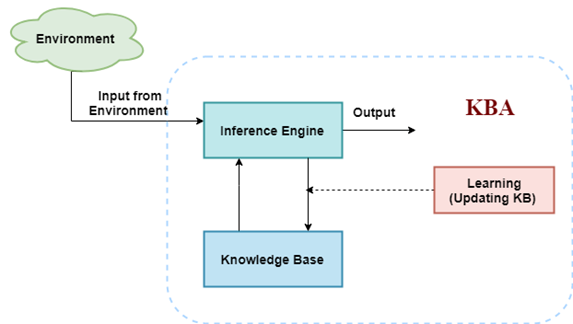
**UNIT2**

1. **Describe Knowledge based agents with a neat diagram. Provide pseudocode for Generic knowledge based agent.**

ans) Knowledge based agents are those agents that use some task-specific knowledge to solve a problem efficiently. They consist of a knowledge base and an inference system.

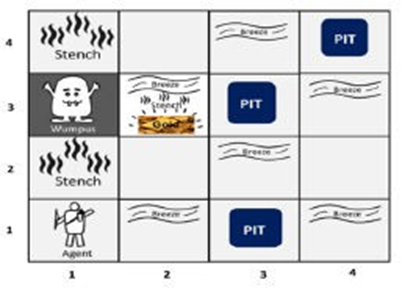
The pseudocode for a generic knowledge based agent is given below:



| function KB-AGENT(percept): //input is percept  static: KB, a knowledge base //could be persistent also  t, a counter// initially 0, indicating time  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t)) //Tell KB what it perceives   action = ASK(KB, MAKE-ACTION-QUERY(t)) //Ask KB what action should it take   TELL(KB, MAKE-ACTION-SENTENCE(action, t))//Tell KB what action was chosen  t=t+1 //Increment counter  return action |
| --- |

1. **Explain the Wumpus world with a diagram.**

Sol: The Wumpus World is a toy problem to illustrate the working of a KB agent. It consists of a cave of rooms connected with passageways.



The goal of the agent is to pick the gold present in a particular room. The PEAS description of Wumpus World is given below:

Performance measure: Gold +1000, Death -1000, Step -1, Arrow -10

Environment: 4X4 Grid of rooms, Agent starts at (1,1)

Actuators: Left turn, right turn, forward step, grab, release, shoot Sensors: Stench, Breeze, Bump, Glitter, Scream

1. **Define the terms Equivalence, Validity and Satisfiability as applied to propositional logic. Give suitable example in each case.**

**Sol:**

***Equivalence***: Two sentences a and b are logically equivalent if they are **true in the same set of models**. eg: p AND q, q AND p are Equivalent . a is equivalent to b if a entails b and b entails a.

***Validity***: A sentence is valid if it is **true in all models.** eg. p OR ~p. They are called tautologies.

***Satisfiability***: A compound proposition is satisfiable if there's **at least one TRUE result**

in truth table. eg. p AND q. All sentences except fallacies are satisfiable.

1. **Write the 7 steps in Knowledge Engineering Processs.**

Sol: Knowledge engineering process is the general process of constructing knowledge base. The seven steps are as follows:

1. **Identify** the task

2. **collect** the relevant knowledge

3. **Decide** on a vocabulary of predicates, functions, and constants

4. **Encode** general knowledge about the domain

5. **Encode** a description of specific problem instance

6. Pose **queries** to the inference procedure and get answers

7. **Debug** the knowledge base

1. **State Definite Clause and horn Clause with an example and also mention their relevance in the inference process.**

Sol:

**Horn clause**

A clause which is a disjunction(∨) of literals with at most one positive literal.

Example : ¬A ∨ ¬C ∨ D

**Definite clause / Strict Horn clause**

It has exactly one positive literal.

Example: ¬p ∨ ¬q ∨ ... ∨ ¬t ∨ u

**Relevance:**

1. Every definite clause can be **written as an implication** whose premise

is a conjunction of positive literals and whose conclusion is a single

positive literal.

2. **Inference with Horn clauses** can be done through the forward-

chaining and backward chaining algorithms

3. **Deciding entailment** with Horn clauses can be done in time that is

linear in the size of the knowledge base

**Literal**:it is either propositional variable (p) or negated propositional variable (-p)

Types of horn clauses:

* Definite clauses: it has exactly one positive literals ex:¬p ∨ ¬q ∨ ... ∨ ¬t ∨ u
* Unit clauses: it doesn't has negative literals ex : u
* Goal clauses: it doesn't has positive literals ex:¬p ∨ ¬q ∨¬e

1. **Discuss the algorithm to convert any given predicate statement into clause form. Give examples in each step.**

**OR**

**Illustrate the procedure of converting any given sentence into Conjunctive Normal form*(CNF)* (in propositional Logic).**

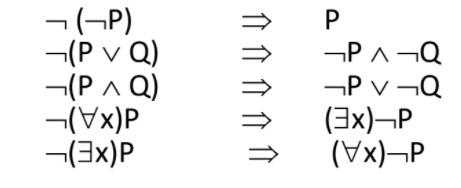
Sol: The algorithm to convert any given predicate to clause form is given below:

1. Eliminate biconditionals and implications (by definition)

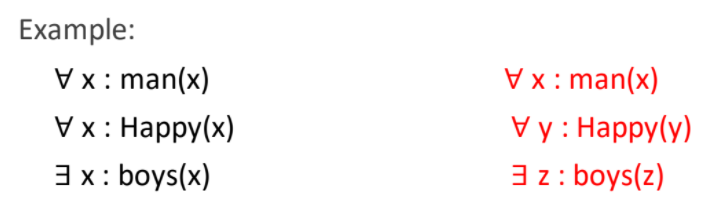
(t↔q)=(t→q)∧(q→t)

(t→q)=(¬t∨q)

2. Reduce the scope of negation (move it inwards)

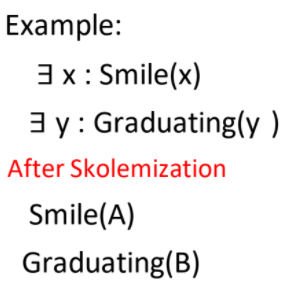


3. Standardize variables apart (give each quantifier a unique variable name)

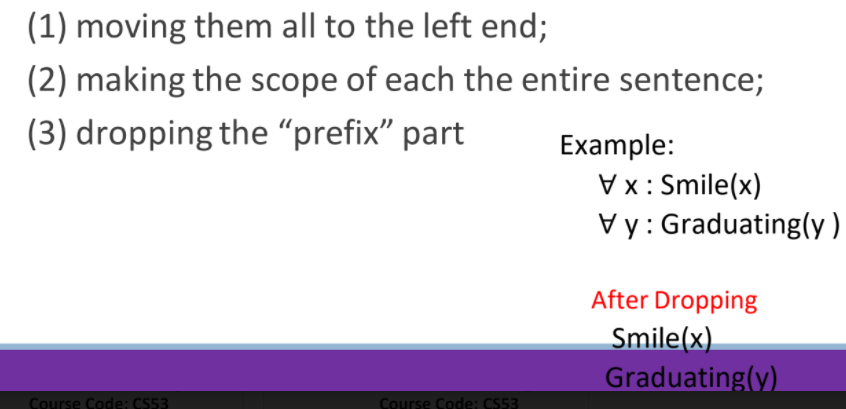


4. Move all quantifiers to the left

5. Eliminate existential quantifiers (skolemization)



6. Drop all universal quantifiers



7. Convert to connect of disjunct form (POS form)

8. Make each conjunct a separate clause

9. Standardize variables apart again (so each clause contains variable names that don't occur in other clauses)

1. **Put the following predicate calculus expression in clause form:**

**∀ (X) (p(X) → { ∀ (Y) [p(Y) → p(f(X, Y))] ∧ ¬ ∀ (Y) [q(X, Y) → p(Y)]})**

**Solution:**

1. Eliminate conditionals (→) using: a → b ≡ ¬ a ∨ b

∀ (X) (¬ p(X) ∨ { ∀ (Y) [¬ p(Y) ∨ p(f(X, Y))] ∧ ¬ ∀ (Y) [¬ q(X, Y) ∨ p(Y)]})

2. When possible, eliminate negations or reduce their scope using

¬ (¬ a) ≡ a

¬ (a ∨ b) ≡ ¬ a ∧ ¬ b ¬ (∃X) a(X) ≡ (∀X) ¬ a(X)

¬ (a ∧ b) ≡ ¬ a ∨ ¬ b ¬ (∀X) a(X) ≡ (∃X) ¬ a(X)

∀ (X) (¬ p(X) ∨ { ∀ (Y) [¬ p(Y) ∨ p(f(X, Y))] ∧ ∃ (Y) [q(X, Y) ∧ ¬ p(Y)]})

3. Standardize variables (each quantifier has a unique variable name)

∀ (X) (¬ p(X) ∨ { ∀ (Y) [¬ p(Y) ∨ p(f(X, Y))] ∧ ∃ (Z) [q(X, Z) ∧ ¬ p(Z)]})

4. Move all ∀ to the front without changing their order

∀ (X) ∀ (Y) (¬ p(X) ∨ { [¬ p(Y) ∨ p(f(X, Y))] ∧ ∃ (Z) [q(X, Z) ∧ ¬ p(Z)]})

5. Eliminate existential quantifiers using Skolem functions

Since the existential quantified Z is within the scope of universal quantified X (not Y), we replace Z with a skolem function g(X):

∀ (X) ∀ (Y) (¬ p(X) ∨ { [¬ p(Y) ∨ p(f(X, Y))] ∧ [q(X, g(X)) ∧ ¬ p(g(X))]})

6. Drop the universal quantifiers as necessary

¬ p(X) ∨ { [¬ p(Y) ∨ p(f(X, Y))] ∧ [q(X, g(X)) ∧ ¬ p(g(X))]}

7. Convert the expression to conjunctive normal form using: p ∨ (q ∧ r) = (p ∨ q) ∧ (p ∨ r)

⇒ ¬ p(X) ∨ { [¬ p(Y) ∨ p(f(X, Y))] ∧ [q(X, g(X)) ∧ ¬ p(g(X))]}

⇒ {¬ p(X) ∨ [¬ p(Y) ∨ p(f(X, Y))]} ∧ {¬ p(X) ∨ [q(X, g(X)) ∧ ¬ p(g(X))]}

⇒ [¬ p(X) ∨ ¬ p(Y) ∨ p(f(X, Y))] ∧ {[¬ p(X) ∨ q(X, g(X))] ∧ [¬ p(X) ∨ ¬ p(g(X))]}

⇒ [¬ p(X) ∨ ¬ p(Y) ∨ p(f(X, Y))] ∧ [¬ p(X) ∨ q(X, g(X))] ∧ [¬ p(X) ∨ ¬ p(g(X))]

8. Eliminate ∧ signs by writing the expression as a set of clauses

¬ p(X) ∨ ¬ p(Y) ∨ p(f(X, Y))

¬ p(X) ∨ q(X, g(X))

¬ p(X) ∨ ¬ p(g(X))

9. Rename variables in clauses so that each clause has unique variable name

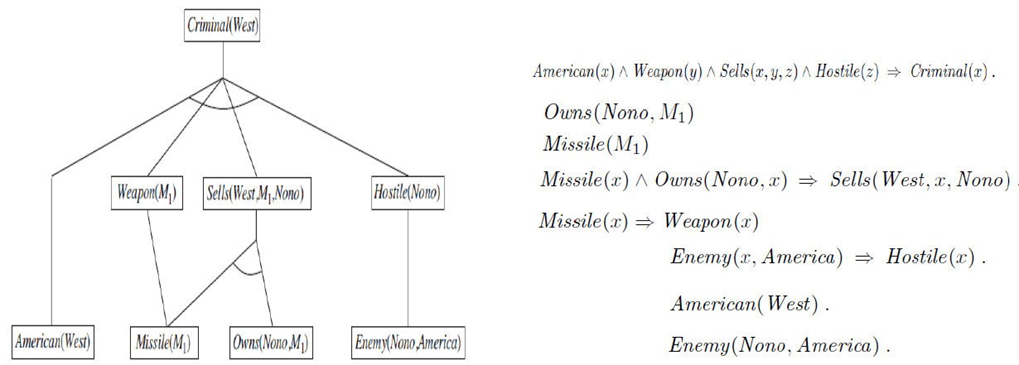
¬ p(X) ∨ ¬ p(Y) ∨ p(f(X, Y))

¬ p(U) ∨ q(U, g(U))

¬ p(V) ∨ ¬ p(g(V))

1. **Apply forward chaining algorithm to the following, to prove that West is a criminal.**

**The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.**



1. **Explain the steps involved in Resolution Refutation proof in detail. Give examples for each step.**

Sol: The steps involved in the Resolution Refutation proof method are given below:

1. Put the premises or axioms in clause form

2. Add the negation of what it to be proved, in clause form, to the set of axioms

3. Resolve these clauses together, producing new clauses that logically follow from them

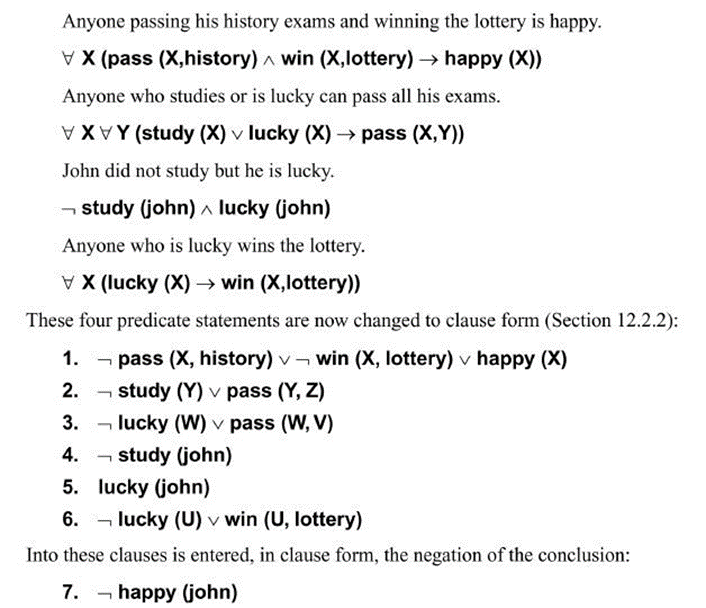
4. Produce a contradiction by generating the empty clause

5. The substitutions used to produce the empty clause are those under which the opposite of the negated goal is true.

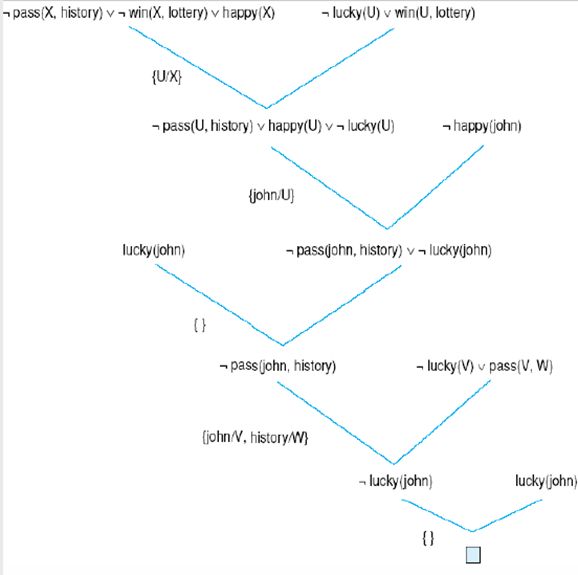
**(give examples for each)**

1. **Given the statements: “Anyone passing his history exams and winning the lottery is happy. Anyone who studies or is lucky can pass all his exams. John did not study but he is lucky. Anyone who is lucky wins the lottery. Prove by resolution process the statement “John is happy”. (V IMP)**

Sol: First, convert to clause form:

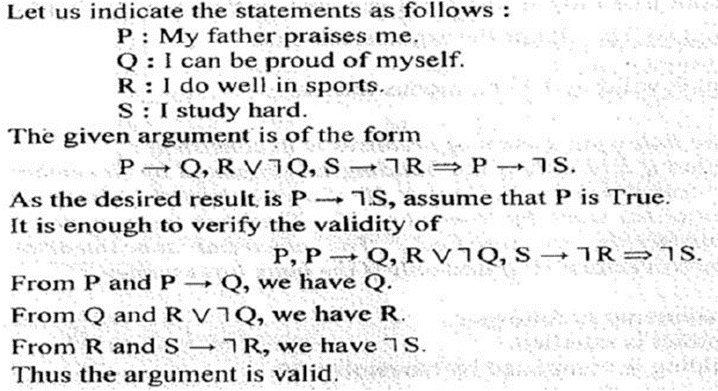


Then, use resolution graph to generate empty clause:



1. **Determine the validity of the following argument using propositional logic:**

My father praises me only if I can be proud of myself. Either I do well in sports or I cannot be proud of myself. If I study hard, then I cannot do well in sports. Therefore, if father praises me, then I do not study well.



1. **Explain Quantifiers**

A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.

These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression.

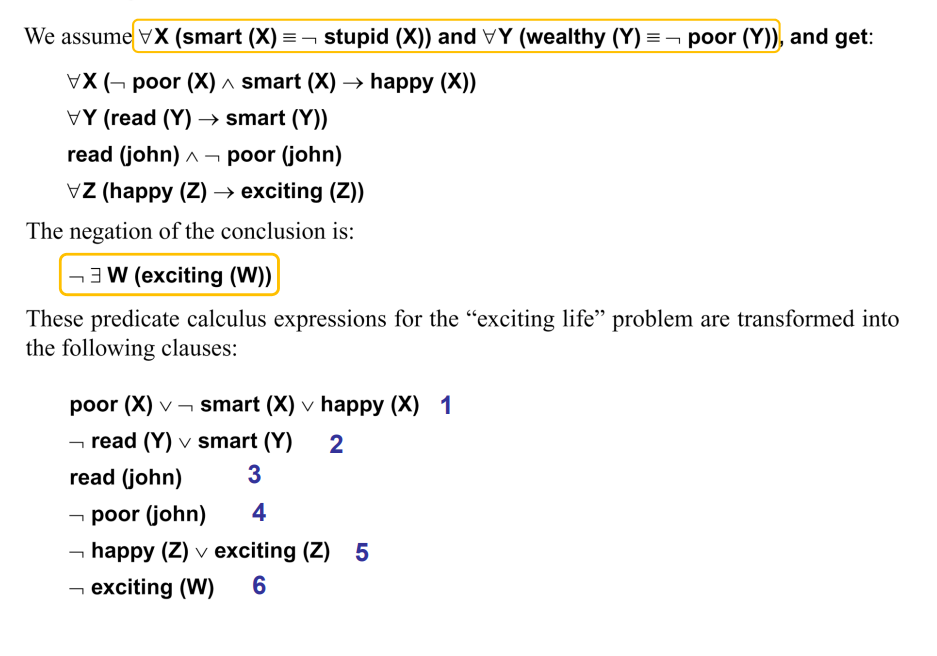
There are two types of quantifier:

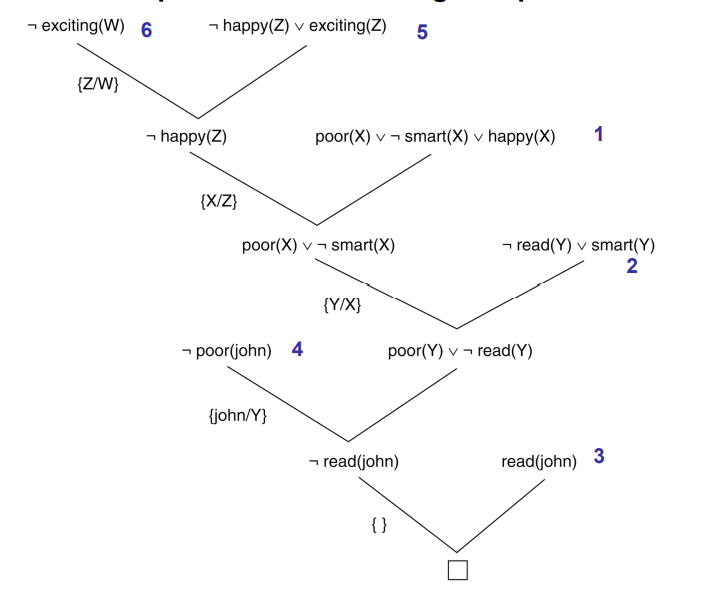
◦ *Universal Quantifier*(for all, everyone, everything)

◦ *Existential quantifier*(for some, at least one).

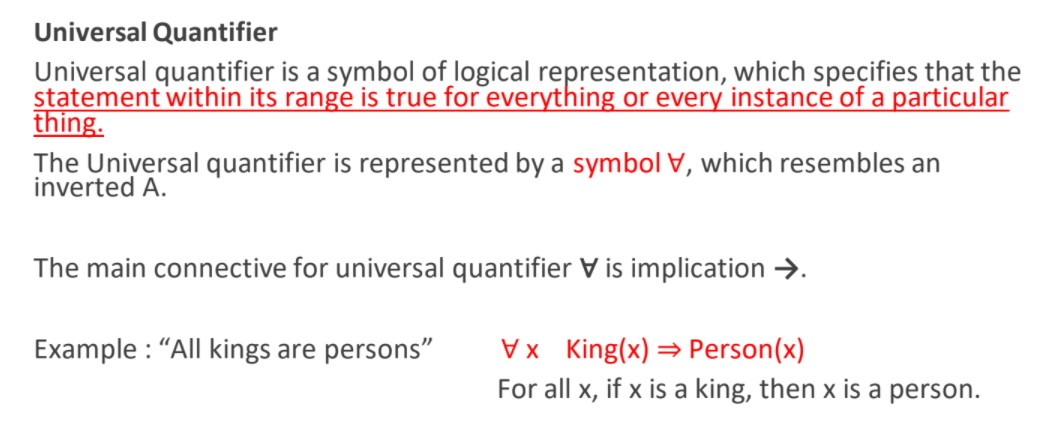
(Elaborate for more marks q)

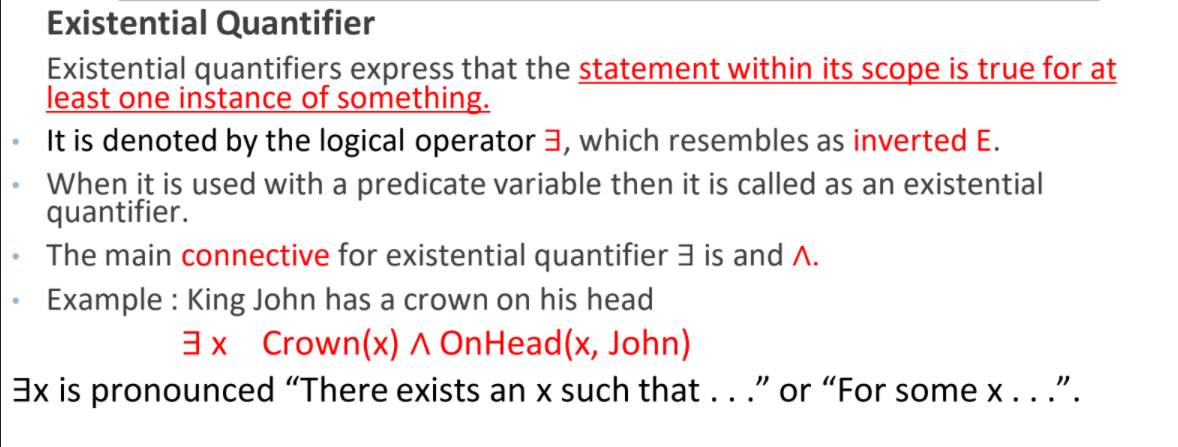
1. **Solve the below query using the answer extraction method. All people who are not poor and are smart are happy. Those people who read are smart. John can read and is not poor. Happy people have exciting lives. Can anyone be found with an exciting life?**



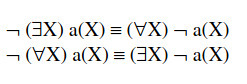


1. **Explain connection between existential and universal quantifiers**

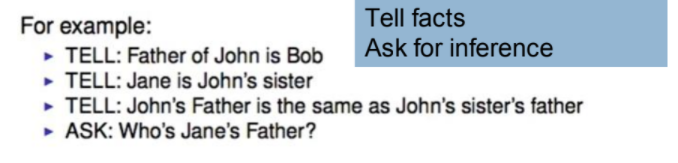




Connection between existential and universal quantifiers

****

1. **Write 5 simple knowledge based sentences**

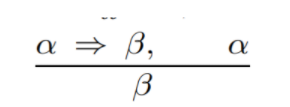
****

**16. Define Modus Ponens and AND elimination. Prove that there is no pit at [1,2]**

**Modus Ponens**

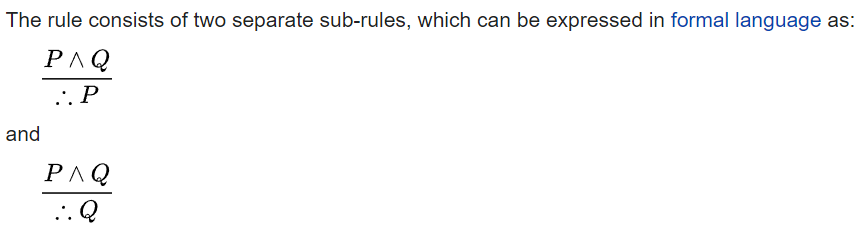
The Modus Ponens rule states that if α and α → β is true, then we can infer that β will be true.

It can be represented as:



Any sentences of the form α ⇒ β and α are given, then the sentence β can be inferred.

**AND Elimination**



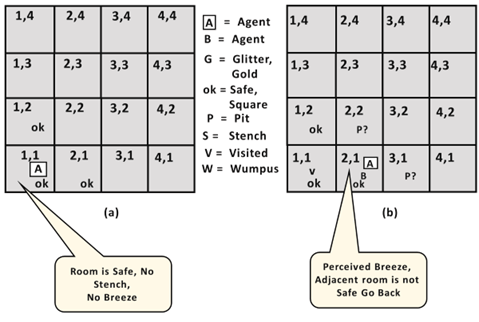
Proof of no pit example - <https://www.javatpoint.com/ai-knowledge-base-for-wumpus-world>

**17. With the help of a suitable example explain the syntax and semantics of first order logic.**

???18. 

**Agent's First step:**

Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so to represent on the below diagram (a) that room is safe we will add symbol OK. Symbol A is used to represent agent, symbol B for the breeze, G for Glitter or gold, V for the visited room, P for pits, W for Wumpus.



At Room [1,1] agent does not feel any breeze or any Stench which means the adjacent squares are also OK.

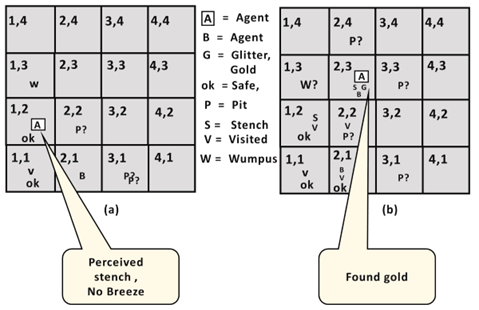
**Agent's second Step:**

Now agent needs to move forward, so it will either move to [1, 2], or [2,1]. Let's suppose agent moves to the room [2, 1], at this room agent perceives some breeze which means Pit is around this room. The pit can be in [3, 1], or [2,2], so we will add symbol P? to say that, is this Pit room?

Now agent will stop and think and will not make any harmful move. The agent will go back to the [1, 1] room. The room [1,1], and [2,1] are visited by the agent, so we will use symbol V to represent the visited squares.

**Agent's third step:**

At the third step, now agent will move to the room [1,2] which is OK. In the room [1,2] agent perceives a stench which means there must be a Wumpus nearby. But Wumpus cannot be in the room [1,1] as by rules of the game, and also not in [2,2] (Agent had not detected any stench when he was at [2,1]). Therefore agent infers that Wumpus is in the room [1,3], and in current state, there is no breeze which means in [2,2] there is no Pit and no Wumpus. So it is safe, and we will mark it OK, and the agent moves further in [2,2].



**Agent's fourth step**:

At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3]. At room [2,3] agent perceives glitter, so it should grab the gold and climb out of the cave.

19.



Figure 9.6 shows a backward-chaining algorithm for definite clauses. FOL-BC-ASK(KB,

goal) will be proved if the knowledge base contains a clause of the form lhs ⇒ goal, where

lhs (left-hand side) is a list of conjuncts. An atomic fact like American(West) is considered

as a clause whose lhs is the empty list. Now a query that contains variables might be proved

in multiple ways. For example, the query Person(x) could be proved with the substitution

{x/John} as well as with {x/Richard}. So we implement FOL-BC-ASK as a generator

a function that returns multiple times, each time giving one possible result.

Backward chaining is a kind of AND/OR search—the OR part because the goal query

can be proved by any rule in the knowledge base, and the AND part because all the conjuncts

in the lhs of a clause must be proved. FOL-BC-OR works by fetching all clauses that might

unify with the goal, standardizing the variables in the clause to be brand-new variables, and

then, if the rhs of the clause does indeed unify with the goal, proving every conjunct in the

lhs, using FOL-BC-AND. That function in turn works by proving each of the conjuncts in

turn, keeping track of the accumulated substitution as we go. Figure 9.7 is the proof tree for

deriving Criminal(West) from sentences (9.3) through (9.10).

Backward chaining, as we have written it, is clearly a depth-first search algorithm.

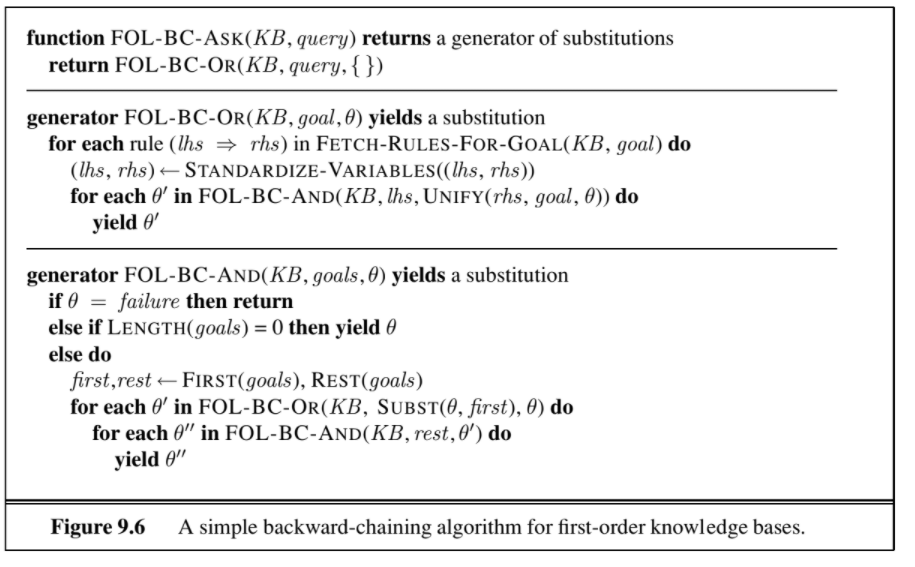
This means that its space requirements are linear in the size of the proof (neglecting, for

now, the space required to accumulate the solutions). It also means that backward chaining

(unlike forward chaining) suffers from problems with repeated states and incompleteness. We

will discuss these problems and some potential solutions, but first we show how backward

chaining is used in logic programming systems.



**20. Differentiate between first order and higher order logic.**

**21. Convert the following into Predicate Logic:**

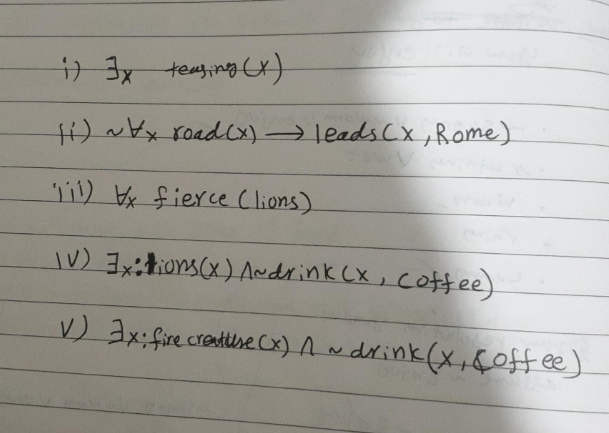
**i) Someone is teasing.**

**ii) It is not true that all roads lead to Rome.**

**iii) All lions are fierce.**

**iv) Some lions do not drink coffee.**

**v) Some fierce creatures do not drink coffee.**

****

**i)** Let P(x) : x is a person Q(x) : x is teasing

(x)[ P(x) Q(x)]

**ii)** Let S(x) : x is a road

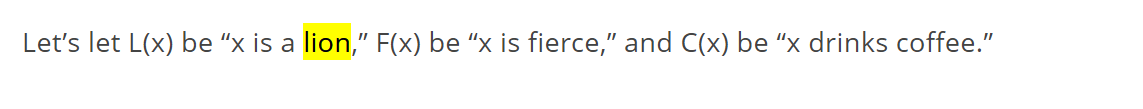
L(x) : x lead to Rome.

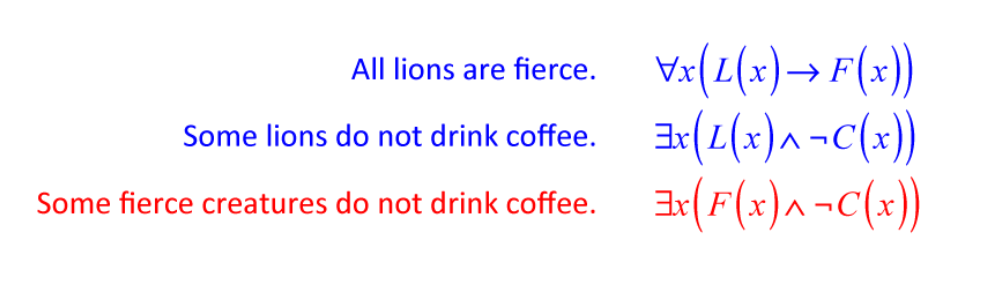
The statement can be written as2

~ (x ) [ S(x)L(x)]

or (x ) [ S(x) ~L(x)]

**Iii, iv, v)**





**22. Define the following terms as applied to predicate logic. Use suitable examples:**

**a)Atomic sentences**

**b)Well-formed Formulae**

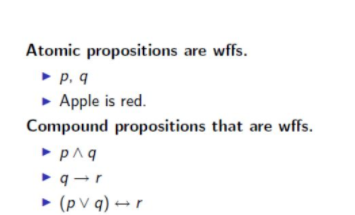
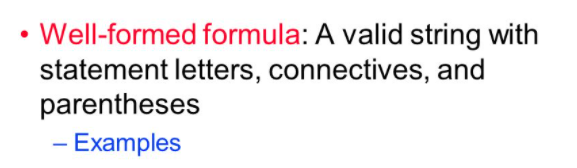
1. **Atomic sentences**

The Atomic sentences consist of a single proposition symbol.

Example: P, Q, R, W1,3 and North.

1. **Complex sentences**

Complex sentences are constructed from simpler sentences,using parentheses and logical connectives.

1. 

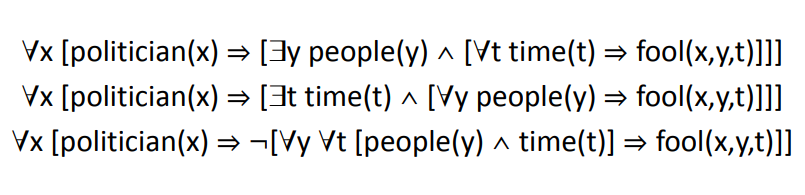
**23. Represent the following sentences in first-order logic, using a consistent vocabulary:**

**i) Politicians can fool some of the people all of the time, and they can fool all of the people some of the time, but they can’t fool all of the people all of the time.**

**ii) If a person cannot receive an idea then he cannot use that idea.**

**iii) All people living in houses that have been built in 2010 or built in 2011 have been taxed.**

i)

****

**ii) ??**

**iii) ??**

**24. Represent the following sentences in first-order logic, using a**

**consistent vocabulary:**

**i) A person born in the UK, each of whose parents are a UK citizen**

**or a UK resident, is a UK citizen by birth.**

**ii) Only one student took AI in spring 2017.**

**iii) No person buys an expensive policy.**

**iv) There is an agent who sells policies only to people who are not**

**insured.**

**v) For every number that exists, there exists a number such that**

**x<y.**

**25. Represent the following statements using First – Order Logic and**

**convert each to CNF**

**i) There are no mushrooms that are poisonous and purple**

**ii) Everyone who loves all animals is loved by someone.**

**26. Let knowledgebase( KB ) be( (P V Q)=>R; (┐P=> ┐R); Q) which**

**corresponds to the three facts we know about the new currency in**

**India:**

**i)If it is new 500 currency note or new 2000 currency note, then I**

**can shop.**

**ii) If it is old 500 currency note, then I cannot shop.**

**iii)I have a new 2000 currency note.**

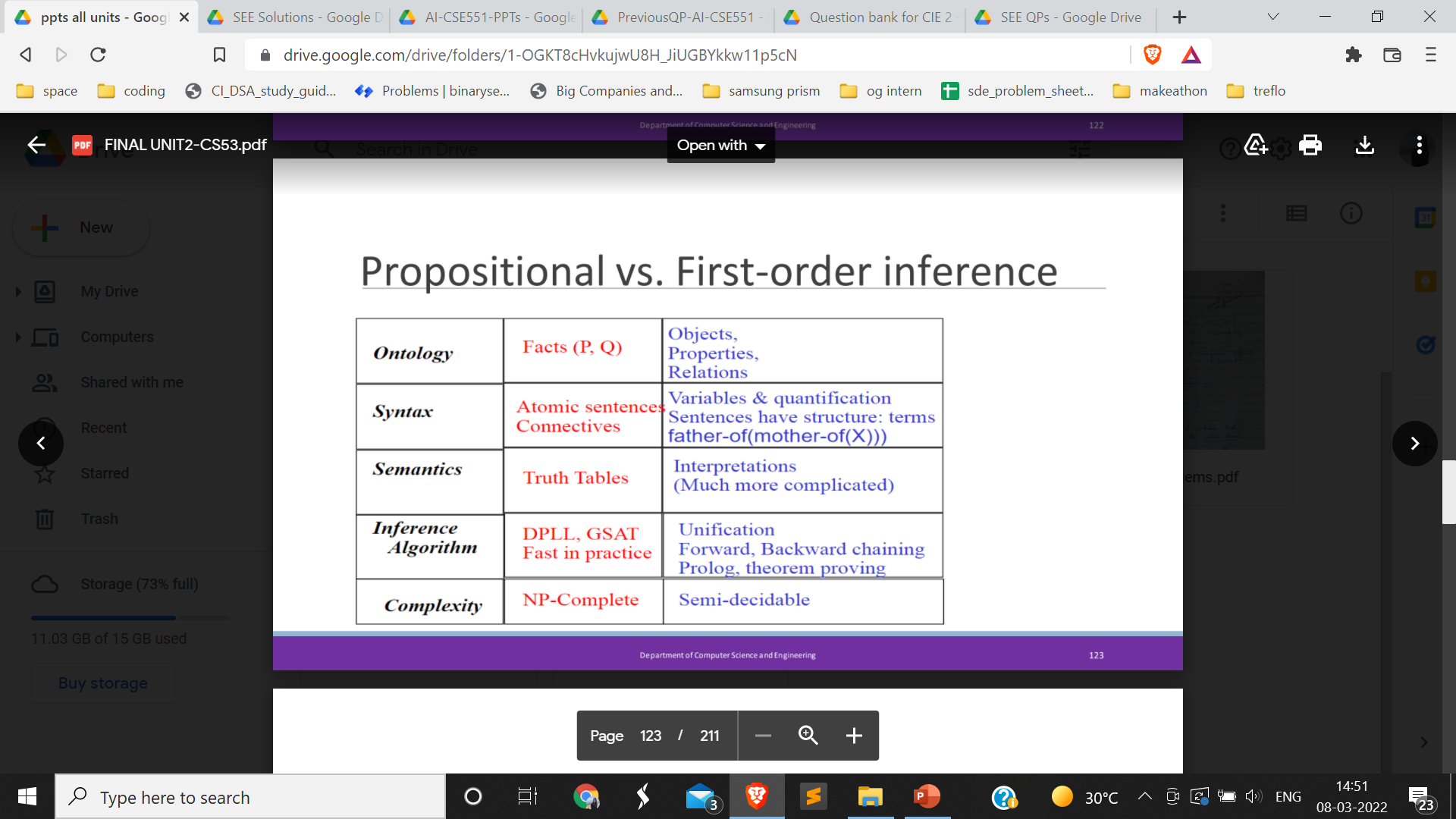
**Let the query R be “ Can I shop?” .**

**Using truth table approach show the query R is entailed by the**

**knowledgebase KB and the sentence KB=> R is valid.**

**27. Write the primary difference between propositional and first order logic.**

**Mention in tabular form the five different logics and their commitments.**

****

**5 different logics??**

**Crowd-sourced solutions :) xD**