristics and cost of getting to that state.
So now clearly you can see best-first is greedy algorithm because it would choose state with lower heuristics but higher overall cost as it doesn't consider cost of getting to that state from initial state.

A* achieves better performance by using heuristics to guide its search. A* combines the advantages of Best-first Search and Uniform Cost Search: ensure to find the optimized path while increasing the algorithm efficiency using heuristics. A* function would be f(n) = g(n) + h(n) with h(n) being the estimated distance between any random vertex n and target vertex, g(n) being the actual distance between the start point and any vertex n. If g(n)=0, the A* turns to be Best-First Search. If h(n)=0, then A* turns to be Uniform-Cost Search.

Q.3.b) What are the properties possessed by a good knowledge representation system? **[8]**

Give P and Q prove that ((P→Q)→P)→P is a tautologous.

Ans: Knowledge representation is probably, the most important ingredient for developing an AI. A representation is a layer between information accessible from outside world and high-level thinking processes. Without knowledge representation it is impossible to identify what thinking processes are, mainly because representation itself is a substratum for a thought.

The following properties should be possessed by a knowledge representation system.

i) **Representational Adequacy:** It is the ability to represent the required knowledge.

ii) **Inferential Adequacy:** It is the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original.

iii) **Inferential Efficiency:** It is the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides.

iv) **Acquisitional Efficiency:** It is the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Prove that ((P→Q)→P)→P is a tautologous.

P	Q	P→Q	(P→Q)→P	((P→Q)→P)→P
T	T	T	T	T
T	F	F	T	T
F	T	T	F	T
F	F	T	F	T

Hence, ((P→Q)→P)→P is all **TRUE** so, it is proved that this expression is a tautologous.

Q.4.a) Convert A←→B←→C into Conjunctive Normal Form. **[7]**

Solⁿ:

1. **Eliminate implications and bi-conditionals using formulas:**

$A \leftrightarrow B \leftrightarrow C$

$\equiv (A \rightarrow B) \wedge (B \rightarrow A) \leftrightarrow C$

$\equiv \big(((A \rightarrow B) \wedge (B \rightarrow A)) \rightarrow C\big) \wedge \big(C \rightarrow ((A \rightarrow B) \wedge (B \rightarrow A))\big)$

$\equiv \big(\neg((A \rightarrow B) \wedge (B \rightarrow A))\ V\ C\big) \wedge \big(\neg C \vee ((A \rightarrow B) \wedge (B \rightarrow A))\big)$

$\equiv \big(\neg\big((\neg A \vee B) \wedge (\neg B \vee A)\big) \vee C\big) \wedge \big(\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A)\big)$

2. **Apply De-Morgan's Law and reduce NOT symbols so as to bring negations before the atoms. Use:**

$$(\neg((\neg A \vee B) \wedge (\neg B \vee A)) \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv ((A \wedge \neg B) \vee (B \wedge \neg A) \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv ((A \wedge \neg B) \vee (B \vee C) \wedge (\neg A \vee C)) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (((A \wedge \neg B) \vee B \vee C) \wedge ((A \wedge \neg B) \vee \neg A \vee C)) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (((A \vee B) \wedge (\neg B \vee B)) \vee C) \wedge ((A \wedge \neg B) \vee \neg A \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (((A \vee B) \wedge \top) \vee C) \wedge ((A \wedge \neg B) \vee \neg A \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (A \vee B \vee C) \wedge ((A \wedge \neg B) \vee \neg A \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (A \vee B \vee C) \wedge (((A \vee \neg A) \wedge (\neg B \vee \neg A)) \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (A \vee B \vee C) \wedge (((\top \wedge (\neg B \vee \neg A)) \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (A \vee B \vee C) \wedge (\neg B \vee \neg A \vee C) \wedge (\neg C \vee (\neg A \vee B) \wedge (\neg B \vee A))$$

$$\equiv (A \vee B \vee C) \wedge (\neg B \vee \neg A \vee C) \wedge (\neg C \vee \neg A \vee B) \wedge (\neg C \vee \neg B \vee A)(CNF)$$

Note: $\boldsymbol{A} \vee \neg \boldsymbol{A}$ is a tautology in classical (i.e., Aristotelian) logic because you can prove that using the deduction rules of the classical proposition calculus no matter what the truth value of A is, the truth value of $\boldsymbol{A} \vee \neg \boldsymbol{A}$ is always true. That is the meaning of tautology.

Q.4.b) Represent the following sentences in first order logic: **[8]**

i) A person with a dust allergy sneeze.
ii) Every flower likes water.
iii) You can fool all of the people some of the time.
iv) No cake lover throws a cake.

Soln:

i) A person with a dust allergy sneeze.

FOL: $\forall X, Y, Z : \text{Person}(X) \wedge Sneeze(Y) \wedge Allergy(Y) \rightarrow \text{Dust}(X, Y, Z)$

ii) Every flower likes water.

FOL: $\forall X : \text{Flower}(X) \rightarrow \text{Likes}(X, \text{water})$

iii) You can fool all of the people some of the time.

FOL: $\exists Y : \text{Time}(Y) \wedge \forall X : Person(X) \rightarrow Canfool(X, Y)$

iv) No cake lover throws a cake.

FOL: $\forall X, Y : \text{CakeLover}(X) \wedge Cake(Y) \rightarrow \neg Throws(X, Y)$

~~$\forall X: (\text{Cake}(X) \wedge Lover(X)) \rightarrow \neg Throws(X)$~~

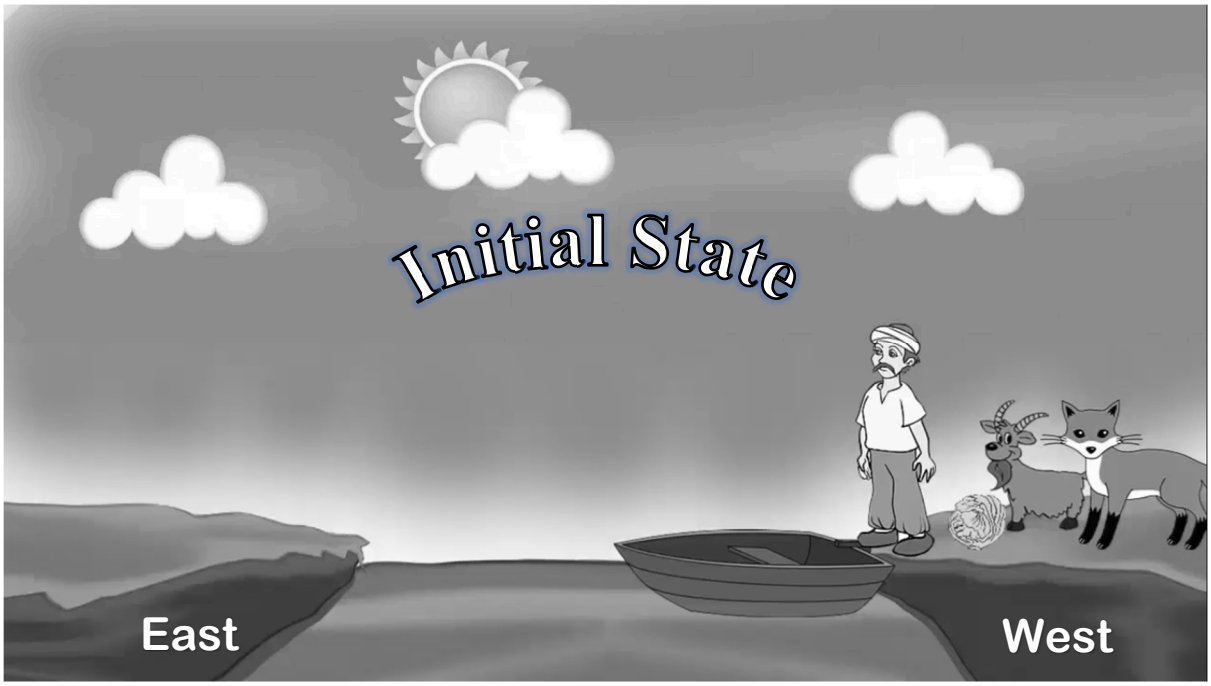
v) You can fool some of the people all of the time.

FOL: $\exists X : \forall Y : \text{Person}(X) \wedge Time(Y) \rightarrow Canfool(X, Y)$

[Spring 2013]

Q.1.b) A farmer has a goat, a wolf & a cabbage on the west side of a river. He wants to get all of his animals & his cabbage across the river onto the east side. The farmer has a row boat but he only has enough room for himself & one other thing. The wolf will eat the goat if they are left together alone. The goat will eat the cabbage if they are left together alone. How can the farmer get everything on the east side? **[8]**

i) Formulate this puzzle as search.
ii) Find solution for this problem.



Sol^{n}:

Solution 1:

i. Farmer takes Goat across (leaving Wolf and Cabbage behind)
ii. Farmer returns alone
iii. Farmer takes Wolf across
iv. Farmer returns with Goat

* We now have the Farmer, the Cabbage and the Goat on one side and the Wolf on the other side

v. Farmer takes Cabbage across
vi. Farmer returns alone
vii. Farmer takes Goat across ***DONE!***

Solution 2:

i. Farmer takes Goat across (leaving Wolf and Cabbage behind)
ii. Farmer returns alone
iii. Farmer takes Cabbage across
iv. Farmer returns with Goat

* We now have the Farmer, the Wolf and the Goat on one side and the Cabbage on the other side

v. Farmer takes Wolf across
vi. Farmer returns alone
vii. Farmer takes Goat across ***DONE!***

Goal: Move everything (the Farmer, the Cabbage, the Goat, the Wolf) on the East side.

Constraint: The farmer has a row boat but he only has enough room for himself & one other thing. The wolf will eat the goat if they are left together alone. The goat will eat the cabbage if they are left together alone.

State: Configuration of the Farmer, the Cabbage, the Goat, the Wolf and boat on each side of river.

Operators: Move boat containing some set of occupants across the river (in either direction) to the other side.

Let,

The Farmer = F,

The Goat = G,

The Cabbage = C,

The Wolf = W

Rules for solving Goat, Wolf and Cabbage Problem:

#Rules	Action	Meaning
1.	(0, F)	Farmer return alone from East side to West side.
2.	(F, 0)	Farmer return alone from West side to East side.
3.	(F, G)	Farmer takes Goat from East side to West side.
4.	(F, G)	Farmer takes Goat from West side to East side.
5.	(F, W)	Farmer takes Wolf from East side to West side.
6.	(F, W)	Farmer takes Wolf from West side to East side.
7.	(F, C)	Farmer takes Cabbage from East side to West side.
8.	(F, C)	Farmer takes Cabbage from West side to East side.

Table: Production Rules for the Goat, Wolf and Cabbage Problem

Applying the rules: Formulation I

Rule Applied	Things in the East side	Things in the West side	Boat Position
Initial State	***0***	***F, G, C, W***	***West side***
4	F, G	C, W	East side
1	G	F, C, W	West side
6	G, F, W	C,	East side
3	W	F, G, C	West side
8	W, F, C	G	East side
1	W, C	G, F	West side
4	W, C, G, F	0	East side
Goal State	***W, C, G, F***	***0***	***East side***

Applying the rules: Formulation II

Rule Applied	Things in the East side	Things in the West side	Boat Position
Initial State	***0***	***F, G, C, W***	***West side***
4	F, G	C, W	East side
1	G	F, C, W	West side
8	G, F, C	W	East side
3	C	F, G, W	West side
6	C, F, W	G	East side
1	C, W	G, F	West side
4	C, W, G, F	0	East side
Goal State	***C, W, G, F***	***0***	***East side***

Q.2.a) Solve the following crypto arithmetic problem. **[7]**

Sol^{n}:

```
    F   O   R   T   Y          2   9   7   8   6
        T   E   N              8   5   0
        T   E   N              8   5   0
-------------------------------------------------
    S   I   X   T   Y          3   1   4   8   6
```

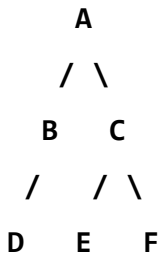
Q.2.b) "Breadth first search is an implementation of queue where as Depth first search is an implementation of stack". Verify this statement with suitable example. **[8]**

Ans:

Breadth First Search:

BFS stands for Breadth First Search is a vertex-based technique for finding a shortest path in graph. It uses a Queue data structure which follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

Here is the short and simple example of BFS:



Output is:

A, B, C, D, E, F

BFS Algorithm:

Step 1: Input the vertices and edges of the graph G = (V, E).

Step 2: Input the source vertex and assign it to the variable S.

Step 3: Add or push the source vertex to the **queue**.

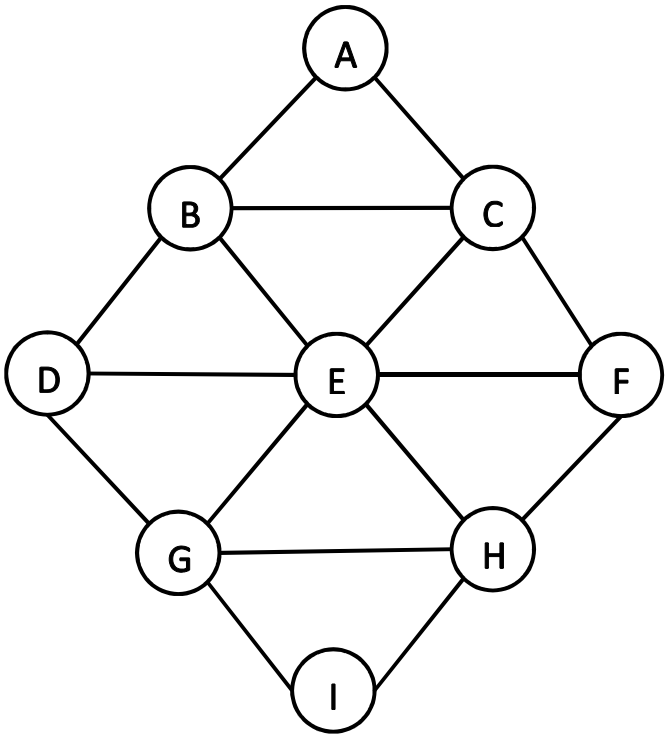
Step 4: Repeat the steps 5 and 6 until the queue is empty ***(i.e. front > rear)*.**

Step 5: Pop the front element of the queue and display it as visited.

Step 6: Push the vertices, which is neighbor to just popped element, if it is not in the queue and displayed (i.e. not visited).

Step 7: Stop

Example:



Suppose the source vertex is **A**

Step 1: Initially push A to the queue.

0	1	2	3	4	5	6	7	8	9
A									
Front Rear									

Step 2: Remove the front element A from the queue and display it. Then push all the neighboring vertices of A to the queue, if it is not in the queue (i.e. not visited).

A → B, C

0	1	2	3	4	5	6	7	8	9
A	B	C							
	Front	Rear							

Output: A

Step 3: Remove the front element B from the queue and display it. Then push all the neighboring vertices of B to the queue, if it is not in the queue (i.e. not visited).

B → ~~A, C,~~ D, E

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E					
		Front		Rear					

Output: A, B

Step 4: Remove the front element C from the queue and display it. Then push all the neighboring vertices of C to the queue, if it is not in the queue (i.e. not visited).

C → ~~A, B,~~ ~~E,~~ F

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F				
			Front		Rear				

Output: A, B, C

Step 5: Remove the front element D from the queue and display it. Then push all the neighboring vertices of D to the queue, if it is not in the queue (i.e. not visited).

D → ~~B,~~ ~~E,~~ G

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G			
				Front		Rear			

Output: A, B, C, D

Step 6: Remove the front element E from the queue and display it. Then push all the neighboring vertices of E to the queue, if it is not in the queue (i.e. not visited).

E → ~~B, C, D,~~ ~~F, G,~~ H

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H		
					Front		Rear		

Output: A, B, C, D, E

Step 7: Remove the front element F from the queue and display it. Then push all the neighboring vertices of F to the queue, if it is not in the queue (i.e. not visited).

F → ~~C, E,~~ ~~H~~

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H		
						Front	Rear		

Output: A, B, C, D, E, F

Step 8: Remove the front element G from the queue and display it. Then push all the neighboring vertices of G to the queue, if it is not in the queue (i.e. not visited).

G → ~~D, E,~~ ~~H,~~ I

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	
							Front	Rear	

Output: A, B, C, D, E, F, G

Step 9: Remove the front element H from the queue and display it. Then push all the neighboring vertices of H to the queue, if it is not in the queue (i.e. not visited).

H → ~~E, F, G,~~ ~~I~~

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	
								Front Rear	

Output: A, B, C, D, E, F, G, H

Step 10: Remove the front element I from the queue and display it. Then push all the neighboring vertices of I to the queue, if it is not in the queue (i.e. not visited).

I → ~~G, H~~

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	
								Rear	Front

Output: A, B, C, D, E, F, G, H, I

Step 11: Here, Front = 9 and Rear = 8

i.e. Front > Rear and all vertices are visited. So, we stop here.

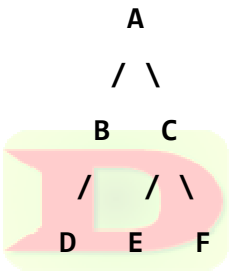
The Breadth first traversal of given graph is: **A, B, C, D, E, F, G, H, I**

----------X----------X----------

Depth First Search:

DFS stands for Depth First Search is an edge-based technique. It uses the Stack data structure, performs two stages, first visited vertices are pushed into stack and second if there is no vertices then visited vertices are popped.

Here is the short and simple example of DFS:



Output is:

A, B, D, C, E, F

DFS Algorithm:

Step 1: Input the vertices and edges of the graph G = (V, E).

Step 2: Input the source vertex and assign it to the variable S.

Step 3: Add or push the source vertex to the ***stack***.

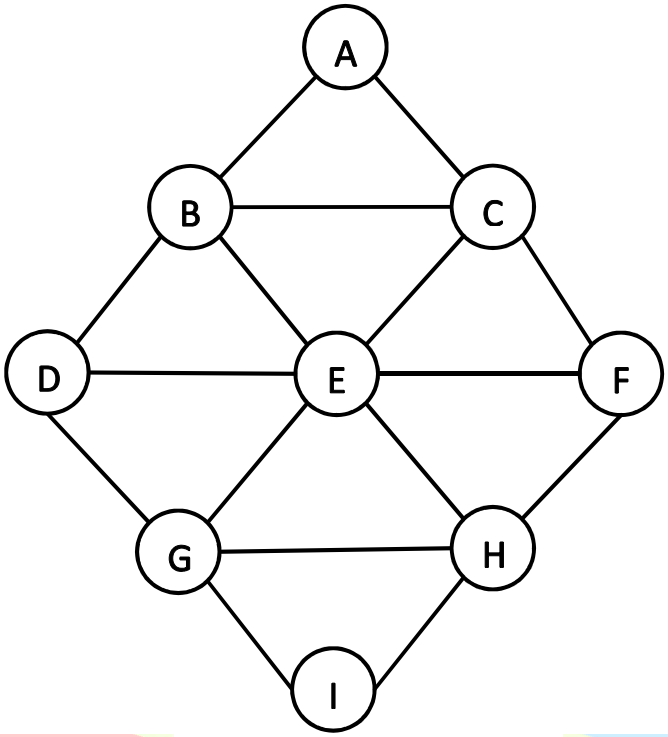
Step 4: Repeat the steps 5 and 6 until the stack is empty and all node visited ***(i.e. Top= -1)***.

Step 5: Pop the top element of the stack and display it as visited.

Step 6: Push the vertices, which is neighbor to just popped element, if it is not in the stack and displayed (i.e. not visited).

Step 7: Stop

Example:



Suppose the source vertex is **A,** **Top = -1**

Step 1: Initially push A to the stack.

0	A	Top

Step 2: Pop top of the stack A and display it. Then, Push all non-visited neighboring elements of A to the stack.

A → B, C

1	C	Top
0	B	

Output: A

Step 3: Pop top of the stack C and display it. Then, Push all non-visited neighboring elements of C to the stack.

C → ~~A,~~ ~~B~~, E, F

2	F	Top
1	E	
0	B	

Output: A, C

Step 4: Pop top of the stack F and display it. Then, Push all non-visited neighboring elements of F to the stack.

F → ~~C~~, ~~E~~, H

2	H	Top
1	E	
0	B	

Output: A, C, F

Step 5: Pop top of the stack H and display it. Then, Push all non-visited neighboring elements of H to the stack.

H → ~~E~~, ~~F~~, G, I

3	I	Top
2	G	
1	E	
0	B	

Output: A, C, F, H

Step 6: Pop top of the stack I and display it. Then, Push all non-visited neighboring elements of I to the stack.

I → ~~G~~, ~~H~~

2	G	Top
1	E	
0	B	

Output: A, C, F, H, I

Step 7: Pop top of the stack G and display it. Then, Push all non-visited neighboring elements of G to the stack.

G → D, ~~E~~, ~~H, I~~

2	D	Top
1	E	
0	B	

Output: A, C, F, H, I, G

Step 8: Pop top of the stack D and display it. Then, Push all non-visited neighboring elements of D to the stack.

D → ~~B~~, ~~E~~, ~~G~~

1	E	Top
0	B	

Output: A, C, F, H, I, G, D

Step 9: Pop top of the stack E and display it. Then, Push all non-visited neighboring elements of E to the stack.

E → ~~B~~, ~~C~~, ~~D~~, ~~F~~, ~~G~~, ~~H~~

0	B	Top

Output: A, C, F, H, I, G, D, E

Step 10: Pop top of the stack B and display it. Then, Push all non-visited neighboring elements of B to the stack.

B → ~~A~~, ~~C~~, ~~D~~, ~~E~~

-1	-	Top

Output: A, C, F, H, I, G, D, E, B

Step 11: Here, Top = -1 i.e. Stack is empty and all the vertices of the graph are visited. So, we stop here.

Depth first search traversal is: **A, C, F, H, I, G, D, E, B**

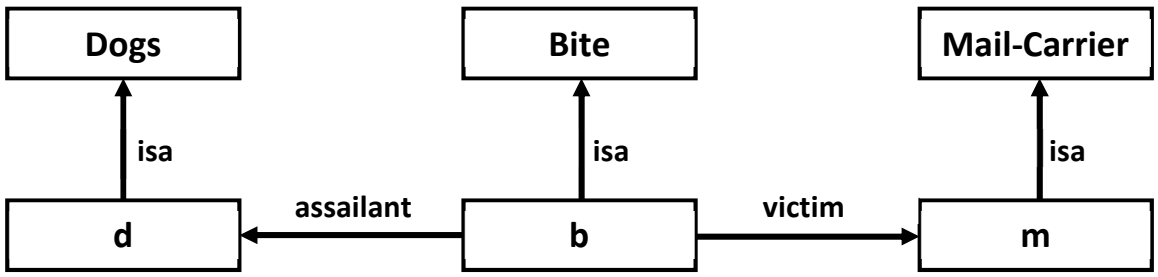
Q.5.a) What is semantic net? For the statement, **"The dog bit the mail carrier."**. Now represent this statement (Knowledge) using partitioned semantic nets.

Ans: A semantic network is a graphic notation for representing knowledge in patterns of interconnected nodes. Semantic networks became popular in artificial intelligence and natural language processing only because it represents knowledge or supports reasoning. These act as another alternative for predicate logic in a form of knowledge representation.

The structural idea is that knowledge can be stored in the form of graphs, with nodes representing objects in the world, and arcs representing relationships between those objects.

Semantic nets consist of nodes, links and link labels.

"The dog bit the mail carrier."



The nodes *Dogs, Bite, and Mail-Carrier* represent the classes of dogs, bitings, and mail carriers, respectively, while the nodes d, b and m represent a particular dog, a particular biting, and a particular mail carrier. This fact can easily be represented by a single net with no partitioning.

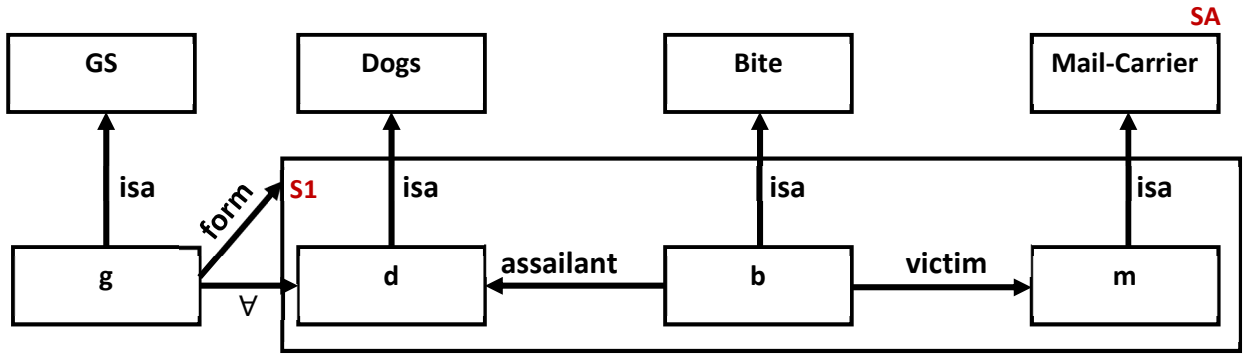
But now suppose that we want to represent the fact

"Every dog has bitten a mail carrier."

Or, in logic:

$$\forall \boldsymbol{X} : \boldsymbol{Dogs}(\boldsymbol{X}) \rightarrow \exists \boldsymbol{Y} : \boldsymbol{Mail\_Carrier}(\boldsymbol{Y}) \wedge \boldsymbol{Bite} \wedge (\boldsymbol{X}, \boldsymbol{Y})$$

To represent this fact, it is necessary to encode the scope f the universally quantified variable X. This can be done using partitioning as shown in below:



GS→ General Statement.

SA→ Space corresponding to whole event.

S1→ Space corresponding to own event.

More on Semantic Network:

Semantic Networks are of two types:

i) **Simple Semantic Network** and
ii) **Partitioned Semantic Network**

Simple Semantic Network: In this net we have:

- Is a relation → classes
- Has a relation → properties
- Instance relation → class and instance

Example:

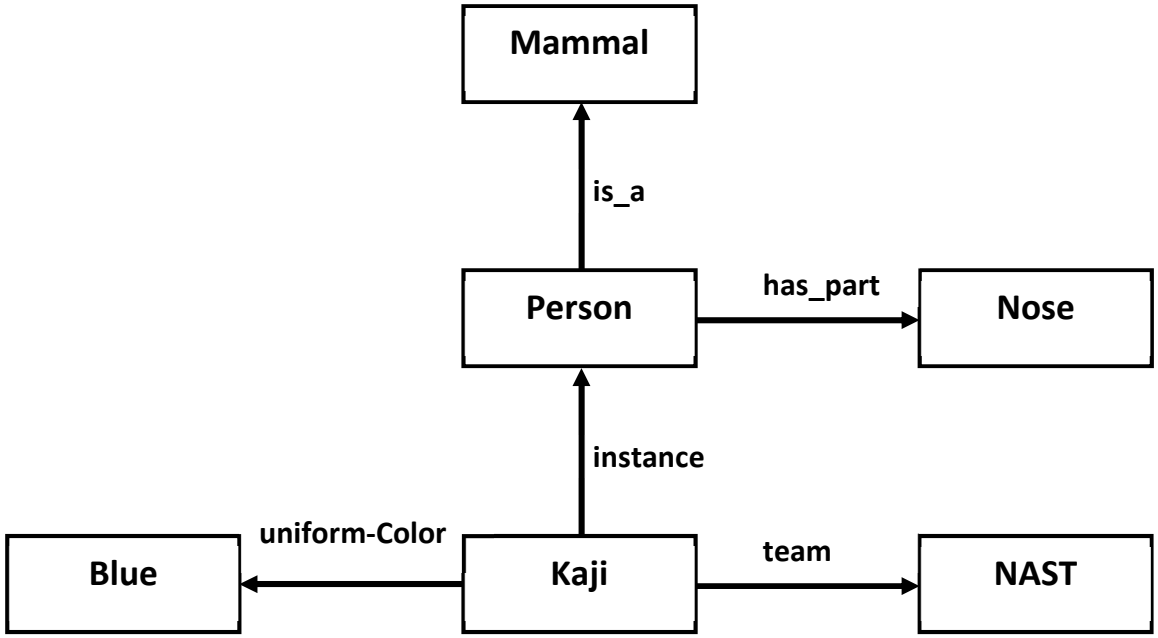
is_a(Person, Mammal)

instance(Kaji, Person)

team(Kaji, NAST)

uniform_color(Kaji, Blue)

has_part(Kaji, Nose)



[Spring 2014]

Q.4.b) Represent the following sentences in first order logic: **[8]**

i) Nabin is intelligent than all other student in his class.
ii) Neiher Sabin nor Nabin is unhappy.
iii) Some cats are domestic pets.
iv) Some turtles are faster than rabbits.

Sol^{n}:

i) Nabin is intelligent than all other student in his class.

FOL: $\forall X : \text{Student}(X) \rightarrow \text{Intelligent}(\text{nabin}, X)$

ii) Neiher Sabin nor Nabin is unhappy.

FOL: $\neg(happy(Sabin) \vee happy(Nabin))$

iii) Some cats are domestic pets.

FOL: $\exists X : \text{Cats}(X) \wedge \text{DomesticPets}(X)$

iv) Some turtles are faster than rabbits.

FOL: $\exists X : \text{Turtle}(X) \wedge \forall Y : \text{Rabbit}(Y) \rightarrow \text{faster}(X, Y)$

[Fall 2015]

Q.1.b) Define State Space in problem solving. Using constraint satisfaction problem, solve the following crypto-arithmetic problem: **CROSS+ROADS=DANGER** **[8]**

Ans: State space search is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or states of an instance are considered, with the intention of finding a goal state with a desired property.

Problems are often modelled as a state space, a set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

State space search often differs from traditional computer science search methods because the state space is implicit: the typical state space graph is much too large to generate and store in memory. Instead, nodes are generated as they are explored, and typically discarded thereafter. A solution to a combinatorial search instance may consist of the goal state itself, or of a path from some initial state to the goal state.

	C	R	O	S	S
	R	O	A	D	S
D	A	N	G	E	R

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

[Fall 2015]

Q.3.b) Define Semantic Network. Draw semantic network of following clauses: **[7]**

Subset_of(Human, Mammal), Subset_of(Male, Human), Subset_of(Female, Human), Has_Mother(Human, Female), Member_of(Mary, Female), Member_of(John, Male), Husband_of(John, Mary).

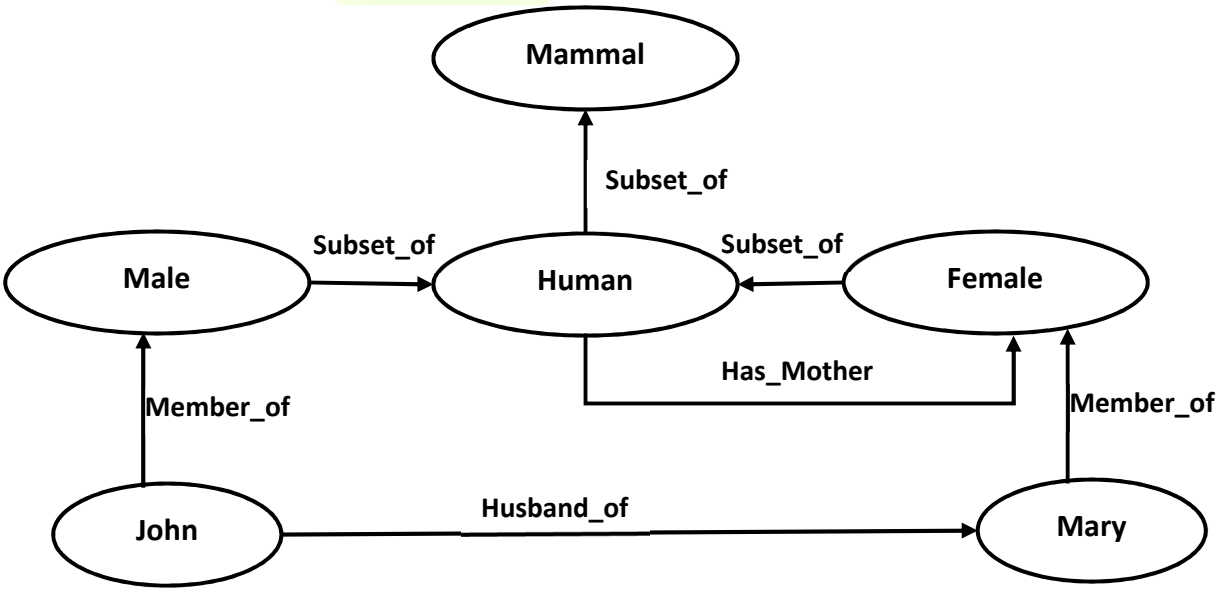
Ans: A semantic network, is a knowledge base that represents semantic relations between concepts in a network. This is often used as a form of knowledge representation. It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between concepts, mapping or connecting semantic fields.

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

Advantages of Semantic network:

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

Following is the semantic network of above clauses:



[Fall 2015]

Q.4.a) Assume the following facts: **[8]**

i) Steve only like easy courses.
ii) Science courses are hard.
iii) All the course in the basket weaving department are easy.
iv) BK301 is a basket weaving course.

Use resolution to answer the question, **"What course would Steve like?"**

Sol^{n}:

i) Steve likes easy courses.

FOL: $\forall X: Easy(X) \rightarrow Likes(steve, X)$

CNF: $\neg Easy(X) \vee Likes(steve, X)$

ii) Science courses are hard.

FOL: $Hard\ (science) \equiv \neg Easy(science)$

CNF: $\neg Easy(science)$

iii) All the courses in the basket weaving departments are easy.

FOL: $\forall Y: BasketWeaving(Y) \rightarrow Easy(Y)$

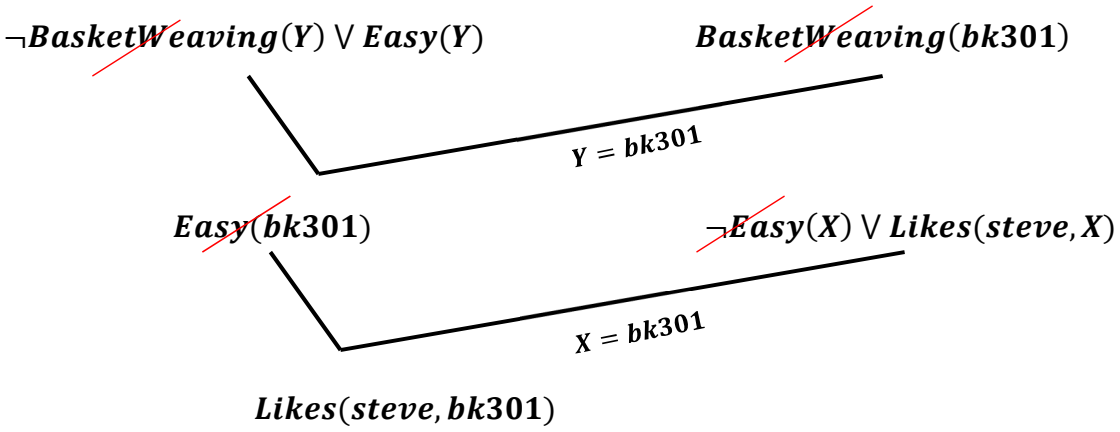
CNF: $\neg BasketWeaving(Y) \vee Easy(Y)$

iv) BK301 is a basket weaving course.

FOL: $BasketWeaving(bk301)$

CNF: $BasketWeaving(bk301)$

The Resolution procedure has been illustrated below:



At last we got $\boldsymbol{Likes(steve, bk301)}$

Thus, **Steve likes Bk301** is the final result.

[Fall 2015]

Q.4.a) Using Truth Table, Prove that P←→Q is equivalent to (P→Q)^(Q→P). **[7]**

Solⁿ:

P	Q	P←→Q	(P → Q)	(Q → P)	(P → Q)^(Q → P)
T	T	T	T	T	T
T	F	F	F	T	F
F	T	F	T	F	F
T	T	T	T	T	T

≡

Q.6.b) [*Solved in Unit 6 Note*]

[Fall 2016]

Q.2.a) Solve the following Crypto arithmetic problems. **[8]**

```
    T   W   O
    T   W   O
-------------------
F   O   U   R
```

Solⁿ:

```
    9   3   8
    9   3   8
-------------------
1   8   7   6
```

Here are other possibilities:

938+938=1876

928+928=1856

867+867=1734

846+846=1692

836+836=1672

765+765=1530

734+734=1468

NIM-Game

Introduction:

NIM is a mathematical game of strategy in which two players take turns removing (or "nimming") objects from distinct heaps or piles. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap or pile. Depending on the version being played, the goal of the game is either to avoid taking the last object, or to take the last object.

Variants of NIM have been played since ancient times. The game is said to have originated in China—it closely resembles the Chinese game of ***jiǎn-shízi, or "picking stones"*** but the origin is uncertain; the earliest European references to NIM are from the beginning of the 16th century. Its current name (NIM) was coined by **Charles L. Bouton** of Harvard University, who also developed the complete theory of the game in 1901, but the origins of the name were never fully explained.

NIM is typically played as a misère game, in which the player to take the last object loses.

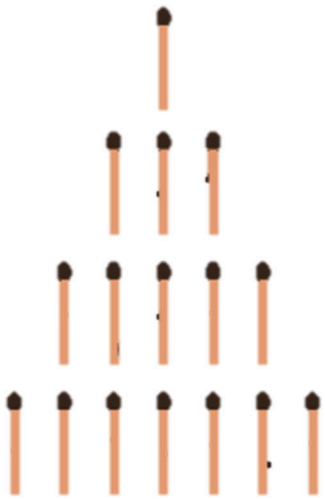
Category: Mathematical game of strategy / Combinatorial game

Name: NIM-game (or Marienbad-game, aka tactix)

Material: Matchsticks or any other counters.

Aim of the game: The player who takes the last counter loses.

Editor's notice: The second player can always win, that makes NIM a distinctly 'impartial game'.



The traditional Nim-game (aka Marienbad-game) consists of four rows of 1, 3, 5 and 7 matchsticks (or any other objects). Two players take any number of matchsticks from one row alternately. The one, who takes the last matchstick loses.

Matches set up in rows for a game of Nim. Players take turns to choose a row and remove any number of matches from it.

[Fall 2016]

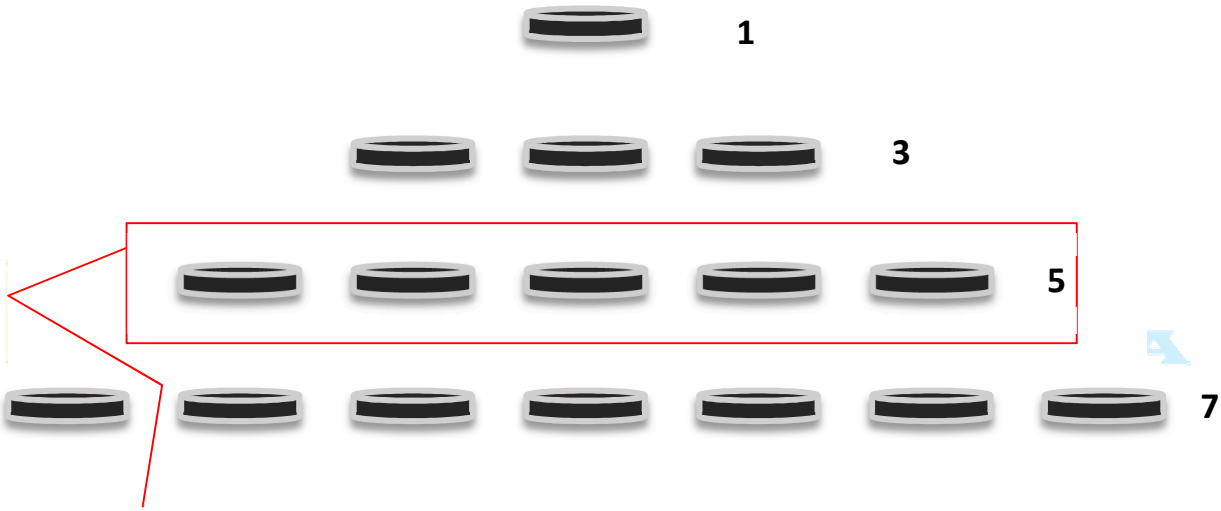
Q.3.a) The game of NIM is played as follows: **[8]**

Two players alternatively remove one, two or three coin from a stack initially containing five coin. The player who picks up the last coin loses.

i) Draw the full game tree.
ii) Show that the player who has the second move can always win.
iii) Execute $\alpha - \beta$ procedure on the game tree. How many terminal nodes are examined?

Soln:

First set up the coins for the general NIM game.



As per the question given,

Two players alternatively remove one, two or three coin from a stack initially containing five coin. The player who picks up the last coin loses.

i) Draw the full game tree.

- We have **n** stacks with **k** coins each.
- Players alternatively one or more coins from one stack at a time.
- The player who removes the last token loses.
- A **square** indicates a move by the **first player** and a **circle** indicates a move by the **second player**.
- If the **terminal vertices are a circle**, we label the vertex with **0** as the first player loses.
- If the **terminal vertices are a square**, we label the vertex with **1** as the first player wins.
- The label of each internal vertex is alternately the maximum and then the minimum of the labels of the children (maximum if the internal vertex is a square and minimum if the internal vertex is a circle)
- In this case, we have **1 stack** with **5 coins.**
- On each turn, the player selects **1, 2 or 3 coins**.

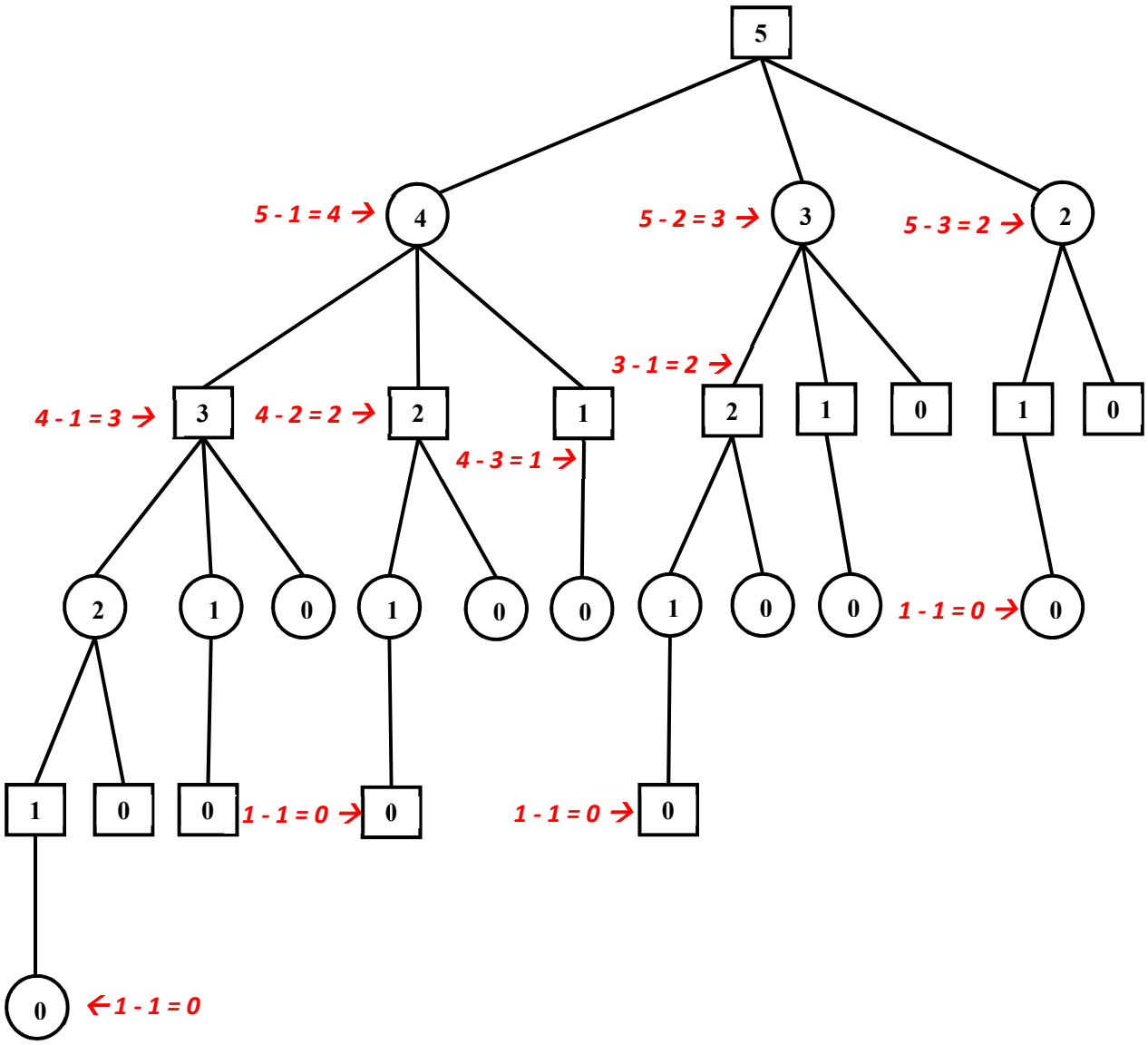


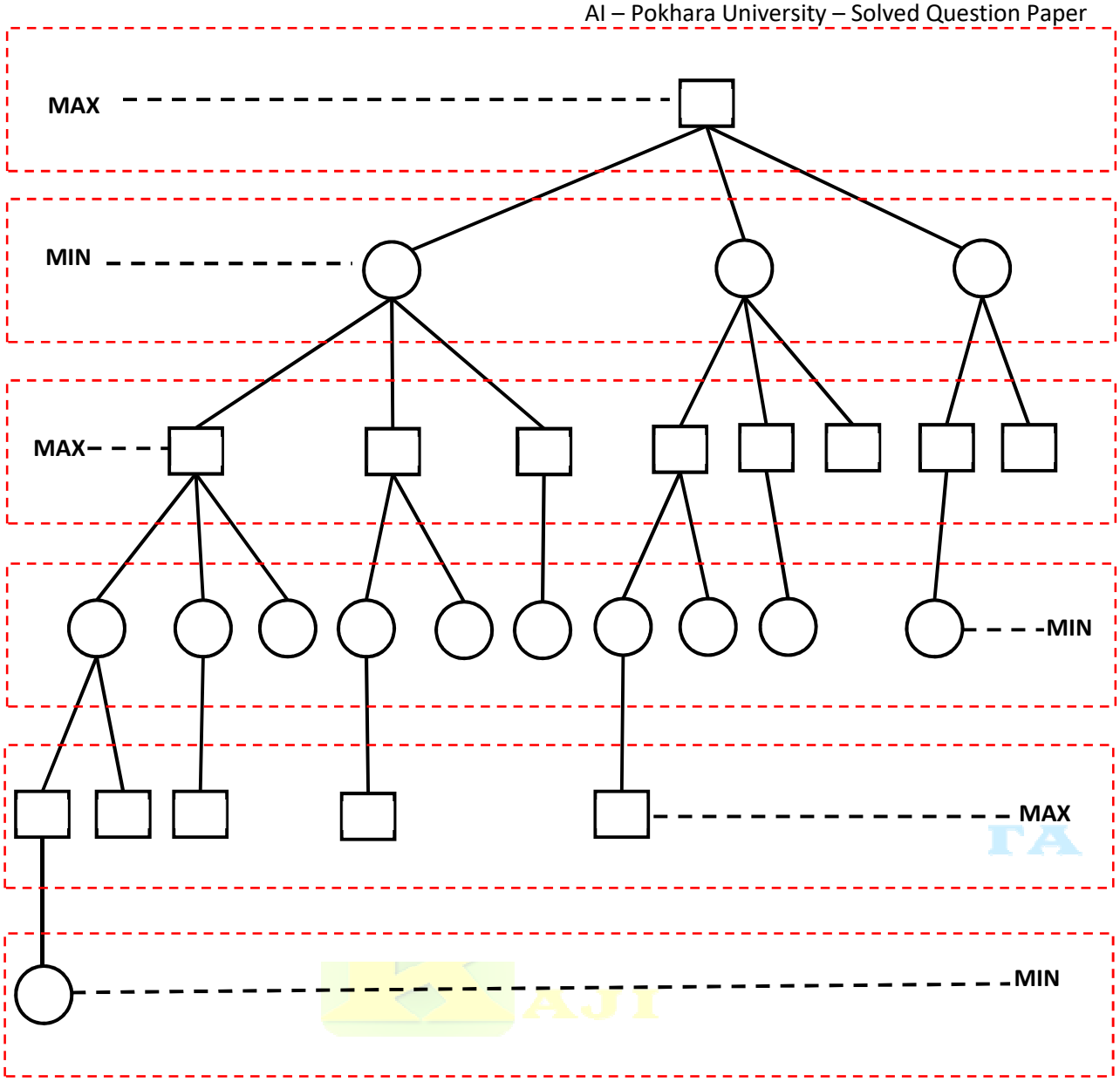
Fig: Full game tree of NIM game (1 stack with 5 coins)

ii) Show that the player who has the second move can always win.

However, it is not always possible that second move can always win, instead it depends on your NIM game strategy level how you play in the game.

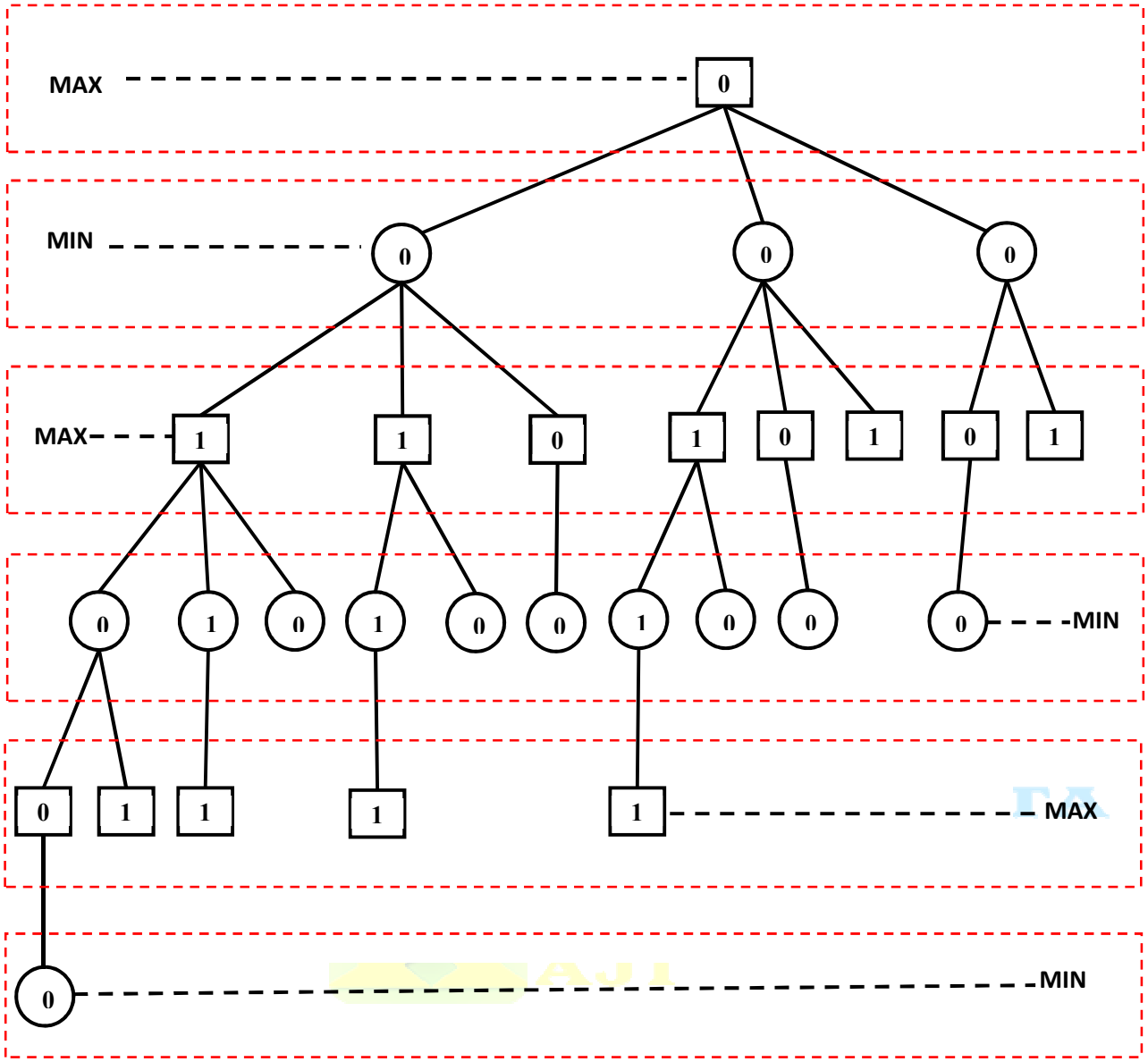
But here in **(1 stack with 5 coins)** the second move always win. We can also show the winning player using **Minimax procedure** or also by using $\boldsymbol{\alpha - \beta}$ **procedure**

Here, I have shown Step-wise-step so that you can understand easily, what actually you need to do. But I exam point of view just draw a single tree.



- If the **terminal vertices are a circle**, we label the vertex with **0** as the first player loses.
- If the **terminal vertices are a square**, we label the vertex with **1** as the first player wins.

Now, fill all the terminal vertices;



After all the terminal nodes are filled then calculate the Minimax procedure to find the winner at the root level.

$\boldsymbol{\alpha = MAX = -\infty}$

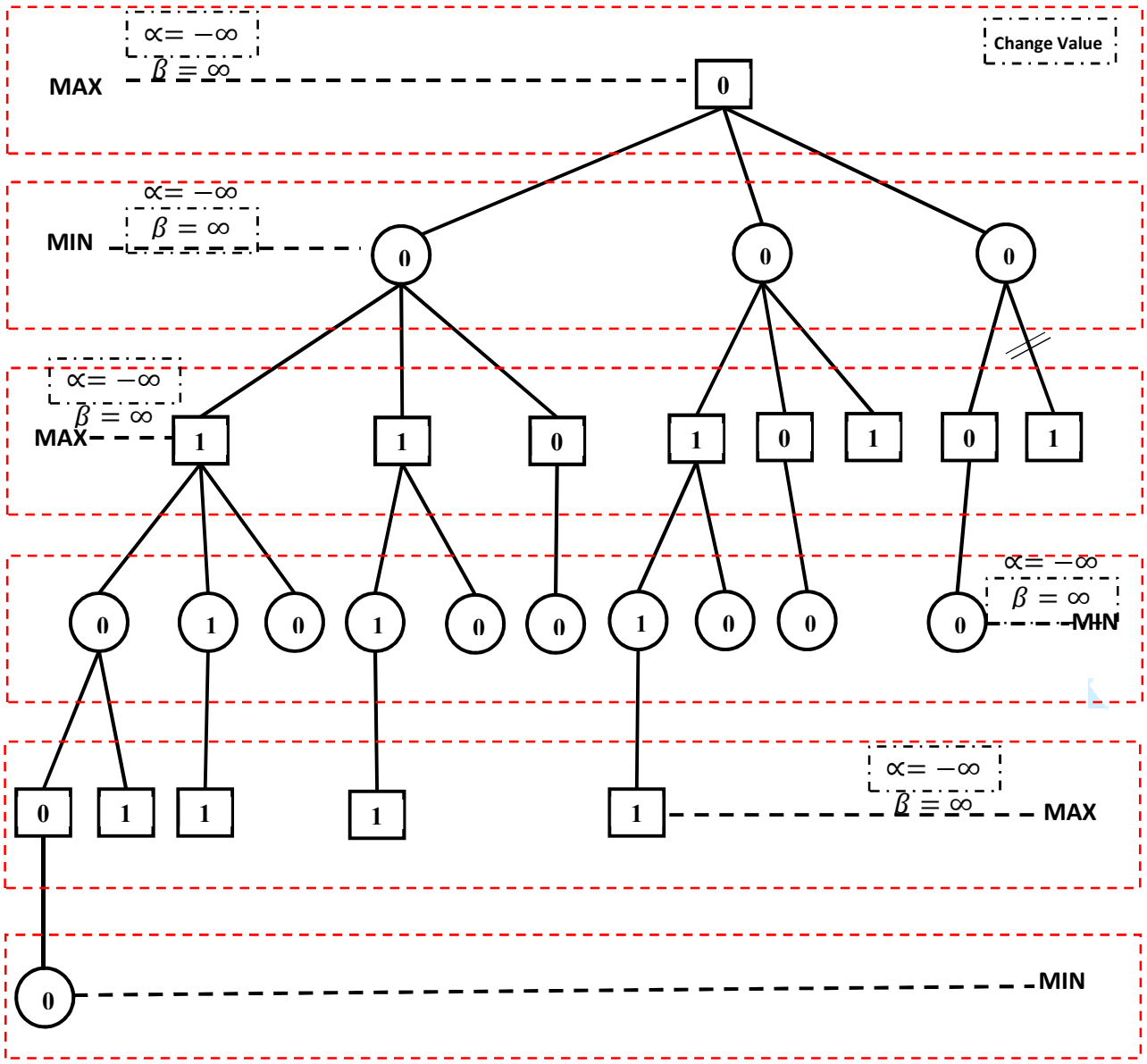
$\boldsymbol{\beta = MIN = \infty}$

Start from bottom to Top and assign the value to the node.

$\boldsymbol{MAX = (-\infty, 0) = 0}$
$\boldsymbol{MIN = (0,1) = 0}$, $\boldsymbol{MIN = (\infty, 1) = 1}$, $\boldsymbol{MIN = (\infty, 1) = 1}$, $\boldsymbol{MIN = (\infty, 1) = 1}$
$\boldsymbol{MAX = (0,1,0) = 1}$, $\boldsymbol{MAX = (1,0) = 1}$, $\boldsymbol{MAX = (-\infty, 0) = 0}$, $\boldsymbol{MAX = (1,0) = 1}$
$\boldsymbol{MAX = (-\infty, 0) = 0}$, $\boldsymbol{MAX = (-\infty, 0) = 0}$
$\boldsymbol{MIN = (1,1,0) = 0}$, $\boldsymbol{MIN = (1,0,1) = 0}$, $\boldsymbol{MIN = (0,1) = 0}$
$\boldsymbol{MAX = (0,0,0) = 0}$

∴ The root of the tree is labeled 0, thus the second player wins in an optimal strategy.

iii) Execute $\alpha - \beta$ procedure on the game tree. How many terminal nodes are examined?



After applying $\alpha - \beta$ **procedure** on the game tree **12** terminal nodes are examined.

[Fall 2016]

Q.4.a) Why do we need to represent knowledge? Convert following sentence into FOL sentence. **[8]**

i) Everyone loves Ram.
ii) Not everyone loves Ravana.
iii) People protest the politicians they dislike.
iv) Politicians can fool some of the people all of the time, all of the people some of the time.
v) Not all citizens are loyal to their employers.

Sol^{n}:

i) Everyone loves Ram.

FOL: $\forall X : \text{Loves}(X, \text{ram})$

ii) Not everyone loves Ravana.

FOL: $\neg\forall X : \text{Loves}(X, \text{ravana})$

iii) People protest the politicians they dislike.

FOL: $\forall X, \forall Y : \text{Person}(X) \wedge Politician(Y) \wedge Protest(X, Y) \rightarrow \neg like(X, Y)$

iv) Politicians can fool some of the people all of the time, all of the people some of the time.

FOL: $\forall X : \text{Politician}(X) \wedge Time(T) \rightarrow (\exists Y, \forall T : \text{Person}(Y) \wedge \text{Fool}(X, Y, T)) \wedge (\exists T, \forall Y : Person(Y) \rightarrow Fools\ (X, Y, T))$

v) Not all citizens are loyal to their employers.

FOL: $\neg\forall X, \forall Y : \text{Citizens}(X) \wedge Employee(Y) \rightarrow \text{loyalto}(X, Y)$

[Fall 2016]

Q.4.b) [*Solved in Unit 5 Note*]

[Fall 2017]

Q.2.a) Discuss Crypto Arithmetic Problem. Solve the following Constraint Satisfaction Problem. **[7]**
SEVEN + EIGHT = TWELVE

Ans: Cryptarithmetic is the science and art of creating and solving cryptarithms. A cryptarithm is a genre of mathematical puzzle in which the digits are replaced by letters of the alphabet or other symbols.

The Crypt-Arithmetic problem in Artificial Intelligence is a type of encryption problem in which the written message in an alphabetical form which is easily readable and understandable is converted into a numeric form which is neither easily readable nor understandable. In simpler words, the crypt-arithmetic problem deals with the converting of the message from the readable plain text to the non-readable ciphertext. The constraints which this problem follows during the conversion is as follows:

1. A number 0-9 is assigned to a particular alphabet.
2. Each different alphabet has a unique number.
3. All the same, alphabets have the same numbers.
4. The numbers should satisfy all the operations that any normal number does.

```
   S  E  V  E  N
   E  I  G  H  T
-----------------------------------
T  W  E  L  V  E
```

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


Other Possible Solutions are:

```
38487+89561=128048
58287+80641=138928
36465+69781=106246
85254+50671=135925
63732+39841=103573
```

[Fall 2017]

Q.3.b) Assume the following facts: **[8]**

i) Lionel Messi is a footballer.

ii) Lionel Messi plays for Barcelona.

iii) Barcelona is an "A" Division Spanish Club.

iv) All the "A" Division Spanish Clubs play La Liga.

Use Resolution to prove that **"Lionel Messi Plays La Liga".**

Sol^{n}:

i) Lionel Messi is a footballer.

FOL: $Footballer(messi)$

CNF: $Footballer(messi)$

ii) Lionel Messi plays for Barcelona.

FOL: $Plays(messi, barcelona)$

CNF: $Plays(messi, barcelona)$

iii) Barcelona is an "A" Division Spanish Club.

FOL: $A\_\text{DivisionSpanishClub}(barcelona)$

CNF: $A\_\text{DivisionSpanishClub}(barcelona)$

iv) All the "A" Division Spanish Clubs plays La Liga.

FOL: $\forall X : A\_\text{DivisionSpanishClub}(X) \rightarrow Plays(X, laliga)$

CNF: $\neg A\_\text{DivisionSpanishClub}(X) \vee Plays(X, laliga)$

To prove that **"Lionel Messi Plays La Liga"** we first go with the contradiction means it is **FALSE.**

Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"Lionel Messi not Plays La Liga":** $\neg Plays(messi, laliga)$

We have given facts in CNF:

i) $Footballer(messi)$

ii) $Plays(messi, barcelona)$

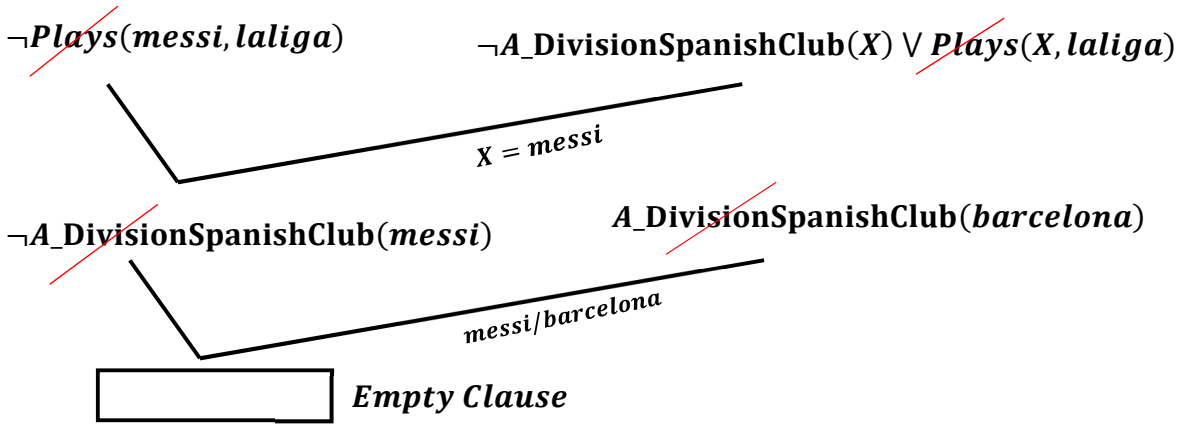
iii) $A\_\text{DivisionSpanishClub}(barcelona)$

iv) $\neg A\_\text{DivisionSpanishClub}(X) \vee Plays(X, laliga)$

Negation of Conclusion:

v) $\neg Plays(messi, laliga)$

The Resolution procedure has been illustrated below:



Here, The **EMPTY** clause shows that the $\neg plays(messi, laliga)$ is false. This produce a contradiction or $plays(messi, laliga)$will not produce contradiction with the known statement. So, we can say that **"Lionel Messi Plays La Liga".**

Thus, the given statement **"Lionel Messi Plays La Liga"** is proved.

[Fall 2017]

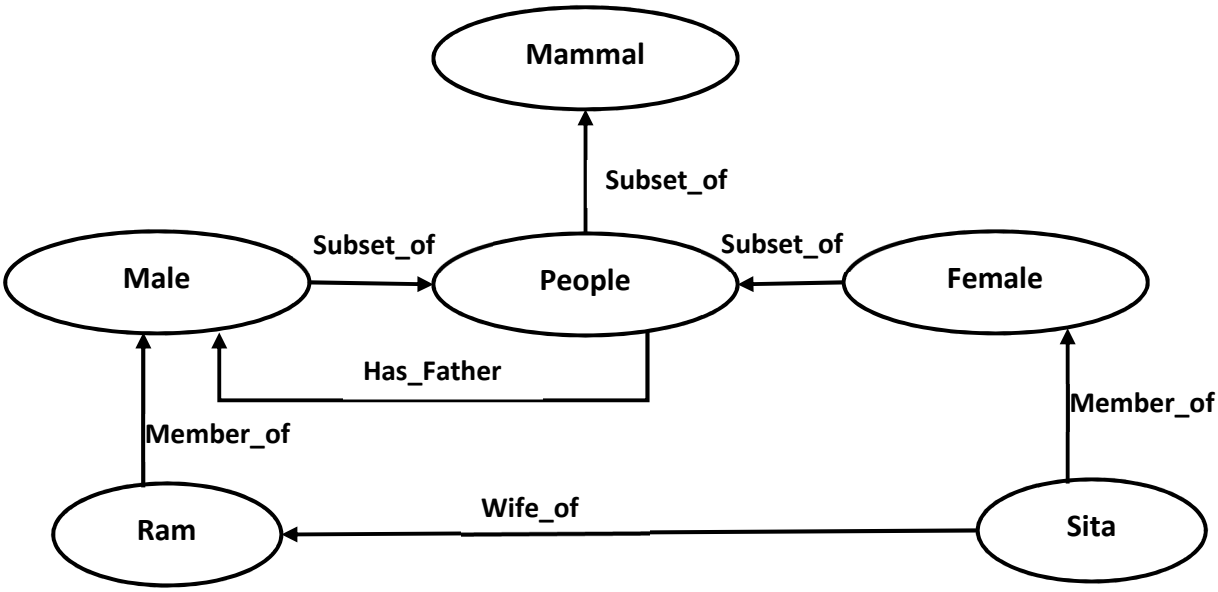
Q.4.a) Differentiate between semantic network and frames. Draw semantic network of following clauses: **[8]**

Subset-of(People, Mammal), Subset-of(Male, People), Subset_of(Female, People), Has_Father(People, Male), Member_of(Ram, Male), Member_of(Sita, Female), Wife_of(Sita, Ram)

Ans:

Semantic Networks	Frames
1. Semantic Network is a structure for representing knowledge as a pattern of interconnected nodes and arcs.	1. A frame is a static data structure that has slots for various objects.
2. Arcs in the Net give the relationship between the nodes.	2. Slots in the frame are linked with another frame slot.
3. Semantic Network are normally considered as *specifications*.	3. Frames are normally considered as *typical descriptions.*
4. Nets typically distinguish strongly between classes and instances.	4. Frames typically do instantiation at the slot level and don not have a clear-cut distinction at the frame level.
5. In Nets, **Nodes Represent:** Various objects / values of the attributes of object. **Arcs Represent:** Relationships among nodes.	5. A frame represents an entity as a set of slots (attributes) and associated values.
6. **Example:** *See the Note*	6. **Example:** *See the Note*

Following is the semantic network of above clauses:



[Spring 2017]

Q.2.a) Solve the following problem as CSP. **[7]**

LOGIC + LOGIC = PROLOG

Sol^{n}:

	L	**O**	**G**	**I**	**C**		**9**	**0**	**4**	**5**	**2**
	L	**O**	**G**	**I**	**C**		**9**	**0**	**4**	**5**	**2**
P	**R**	**O**	**L**	**O**	**G**		**1**	**8**	**0**	**9**	**0**	**4**

[Spring 2017]

Q.3.b) Prove **"Vinod buys a ticket"** using resolution considering following premises: **[8]**

Everyone who enters in a theatre has to buy a ticket. Person who doesn't have money can't buy a ticket. Vinod enter a theater.

Sol^{n}:

i) Everyone who enters in a theatre has to buy a ticket.

FOL: $\forall X : Enter(X, threater) \rightarrow buy(X, ticket)$

CNF: $\neg Enter(X, threater) \vee buy(X, ticket)$

ii) Person who doesn't have money can't buy a ticket.

FOL: $\forall X : Person(Y) \wedge \neg money(Y) \rightarrow \neg buy(Y, ticket)$

CNF: $\neg Person(Y) \vee money(Y) \vee \neg buy(Y, ticket)$

iii) Vinod enter a theater.

FOL: $Enter(vinod, theater)$

CNF: $Enter(vinod, theater)$

To prove that **"Vinod buys a ticket"** we first go with the contradiction means it is **FALSE.** Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"Vinod does not buy a ticket":** $\neg buy(vinod, ticket)$

We have given facts in CNF:

i) $\neg Enter(X, threater) \vee buy(X, ticket)$

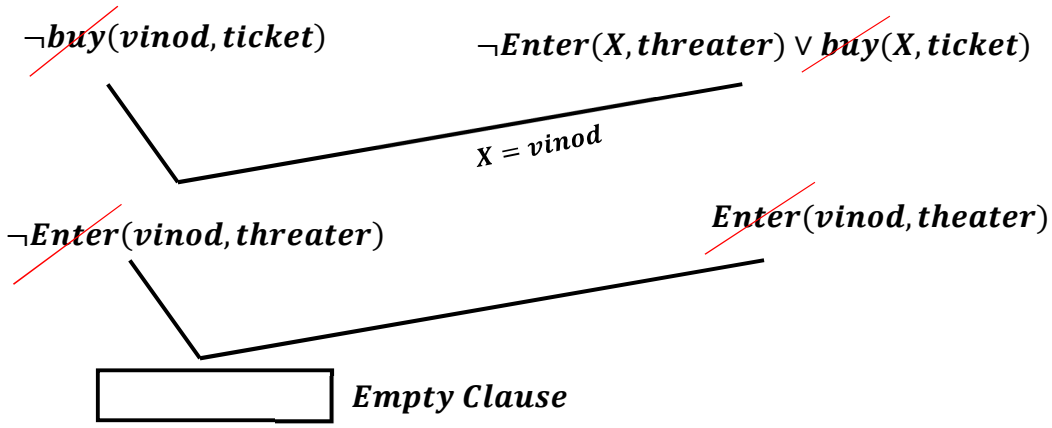
ii) $\neg Person(X) \vee money(X) \vee \neg buy(X, ticket)$

iii) $Enter(vinod, theater)$

Negation of Conclusion:

iv) $\neg buy(vinod, ticket)$

The Resolution procedure has been illustrated below:



Here, The **EMPTY** clause shows that the $\neg buys(vinod, ticket)$ is false. This produce a contradiction or $buys(vinod, ticket)$will not produce contradiction with the known statement. So, we can say that **"Vinod buys a ticket".**

Thus, the given statement **"Vinod buys a ticket"** is proved.

[Spring 2017]

Q.4.a) Convert the following statement to clause form and discuss the steps. **[8]**

$$\forall x[B(x) \rightarrow (\exists y[Q(x,y) \wedge \neg P(y)]$$

Sol^n: Here are step-by-step procedure to convert the statement $\forall x[B(x) \rightarrow (\exists y[Q(x,y) \wedge \neg P(y)]$ to clause form:

Step 1: Eliminate the implication ($\rightarrow$):

$$i.e. \quad P \rightarrow Q \equiv \neg P \vee Q$$

$\forall x[B(x) \rightarrow (\exists y[Q(x,y) \wedge \neg P(y)]$

$\forall x[\neg B(x) \vee (\exists y[Q(x,y) \wedge \neg P(y)]$

Step 2: Move the negation down to atomic formulas (by using the following rules):

- $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
- $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
- $\neg(\neg(P)) \equiv P$
- $\neg\forall x(P(x)) \equiv \exists x(\neg P(x))$
- $\neg\exists x(P(x)) \equiv \forall x(\neg P(x))$

$\forall x[\neg B(x) \vee (\exists y[Q(x,y) \wedge \neg P(y)]$ ***//Here, no need to do any thing***

Step 3: Eliminate the Existential Quantifiers:

The function that is eliminate the existential are called "Skolem function". i.e. $\exists y$

$\forall x[\neg B(x) \vee ([Q(x, f(x)) \wedge \neg P(f(x))]$

Step 4: Rename variables, as necessary, so that no two variables are the same:

$\forall x[\neg B(x) \vee ([Q(x, f(x)) \wedge \neg P(f(x))]$ ***//Here, no need to do any thing***

Step 5: Move the Universal Quantifiers to the left of the statement:

$\forall x[\neg B(x) \vee ([Q(x, f(x)) \wedge \neg P(f(x))]$ ***//Here, no need to do any thing***

Step 6: Move the Disjunction down to the literals, using Distributive laws:

- $P \vee (Q \wedge R \wedge S \wedge \ldots.) \equiv (P \vee Q) \wedge (P \vee R) \wedge (P \vee S) \wedge \ldots.$
- $P \wedge (Q \vee R \vee S \vee \ldots.) \equiv (P \wedge Q) \vee (P \wedge R) \vee (P \wedge S) \wedge \ldots.$

$\forall x[\neg B(x) \vee ([Q(x, f(x)) \wedge \neg P(f(x))]$

$\forall x[(\neg B(x) \vee (Q(x, f(x))) \wedge (\neg B(x) \vee \neg P(f(x)))$

Step 7: Eliminate the Conjunctions:

$\forall x[\neg B(x) \vee (Q(x, f(x))]$

$\forall x[\neg B(x) \vee (P(f(x))]$

Step 8: Rename all the variables, as necessary, so that no two variables are the same:

$\forall x[\neg B(x) \vee (Q(x, f(x))]$

$\forall y[\neg B(y) \vee (P(f(y))]$

Step 9: Eliminate the Universal Quantifiers:

$\neg B(x) \vee (Q(x, f(x))$

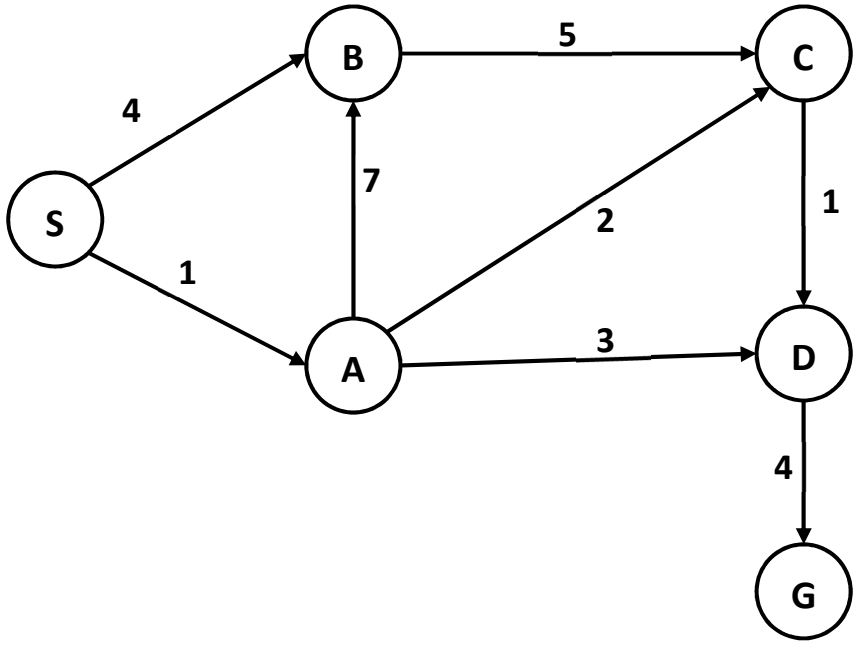
$\neg B(y) \vee (P(f(y))$

Therefore, the final Clause Form of CNF is:

$$\neg B(x) \vee (Q(x, f(x))$$
$$\neg B(y) \vee (P(f(y))$$

[Fall 2019]

Q.2.a) Using A* search algorithm, find the optimal route from S to G. **[8]**



Sol^{n}:

Here, Heuristic values are missing so let us assume heuristics values of all the states randomly,

States	Heuristics h(n)
S	6
A	7
B	5
C	2
D	4
G	0

$$f(n) = g(n) + h(n)$$

S
= 0 + 6 = 6

S→A
= 1 + 7 = 8

S→B
= 1 + 5 = 6

S→B→C
= 9 + 2 = 11 (4 + 5 + 2)

S→A→B
= 8 + 5 = 13 (1 + 7 + 5)

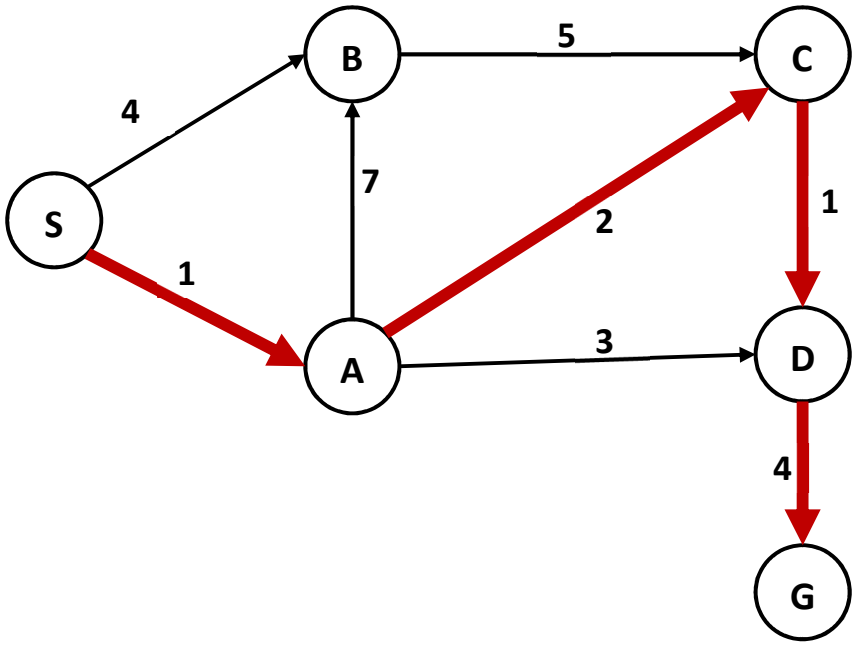
S→A→C
= 3 + 2 = 5 (1 + 2 + 2)

S→A→D
= 4 + 4 = 8 (1 + 3 + 4)

S→A→C→D
= 4 + 4 = 8 (1 + 2 + 1 + 4)

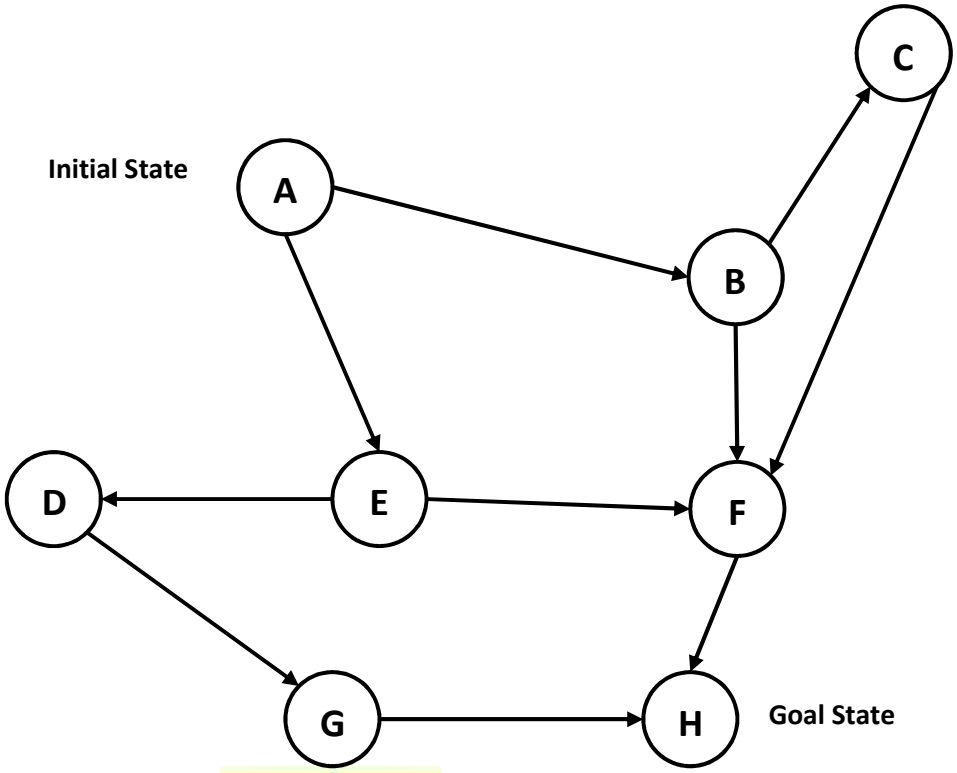
S→A→C→D→G
= 8 + 0 = 8 (1 + 2 + 1 + 4 + 0)

We will follow S→A→C→D→G path to reach the goal state because it has low cost path. Hence it is optimized.



[Fall 2019]

Q.2.b) Consider a graph of cities as shown in the below figure: **[8]**



i) Represent state space of this graph using prolog.

ii) Write a simple search algorithm in prolog to reach to the goal state from initial state.

Solⁿ:

i) **Represent state space of this graph using prolog:**

Make a link from one node to another node. Means if you have a node P → Q then you have to write a predicate name with two nodes as an arguments/objects;

Link (P, Q). Here **Link** is a predicate name and **P, Q** are two arguments/objects.

How you can read this prolog in English statement is: P links Q.

Like wise make all the State-Space;

```
link(a, b).
link(a, e).
link(b, c).
link(b, f).
link(c, f).
link(d, g).
link(e, d).
link(e, f).
link(f, h).
link(g, h).
```

ii) Write a simple search algorithm in prolog to reach to the goal state from initial state:

Simple Search Algorithm in Prolog:

$$go(X, X, [X]).$$
$$go(X, Y, [X|T]) :- link(X, Z), go(Z, Y, T).$$

Consultation in Prolog ***(When you run the prolog program)***:

$?- go(a, h, X).$

$X = [a, b, c, f, h]$;

$X = [a, b, f, h]$;

$X = [a, e, d, g, h]$;

$X = [a, e, f, h]$;

$false.$

exam solution 2.pl

File Edit Browse Compile Prolog Pce Help

exam solution 2.pl

```
link(a, b).
link(a, e).
link(b, c).
link(b, f).
link(c, f).
link(d, g).
link(e, d).
link(e, f).
link(f, h).
link(g, h).


go(X,X,[X]).
go(X,Y,[X|T]):- link(X,Z),go(Z,Y,T).
```

SWI-Prolog -- c:/Users/dpkbh/Desktop/All LABS/AI Lab/Exam Solution 2.pl

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/users/dpkbh/desktop/all labs/ai lab/exam solution 2 compiled 0.00 sec, -1 clauses
?- go(a,h,X).
X = [a, b, c, f, h] ;
X = [a, b, f, h] ;
X = [a, e, d, g, h] ;
X = [a, e, f, h] ;
false.

?-
```

[Fall 2019]

Q.3.b) Represent the following statement using predicate logic.

i) Anyone who buys carrots by the bushel owns either a rabbit or a grocery store.

ii) Every dog chases some rabbit.

iii) Someone who hates something owned by another person will not date that person.

iv) If Mary does not own a grocery store, she will not date John.

Sol^{n}:

i) Anyone who buys carrots by the bushel owns either a rabbit or a grocery store.

FOL: $\forall X : (\text{Buy}(X) \rightarrow \exists Y : \big(\text{Owns}(X, Y) \wedge \big(\text{Rabbit}(Y) \vee \mathit{Grocery}(Y)\big)\big)$

ii) Every dog chases some rabbit.

FOL: $\forall X : \big(\text{Dog}(X) \rightarrow \exists Y \, \big(\text{Rabbit}(Y) \wedge \text{Chase}(X, Y)\big)\big)$

iii) Someone who hates something owned by another person will not date that person.

FOL: $\forall X : \forall Y : \forall Z \ : \big(\text{Owns}(Y, Z) \wedge \text{Hates}(X, Z) \rightarrow \neg\text{Date}(X, Y)\big)$

iv) If Mary does not own a grocery store, she will not date John.

FOL: $\Big(\big(\neg\exists X : \big(\text{Grocery}(X) \wedge \text{Own}(\text{mary}, X)\big)\big) \rightarrow \neg\text{Date}(\mathit{mary}, \text{john})\Big)$ **OR**

$\forall X : \big(\text{Grocery}(X) \ \wedge \neg\text{Own}(\text{mary}, X)\big) \rightarrow \neg\text{Date}(\mathit{mary}, \text{john})$

[Fall 2019] Q.5.a) ***See the Solution in Unit 5.***

All the Theory Question's Answers are there in respective Unit. Please read it well.

[Spring 2019]

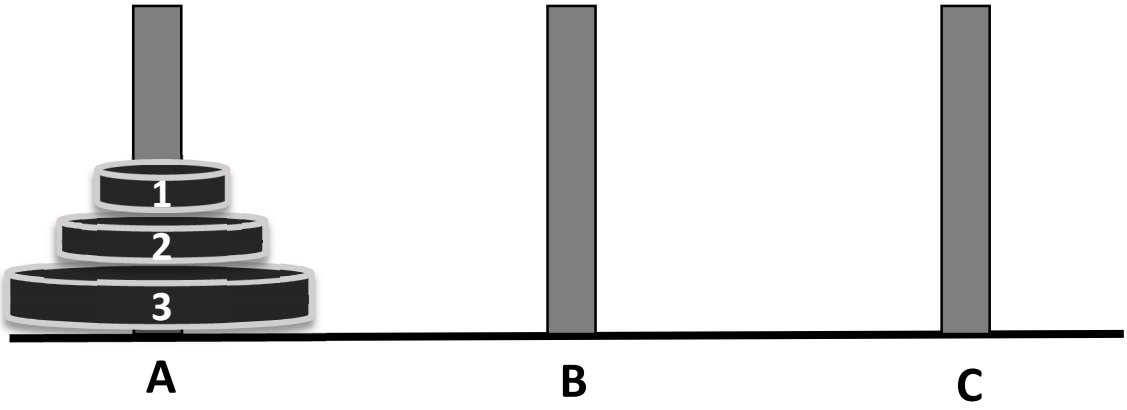
Q.2.a) There are three pegs, labelled A, B and C. There are 3 disks on peg. **[8]**
The top disk has a diameter of 1, the middle disk has a diameter of 2, and the bottom disk has a diameter of 3. There are no disks on peg C. You must move one disk at time and you cannot place a larger disk on top of smaller disk. The problem is to get all the disks on peg C.

i) Find a representation for the states of this problem.

ii) Describe all of the operators that might be applied to a state.

Sol^{n}: There are three pegs, labelled A, B and C. On peg A we have 3 disks with different diameters and are labelled as (1, 2, 3) in increasing of diameter from top. The top disk has a diameter of 1, the middle disk has a diameter of 2, and the bottom disk has a diameter of 3.

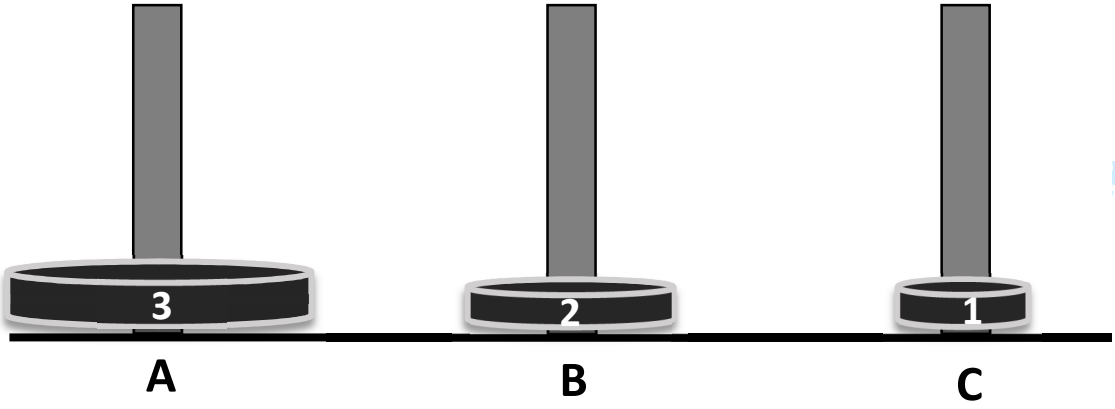
i) States of this problem:



ii) Actions:

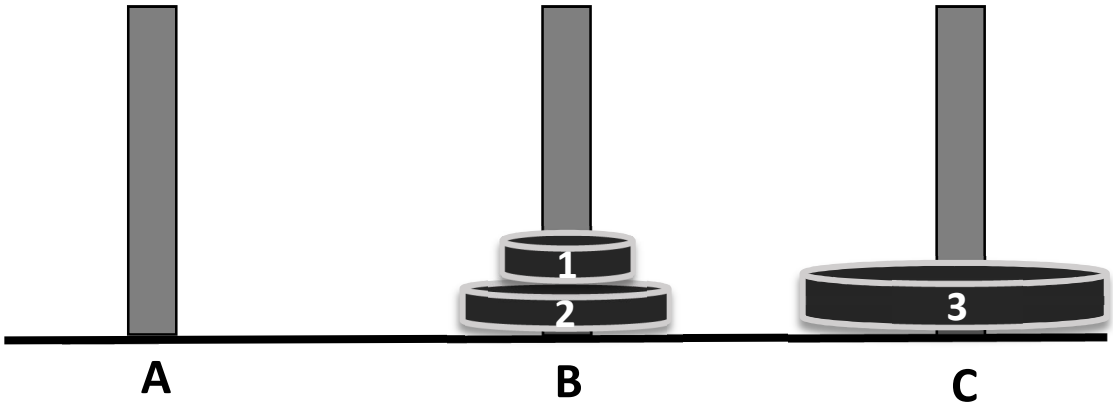
Step 1: Place disk 1 of peg A on peg C and disk 2 of peg A on peg B.

Now, each peg has only one disk. (Peg A→ disk 3, Peg B→ disk 2, Peg C→ disk 1).

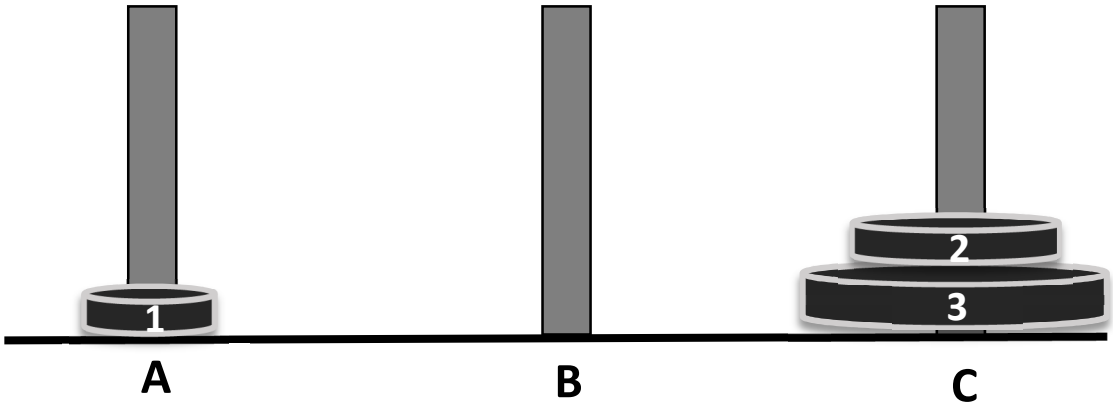


Step 2: Place disk 1 of peg C on the top of disk 2 of peg B and move disk 3 of peg A to peg C.

Now, Peg A has 0 disk ***(no disk),*** Peg B has 2 disks ***(disk 2 at bottom and disk 1 at top of disk 2),*** Peg C has 1 disk ***(disk 3 only).***

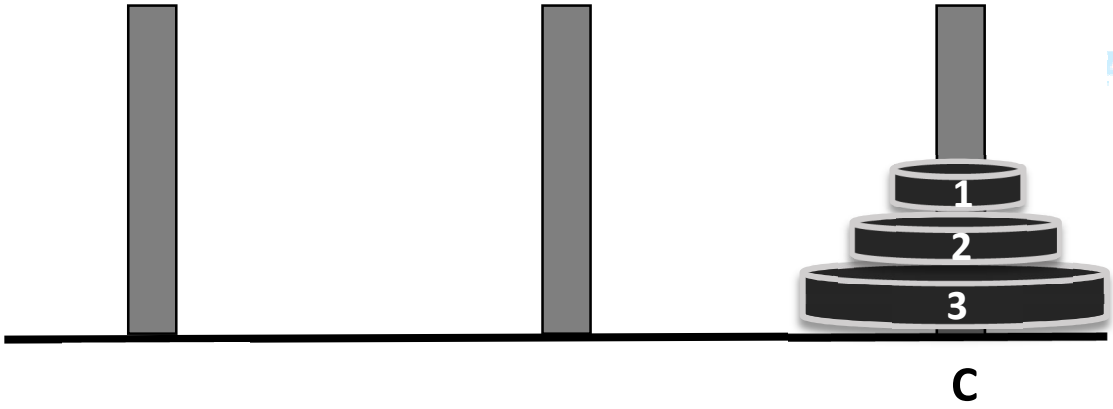


Step 3: Place disk 1 of peg B on peg A and disk 2 of peg B on top of disk 1 of peg C. Now, Peg A has 1 disk ***(disk 1 only),*** Peg B has 0 disk ***(no disk),*** Peg C has 2 disks ***(disk 3 at its bottom, on its top disk 2 placed).***



Step 4: Place the disk 1 of peg A to peg C.

Now, Peg A has 0 disk ***(no disk),*** Peg B has 0 disk ***(no disk),*** Peg C has 3 disks ***(disk 3 at its bottom, disk 2 in middle and on its top disk 1 placed).***



To get the final result we could have total 7 moves and we have performed in 4 states.

[Spring 2019]

Q.2.b) ***See the Solution in Unit 2.***

All the Theory Question's Answers are there in respective Unit. Please read it well.

[Spring 2019]

Q.4.a) Prove **"You are not doing strawberry picking"** using resolution: **[8]**

i) If it is sunny and warm day you will enjoy.

ii) If it is warm and pleasant day you will do strawberry picking.

iii) If it is raining then no strawberry picking.

iv) If it is raining you will get wet.

v) It is warm day.

vi) It is raining.

vii) It is sunny.

Sol^{n}: *A formal and genuine way to solve the above problem is illustrated as below:*

Any propositional statements can be transformed into conjunctive normal form using **AND (∧)** between the clauses. So, to convert the propositional statements into CNF, we write **AND** between each clause. In order to convert the 7 statements into conjunctive normal form, we first note that all 7 of the statements are assumed to be true. Hence, we are actually saying that the following single statement is true.

(1) and (2) and (3) and (4) and (5) and (6) and (7)

Here, **AND** is replaced by ∧ to show them in conjunction of clauses (in CNF). Thus, it will become: (1) ∧ (2) ∧ (3) ∧ (4) ∧ (5) ∧ (6) ∧ (7)

i) If it is sunny and warm day you will enjoy.

FOL: $(sunny \wedge warm) \rightarrow enjoy$ $\quad P \rightarrow Q \text{ is equivalent to } \neg P \vee Q$

CNF: $(\neg(sunny \wedge warm) \vee enjoy) \wedge \equiv$ $\quad \neg(P \wedge Q) \text{ is equivalent to } \neg P \vee \neg Q$

$(\neg sunny \vee \neg warm \vee enjoy) \wedge$

ii) If it is warm and pleasant day you will do strawberry picking.

FOL: $(warm \wedge pleasant) \rightarrow strawberry\_picking$

CNF: $(\neg(warm \wedge pleasant) \vee strawberry\_picking) \wedge \equiv$

$(\neg warm \vee \neg pleasant \vee strawberry\_picking) \wedge$

iii) If it is raining then no strawberry picking.

FOL: $(rainng) \rightarrow \neg strawberry\_picking$

CNF: $(\neg raining \vee \neg strawberry\_picking) \wedge$

iv) If it is raining you will get wet.

FOL: $raining \rightarrow wet$

CNF: $(\neg raining \vee wet) \wedge$

v) It is warm day.

FOL: $warm$

CNF: $(warm) \wedge$

vi) It is raining.

FOL: $rainning$

CNF: $(rainning) \wedge$

vii) It is sunny.

FOL: $sunny$

CNF: $(sunny)$

To prove that **"You are not doing strawberry picking"** we first go with the contradiction means it is **FALSE.**

Given that we should make the prove statement as FALSE ***(means ~ negation of conclusion)*** so that our result will get TRUE which means result will return an EMPTY statement.

Let us assume **"You are doing strawberry picking":** $strawberry\_picking$

We have given facts in CNF:

i) $(\neg sunny \vee \neg warm \vee enjoy)$

ii) $(\neg warm \vee \neg pleasant \vee strawberry\_picking)$

iii) $(\neg raining \vee \neg strawberry\_picking)$

iv) $(\neg raining \vee wet)$

v) $(warm)$

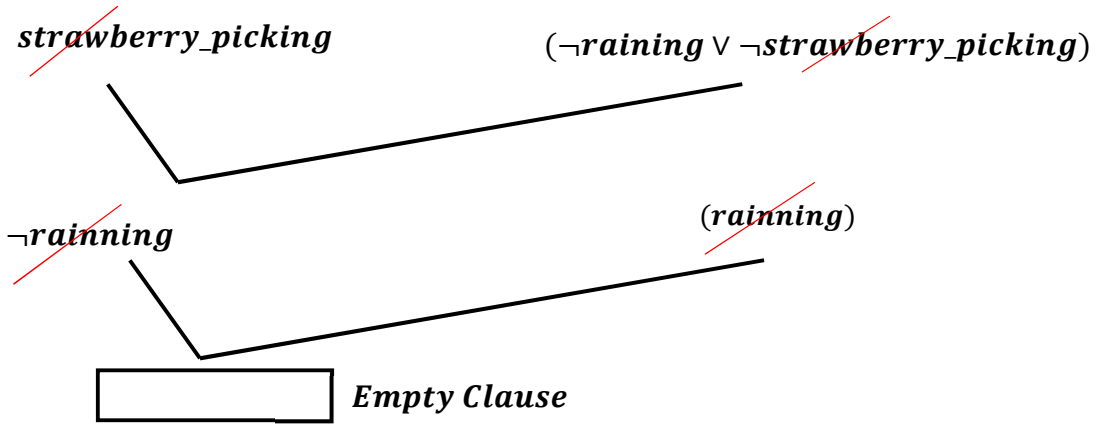
vi) $(rainning)$

vii) $(sunny)$

Negation of Conclusion:

viii) $strawberry\_picking$

The Resolution procedure has been illustrated below:



Here, The **EMPTY** clause shows that the ***strawberry_picking*** is false. This produce a contradiction or ***¬strawberry_picking*** will not produce contradiction with the known statement. So, we can say that **"You are not doing strawberry picking".**

Thus, the given statement **"You are not doing strawberry picking"** is proved.

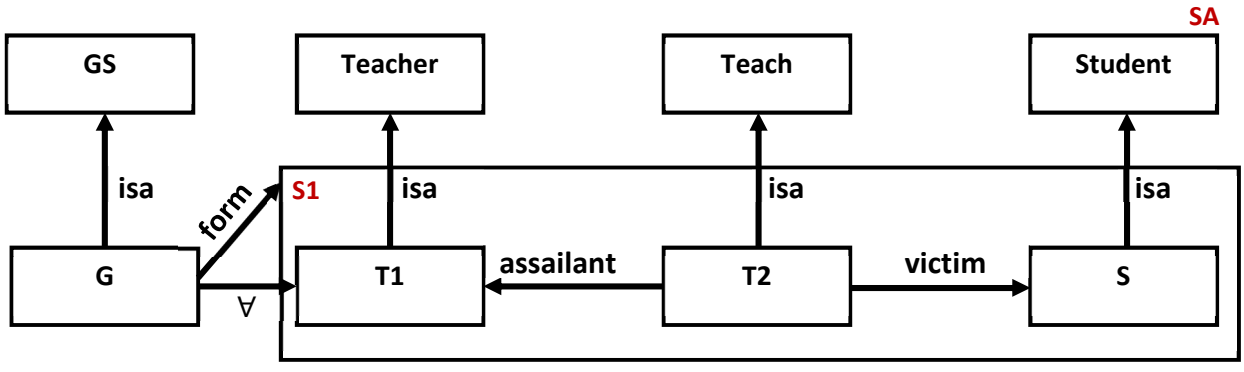
[Spring 2019]

Q.4.b) What is semantic Net? Represent the following fact using partitions semantic Nets. "Every teacher has taught a student." **[8]**

Ans:

[Definition is already discussed in above questions]

Partitioning Semantic Net is shown in below:



GS→ General Statement.

SA→ Space corresponding to whole event.

S1→ Space corresponding to own event.

[Spring 2019]

Q.5.a) ***See the Solution in Unit 5.***

All the Theory Question's Answers are there in respective Unit. Please read it well.

[Spring 2019]

Q.5.b) Assume that the following facts are already entered into the PROLOG database:

$father(dashrath, ram).$ **//Dasharath is the father of Ram**

$father(ram, luv).$

$father(ram, kush).$

$mother(kaushalya, ram).$ **//Kaushalya is the mother of Ram**

$male(dashrath).$ **//Dasharath is a male**

$female(kaushalya).$ **//Kaushalya is a female**

$parent(X, Y) :- father(X, Y); mother(X, Y).$ **//X is a parent of Y**

Sol^{n}: If we consult the above database and run it into PROLOG terminal we will get:

(Also, in exam write like this way you will secure good marks!)

$?- \textbf{parent}(X, Y).$	$?- \textbf{father}(X, Y).$
$X = dashrath,$ $Y = ram\ ;$ $X = ram,$ $Y = luv\ ;$ $X = ram,$ $Y = kush\ ;$ $X = kaushalya,$ $Y = ram.$	$X = dashrath,$ $Y = ram\ ;$ $X = ram,$ $Y = luv\ ;$ $X = ram,$ $Y = kush.$

$?- \textbf{mother}(X, Y).$
$X = kaushalya,$ $Y = ram.$

examsolution1.pl

File Edit Browse Compile Prolog Pce Help

examsolution1.pl

```
father(dashrath,ram).
father(ram,luv).
father(ram,kush).
mother(kaushalya,ram).
male(dashrath).
female(kaushalya).
parent(X,Y):-father(X,Y);mother(X,Y).
```

SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/users/dpkbh/desktop/all labs/ai lab/examsolution1 compiled 0.00 sec, -1 clauses
?- parent(X,Y).
X = dashrath,
Y = ram ;
X = ram,
Y = luv ;
X = ram,
Y = kush ;
X = kaushalya,
Y = ram.

?- father(X,Y).
X = dashrath,
Y = ram ;
X = ram,
Y = luv ;
X = ram,
Y = kush.

?- mother(X,Y).
X = kaushalya,
Y = ram.
```

[Spring 2019]

Q.6.a) ***See the Solution in Unit 5.***

All the Theory Question's Answers are there in respective Unit. Please read it well.

Good Luck!

********Best Wishes for Board Examination********

THE END
