Artificial Intelligence Lab Report
Lab3: FIRST ORDER PREDICATE LOGIC

Submitted By: SUYOG DHAKAL (075BCT092)

Theory:

Propositional Logic

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

Example

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)
4. d) 5 is a prime number.

First Order Predicate Logic

First Order Predicate Logic(FOPL) or simply predicate logic can be used to express wide range of statements in ways that permit us to reason and explore relationships between objects. For example, consider a statement "X is a man." which has two parts; first the variable X, is the subject of the statement and the second part "is a man" is called predicate which represents the property that the subject of the statement can have. It may be denoted as man(X). Once a variable has been assigned to the propositional function man(X), it becomes propositional logic and has a associated truth value.

Some examples of first order predicate logic used to represent natural language statements are:

Ram loves all animals.

- $\forall x Animals(x) \Rightarrow Loves(ram, x)$

Poppy is a dog.

- Dog (Poppy)

Grandparent is a parent of one's parent

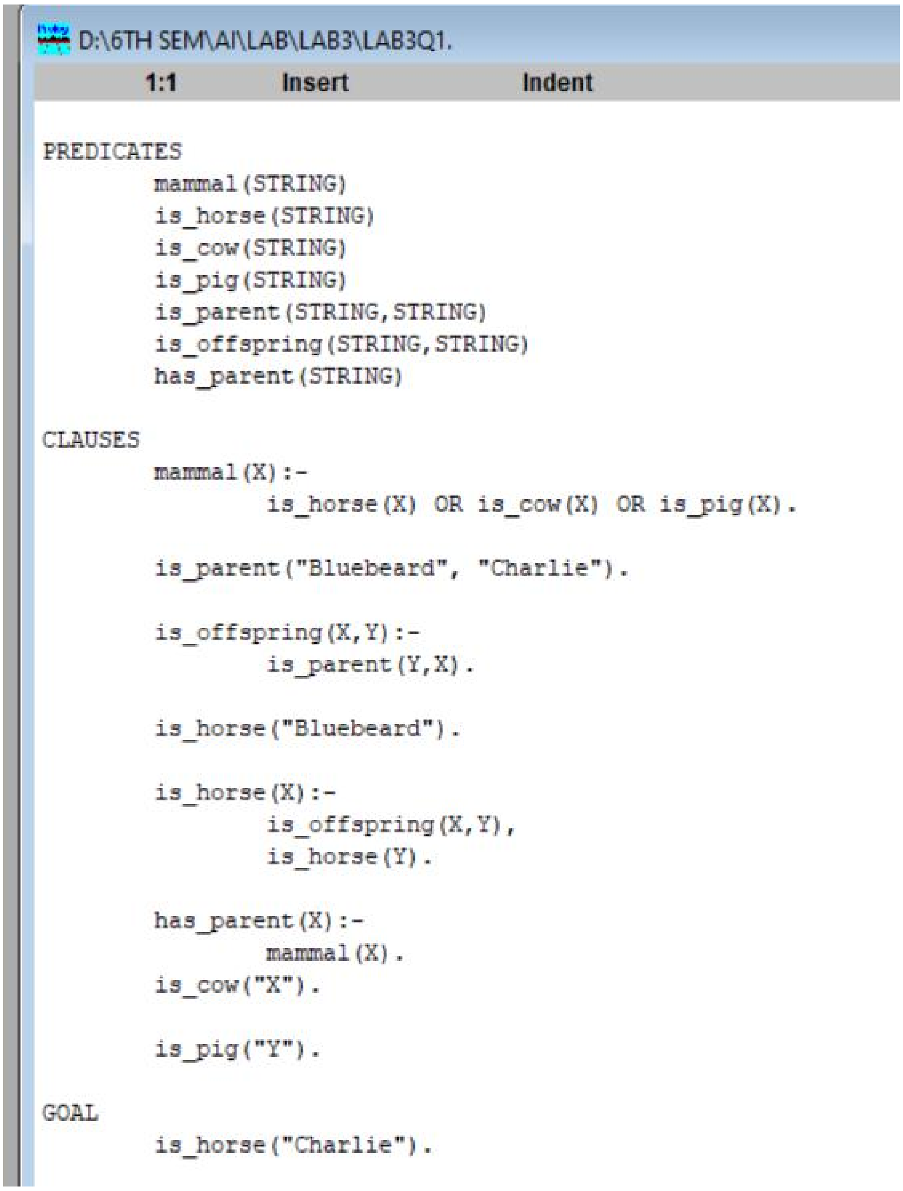
- $\forall x, y Grandparent(x, y) \Leftrightarrow \exists z Parent(x, z) \cap Parent(z, y)$

Parent and child are inverse relation.

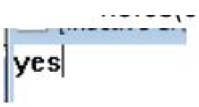
- $\forall x, y Parent(x, y) \Leftrightarrow Child(y, x)$

Assignments:

1.

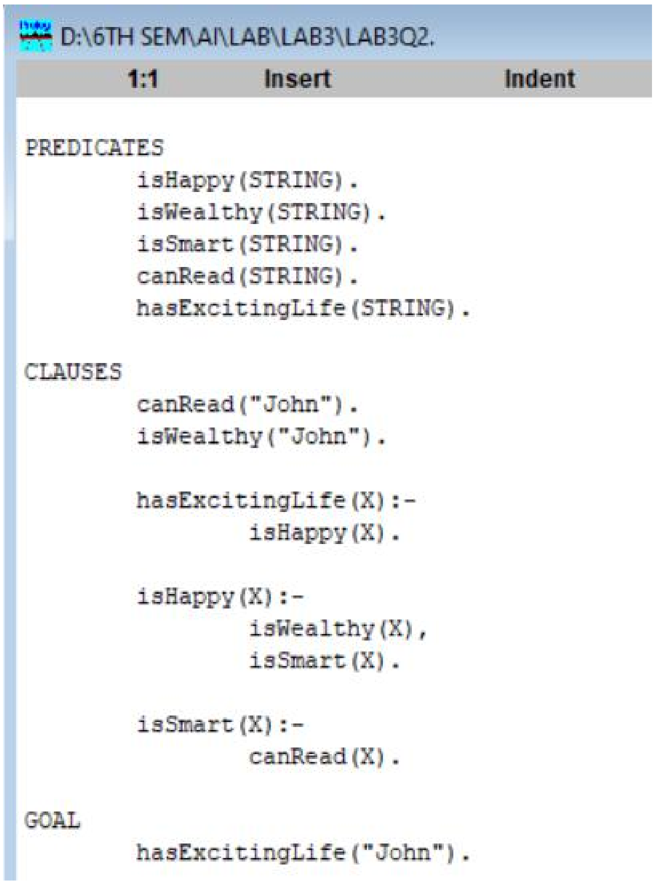


```
PREDICATES
        mammal(STRING)
        is_horse(STRING)
        is_cow(STRING)
        is_pig(STRING)
        is_parent(STRING,STRING)
        is_offspring(STRING,STRING)
        has_parent(STRING)

CLAUSES
        mammal(X):-
                is_horse(X) OR is_cow(X) OR is_pig(X).

        is_parent("Bluebeard", "Charlie").

        is_offspring(X,Y):-
                is_parent(Y,X).

        is_horse("Bluebeard").

        is_horse(X):-
                is_offspring(X,Y),
                is_horse(Y).

        has_parent(X):-
                mammal(X).
        is_cow("X").

        is_pig("Y").

GOAL
        is_horse("Charlie").
```

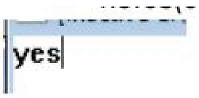


```
yes
```

2.

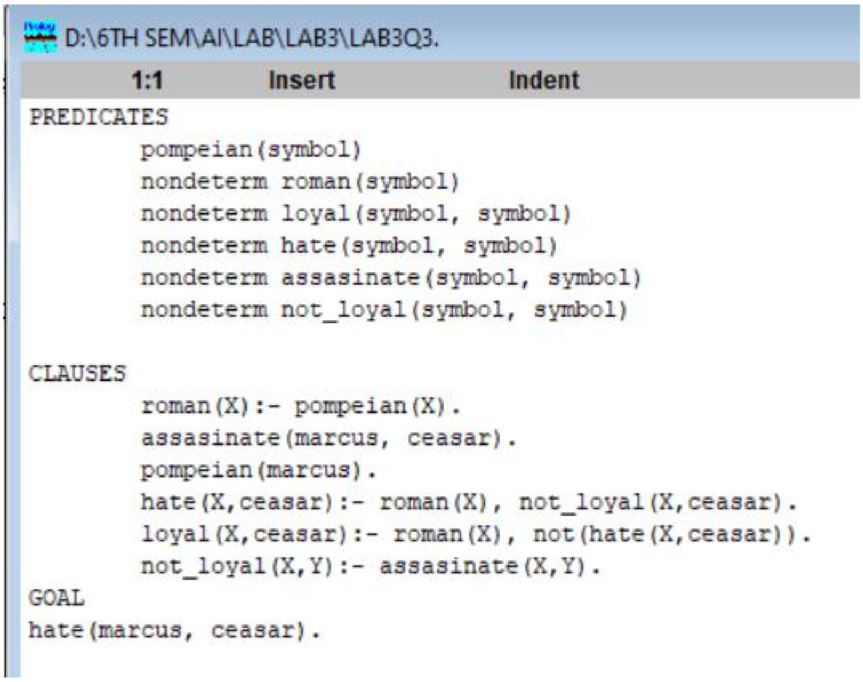


```
PREDICATES
        isHappy(STRING).
        isWealthy(STRING).
        isSmart(STRING).
        canRead(STRING).
        hasExcitingLife(STRING).

CLAUSES
        canRead("John").
        isWealthy("John").

        hasExcitingLife(X):-
                isHappy(X).

        isHappy(X):-
                isWealthy(X),
                isSmart(X).

        isSmart(X):-
                canRead(X).

GOAL
        hasExcitingLife("John").
```

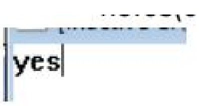


```
yes
```

3.

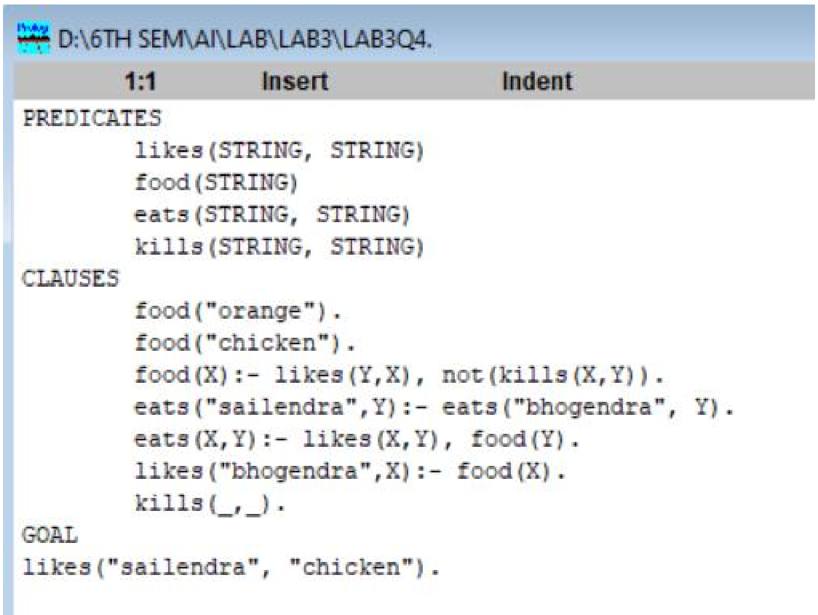


```
PREDICATES
        pompeian(symbol)
        nondeterm roman(symbol)
        nondeterm loyal(symbol, symbol)
        nondeterm hate(symbol, symbol)
        nondeterm assasinate(symbol, symbol)
        nondeterm not_loyal(symbol, symbol)

CLAUSES
        roman(X):- pompeian(X).
        assasinate(marcus, ceasar).
        pompeian(marcus).
        hate(X,ceasar):- roman(X), not_loyal(X,ceasar).
        loyal(X,ceasar):- roman(X), not(hate(X,ceasar)).
        not_loyal(X,Y):- assasinate(X,Y).
GOAL
hate(marcus, ceasar).
```

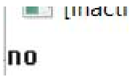


```
yes
```

4.

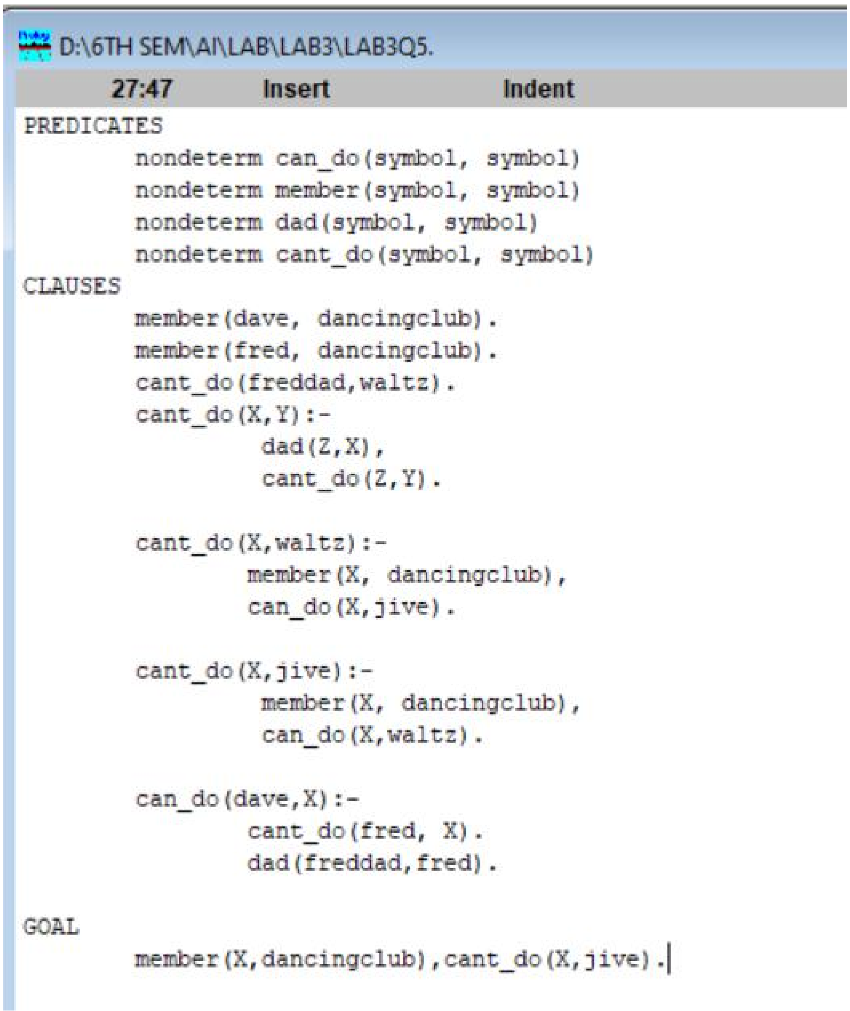


```
PREDICATES
        likes(STRING, STRING)
        food(STRING)
        eats(STRING, STRING)
        kills(STRING, STRING)
CLAUSES
        food("orange").
        food("chicken").
        food(X):- likes(Y,X), not(kills(X,Y)).
        eats("sailendra",Y):- eats("bhogendra", Y).
        eats(X,Y):- likes(X,Y), food(Y).
        likes("bhogendra",X):- food(X).
        kills(_,_).
GOAL
likes("sailendra", "chicken").
```

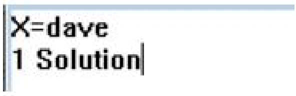


```
no
```

5.



```
PREDICATES
        nondeterm can_do(symbol, symbol)
        nondeterm member(symbol, symbol)
        nondeterm dad(symbol, symbol)
        nondeterm cant_do(symbol, symbol)
CLAUSES
        member(dave, dancingclub).
        member(fred, dancingclub).
        cant_do(freddad,waltz).
        cant_do(X,Y):-
                 dad(Z,X),
                 cant_do(Z,Y).

        cant_do(X,waltz):-
                member(X, dancingclub),
                can_do(X,jive).

        cant_do(X,jive):-
                 member(X, dancingclub),
                 can_do(X,waltz).

        can_do(dave,X):-
                cant_do(fred, X).
                dad(freddad,fred).

GOAL
        member(X,dancingclub),cant_do(X,jive).
```



```
X=dave
1 Solution
```

Discussion

In Lab3 we learned to solve first order predicate logic problems. In the beginning we learned how to analyze and assign predicates and clauses.while solving the logical negation it was problematic.

Conclusion

Hence, we wrote program and output was analyzed and report was made accordingly.