AI LAB-2

Objectives: Study of RULES & UNIFICATION.

Description:

Rules permit Prolog to infer new facts from existing facts.

A Rule is an expression that indicates that the truth of a particular fact depends upon one or more other facts.

The rule could be expressed as the following Turbo Prolog clause:

```
Hypothesis(vitc_deficiency)if
        symptom(arthritis)and
        symptom(infection_sensitivity).
```

The conclusion is stated first and is followed by the word if. The conditions upon which the conclusion depends are stated next, each connected by the word *and*.

Every rule has a conclusion (or head) and an antecedent (or body).The antecedent consists of one or more premises. The premises in the antecedent form a conjunction of goals that must be satisfied for the conclusion to be true. If all the premises are true, the conclusion is true; if any of premises fails, the conclusion fails.

e.g: the above rule can be written as:

```
hypothesis(vitc_deficiency):-
        symptom(arthritis),
        symptom(infection_sensitivity).
```

The **:-** operator is called a *break.* A comma expresses an *and* relationship, and semicolon expresses an or relationship .However if you wish to express an *or* relationship, it is generally clearer to use two rules:

```
hypothesis(vitc_deficiency):-
        symptom(arthritis),
        symptom(infection_sensitivity).

hypothesis(vitc_deficiency):-
        symptom(colitis),
        symptom(infection_sensitivity).
```

VARIABLES IN RULES

You can use variables in rules to make a general statement about a relationship.
For e.g:

```
hypothesis(Patient,measles):-
  symptom(Patient,fever),
  symptom(Patient,cough),
  symptom(Patient,conjuntivitis),
  symptom(Patient,runny_nose),
  symptom(Patient,rash).
```

This rule states that if Patient has a fever, cough, conjunctivitis, runny-nose and rash then there is evidence that Patient has measles.

PROLOG EXECUTION RULES:

- Prolog executes using a matching process.
- When the original goal is specified, Prolog tries to find a fact or the head (conclusion) of a rule that matches this goal.
- If a fact is found, the goal succeeds immediately.
- If a rule is found, Prolog then tries to prove the head of the rule by using the antecedent (the body of premises) as a new compound goals an proving each premise of the antecedent. If any premise of the antecedent fails, Prolog backtracks and tries to solve the preceding premises with other bindings. If Prolog is not successful, it tries to find another fact or rule that matches the original goal. If all premises succeed, the original goal succeeds.
- Turbo Prolog continues to execute until all possible solutions for the goal are tested.

UNIFICATION

The process by which Prolog tries to match a term against the fact or the heads of other rules in an effort to prove a goal is called *unification*.

Basic rules for unification:

- A variable that is free will unify with any term that satisfies the preceding conditions. After unification, the variable is bound to the value of the term.
- A constant can unify with itself or any free variable. If the constant is unified with a variable, the variable will be bound to the value of the constant.
- A free variable will unify with any other free variable. After unifying, the two variables will act as one. If one of the variables becomes bound, the other will be bound to the same value.

Predicates unify with each other if

- They have the same relation name
- They have the same number of arguments.
- All arguments pairs unify with each other.

EXECUTION CONTROL.

Prolog follows these general rules:

- All clauses for the same predicate must be grouped together in the program.
- Within a specific predicate group, Prolog begins testing for a match (unification) at the first fact or rule head that matches the specified goal.
- If the head of a rule unifies with the goals, the antecedent becomes a new subgoal and must be proven next.

Exercise:

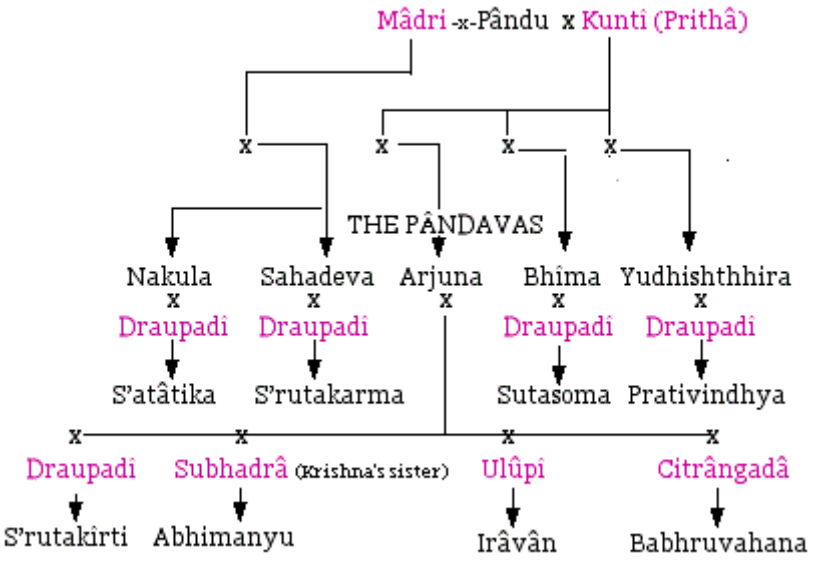
1. **Write a prolog program for the following facts and rules and answer the given question**

i. Parva has symptom fever
ii. Parva has symptom rash
iii. Parva has symptom headache
iv. Parva has symptom runny nose
v. Vidhi has symptom chills
vi. Vidhi has symptom fever
vii. Vidhi has symptom headache
viii. Vivan has symptom runny nose
ix. Vivan has symptom rash
x. Vivan has symptom flu

Rule 1: Patient has Disease measles if Patient has symptoms fever, cough, conjunctivitis and rash.
Rule 2: Patient has Disease german measles if Patient has symptoms fever, headache, runny nose and rash.
Rule 3: Patient has Disease flu if Patient has symptoms fever, headache, body-ache and chills.
Rule 4: Patient has Disease common cold if Patient has symptoms headache, sneezing, sore throat, chills and runny nose.
Rule 5: Patient has Disease mumps if Patient has symptoms fever and swollen glands.
Rules 6: Patient has Disease chicken pox if Patient has symptoms fever, rash, body-ache and chills.

Question: Identify patient with any particular disease based on rules and facts given above.

2. **Write a program for family tree either a given below or by creating your own which contains three predicates: male, female, parent.**
   Make rules for family relations: father, mother, grandfather, grandmother, brother, sister, uncle, aunt, nephew and niece.



3. **Write a prolog program for the following facts and rules, and trace the given goals:**
   i. hardware is easy course
   ii. Books for hardware are available
   iii. logic is not easy course
   iv. graphics is easy course
   v. graphics has 8 credits
   vi. graphics has lab component
   vii. Books for database are available
   viii. Mary takes compilers

Rule1: X takes Y, if Y is easy course and books for Y are available
Rule2: X takes Y, if Y has 8 credits and Y has lab component

Goals: a) Does Mary take graphics course?
b) Which course Mary takes?