EXNO: 13

DATE:

UNIFICATION AND RESOLUTION

AIM:

To execute programs based on unification and Resolution. Deduction in Prolog is based on the unification and Instantiation. Matching terms are unified and variables get Instantiated

Example 1: In the below Prolog program, unification and Instantiation take place after Querying Facts:

likes (John, Jane)
likes (Jane, John)

Query:-

?- likes (John, X)
Answer : X = Jane

Here upon asking the Query first Prolog in start to search matching terms in predicate with two arguements and it can match like (John. . . .). ie unification Then it look for value of X in Query and returns answer X = Jane.

Example 2: At the Prolog query prompt, when you below Query ?- owns (X, car (bmw)) = owns (Y, car (c)).

Here owns (X, car (bmw)) and owns (Y, car (c)) unifies - because

i] Predicate names are same on both side
ii] Number of arguements for that Predicate equal both side.
iii] 2nd arguement with Predicate inside the brackets are same both side and even in that Predicate again number of arguements are same.

But when you write ?- owns(X, cat(Bmw)) = likes(Y, cat(C)), then Prolog will return false, since it can not match owns: and likes:

Resolution is one kind of Proof technique works that way -

i) select two clauses that contain caffbict terms.
ii) combine those two clauses and
iii) cancel out the conflicting terms.

For example we have following statement,

(1) If it is a Pleasant day you will do Strawberry Picking.
(2) If you are doing Strawberry Picking you are happy

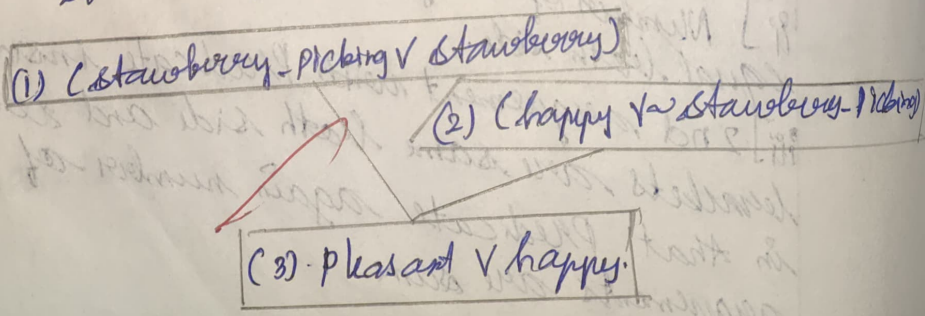
Above statements can be written Propositional logic like this -

(1) Strawberry_Picking ← Pleasant
(2) happy ← Strawberry_Picking

And again these statements can be CNS this -

(1) (Strawberry_Picking ∨ ~Pleasant) ∧
(2) (happy ∨ ~Strawberry_Picking) ∧
(3) ~Pleasant ∨ happy

How? see the figure on right.
When we write about two clause in infer an implies form, we have
If it is pleasant day you are happy.



Let's see an example to understand how Resolution and Refutation work. In below example part (I) represents the English meaning for clauses Part (II) represents the Propositional Logic, Part (III) represents the CNF of Part (II) and Part (IV) want to prove using Refutation Proof method.

Part (I): English Sentences.

1) If it is sunny and warm day you will enjoy.
2) If it is warm and pleasant day you will do strawberry picking.
3) If it is raining then no strawberry picking.
4) If it is raining you will get wet
5) It is warm day
6) It is raining.
7) It is sunny.

Part(II): propotional Logic

1) enjoy ← Sunny ∧ warm
2) strawberry-picking ← warm ∧ pleasant
3) ~ strawberry-picking ← raining.
4) wet ← raining
5) warm
6) raining
7) sunny.

Part(III): CNF

1) (enjoy ∨ ~ Sunny ∨ warm) ∧
2) (strawberry - picking ∨ ~ warm ∨ - pleasant) ∧
3) (~ strawberry - picking ∨ ~ raining) ∧
4) (wet ∨ ~ raining) ∧
5) (warm) ∧
6) (raining) ∧
7) (sunny).

Part (iv) : Other statements we want to prove by refutation

(Goal 1) You are not doing strawberry picking

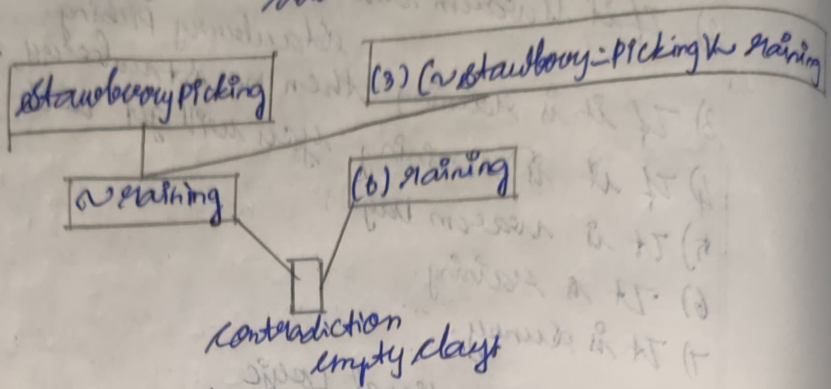
(Goal 2) You will enjoy

(Goal 3) Try it yourself : You will get wet.

Goal 1 : you are not doing strawberry picking

Prove : ~strawberry picking

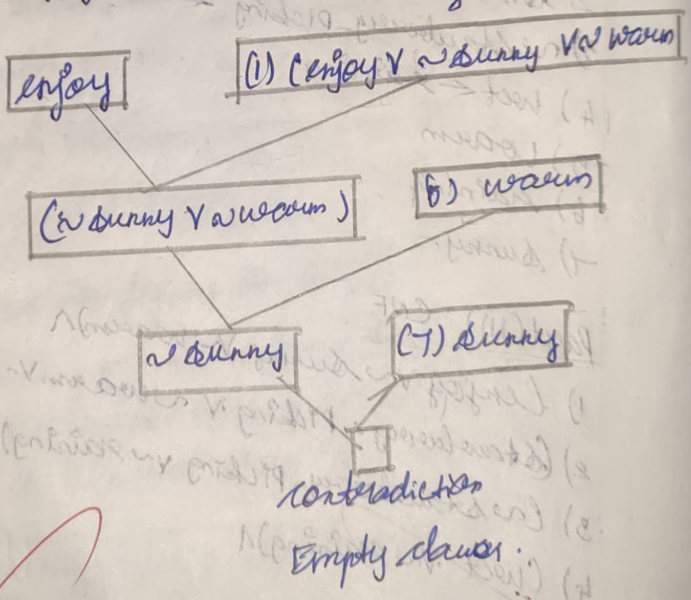
Assume : strawberry_picking (negate the goal and add it to given clauses).



Goal 2 : You will enjoy

Prove : enjoy

Assume : ~ enjoy (negate the goal and add it to given clause)



SOURCE CODE

```
enjoy :- sunny, warm
strawberry_picking :- warm, pleasant.
not strawberry_picking :- raining
wet :- raining
warm
raining
sunny
```

OUTPUT

```
?- not strawberry_picking
true
?- enjoy
true
?- wet
true.
```

RESULT:

Thus unification and Resolution is implemented and executed successfully.