COMP348 PRINCIPLES OF PROGRAMMING LANGUAGES

TUTORIAL – 1

PROLOG – PART I Clauses, Facts, and Rules

Acknowledgement

? The following slides are reproduced from the tutorial slides of the course COMP 348 by Dr. Mohamed Taleb.

Prolog Structure

Clauses

- Facts: An expression that makes a declarative statement.
- Rules: An expression that uses logical implications to describe relationships between facts.

Queries

- Grounded Queries: Responds with a boolean value (true or false) based on whether it evaluates to true or not.
- Non-Grounded Queries: Used to search through and retrieve some data.

Clauses

Exercise #1

Some synopsis of the database we are going to use in prolog.

- Database:

```
object(galatea).
object(larissa).
object(thalassa).
mass(mercury, 0.33).
mass(venus, 4.87).
mass(earth, 5.98).
orbits(mercury, sun).
orbits(venus, sun).
orbits(earth, sun).
```

Exercise #1(CONT.)

- Suppose we let object stand for the isObject relation.
  Let orbits stand for the orbits relation, and
  let p stand for isPlanet relation.

Question

Define a formula called P, to say that if o is an object with mass equal to or greater than 0.3 and o orbits around the sun, then we conclude that o is a planet. Use mass(o) to represent the mass of object.

Solution

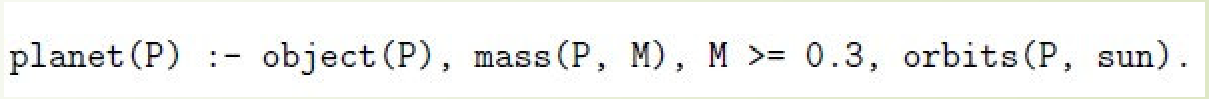
$$P = \forall o \; (obj(o) \; \wedge \; (mass(o) \; >= \; 0.3) \; \wedge \; orb(o, sun)) \; \rightarrow \; p(o))$$

Exercise #1(CONT.)

Question

Define a Prolog rule planet(P) for the isPlanet relation.

Solution



```
planet(P) :- object(P), mass(P, M), M >= 0.3, orbits(P, sun).
```

- ❑ Consider the following query and its result:
  ?- planet(X).

Result

```
X = mercury ;
X = venus ;
X = earth ;
X = mars ;
X = jupiter ;
X = saturn ;
X = uranus ;
X = neptune ;
```

Exercise #1(CONT.)

- Let s stand for isSatellite relation. Define a formula (call is S), to say that if an object o orbits around a planet, then we conclude that o is a satellite.

$$S = \forall o \forall x \ (obj(o) \ \wedge \ orb(o,x) \ \wedge \ p(x)) \ \rightarrow \ s(o))$$

- Define a Prolog rule satellite(S) for the isSatellite relation.

```
satellite(S) :- object(S), orbits(S, P), planet(P).
```

- Consider the following query and its result:
  ?- satellite(X).

Exercise #1(CONT.)

- ❑ Phobos is an object in our solar system. Is Phobos a satellite?
  1. Translate this question into a query.
  2. What type of query is it?

Solution

1. The query is:
satellite(photos).

2. The type of the query is :
Ground Query

Exercise #1(CONT.)

Question:

Demonstrate step-by-step how the above query proceeds until indicating success or failure. You must explain this only in terms of unification, resolution, substitution and instantiation.

Explanation

1.Prolog will search the database from top to bottom trying to find a clause that can be matched with the query.

2.The query satellite(phobos) will unify with the rule satellite(S) rule, instantiating S to phobos. Resolution will apply the substitution of the variables and produce a new rule:

```
satellite(phobos) :- object(phobos), orbits(phobos, P), planet(P).
```

3.All three goals in the body of the rule have to be satisfied for the head of the rule to be satisfied.

Exercise #1(CONT.)

Explanation(Cont..)

a) The first goal is unified with the fact object(phobos).

b) The second goal is unified with the fact orbits(phobos, mars). Instantiating P to mars.

c) Prolog will now try to satisfy the third goal. It will unify planet(mars) with the rule planet(P) instantiating P to mars. Resolution will apply the substitution of the variables and produce a new rule:

```
planet(mars) :- object(mars),
mass(mars, M),
M >=0.3,
orbits(mars, sun).
```

4. The first and fourth goals are unified with the facts object(mars), and orbits(mars, sun) respectively. The second goal unifies with the fact mass(mars, 0.64) instantiating M to 0.64. The third goal will be evaluated and succeed. As a result the original query succeeds.

Exercise #2

- Database:

```
lectures(turing, 9020).
lectures(codd, 9311).
lectures(backus, 9021).
lectures(ritchie, 9201).
lectures(minsky, 9414).
lectures(codd, 9314).

studies(fred, 9020).
studies(jack, 9311).
studies(jill, 9314).
studies(jill, 9414).
studies(henry, 9414).
studies(henry, 9314).

%year(X, Y): person X is in year Y
year(fred, 1).
year(jack, 2).
year(jill, 2).
year(henry, 4).
teaches(Teacher, Student) :- lectures(Teacher, Course), studies(Student, Course).
```

Exercise #2(CONT.)

- ❑ If turing lectures in course 9020?

  Solution

  ```
  ?- lectures(turing, 9020).
  ```

- ❑ Which course(s) Prof. codd teaches?

  Solution

  ```
  ?- lectures(codd, Course).
  ```

- ❑ Does turing teach fred?

  Solution

  ```
  ?- lectures(turing, Course), studies(fred, Course).
  ```

Exercise #2(CONT.)

- Who does codd teach?

  Solution

  ```
  ?- lectures(codd, Course), studies(Student, Course).
  ```

- more_advanced(S1, S2) :- year(S1, Year1), year(S2, Year2), Year1 > Year2.

  Question

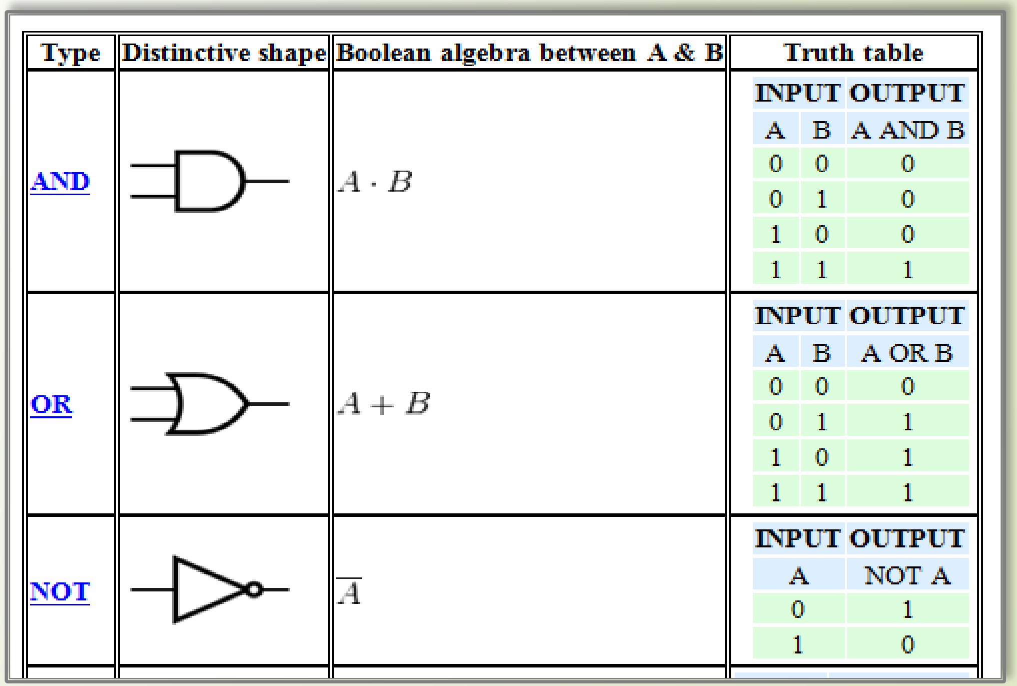
  ```
  ?- more_advanced(henry, fred).
  ```

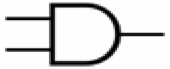
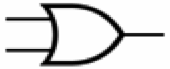
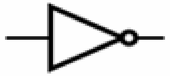
  Solution

  ```
  true
  ```

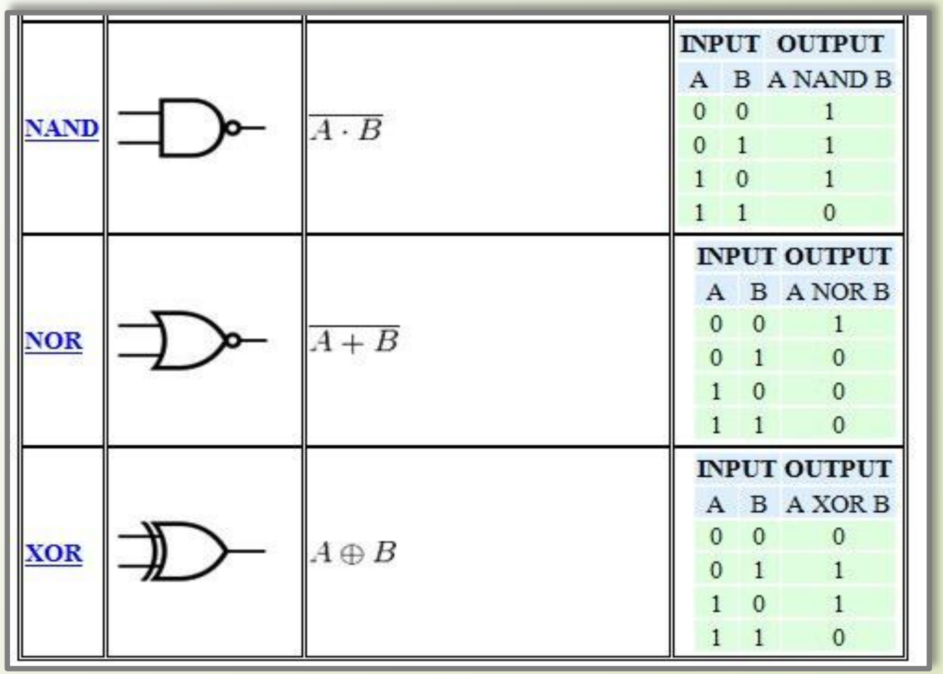
Boolean

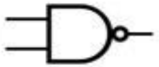
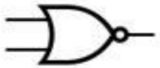
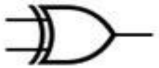
Logic Gates



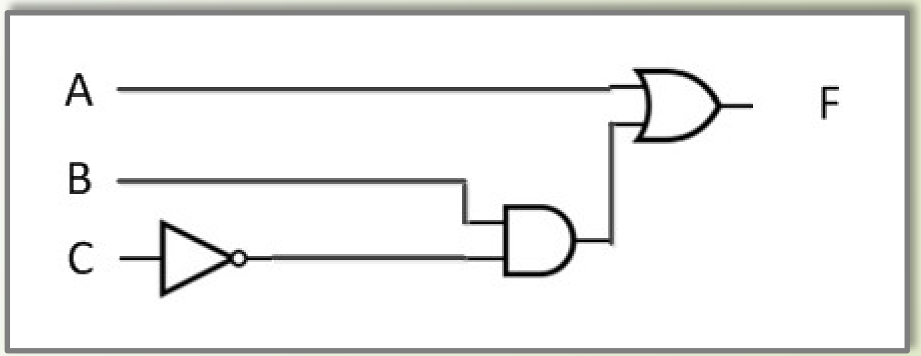
Type	Distinctive shape	Boolean algebra between A & B	Truth table
AND		$A \cdot B$	INPUT OUTPUT A B A AND B 0 0 0 0 1 0 1 0 0 1 1 1
OR		$A + B$	INPUT OUTPUT A B A OR B 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$\overline{A}$	INPUT OUTPUT A NOT A 0 1 1 0

Logic Gates (Cont.)



			INPUT		OUTPUT
			A	B	A NAND B
NAND		$\overline{A \cdot B}$	0	0	1
			0	1	1
			1	0	1
			1	1	0
			INPUT		OUTPUT
			A	B	A NOR B
NOR		$\overline{A + B}$	0	0	1
			0	1	0
			1	0	0
			1	1	0
			INPUT		OUTPUT
			A	B	A XOR B
XOR		$A \oplus B$	0	0	0
			0	1	1
			1	0	1
			1	1	0

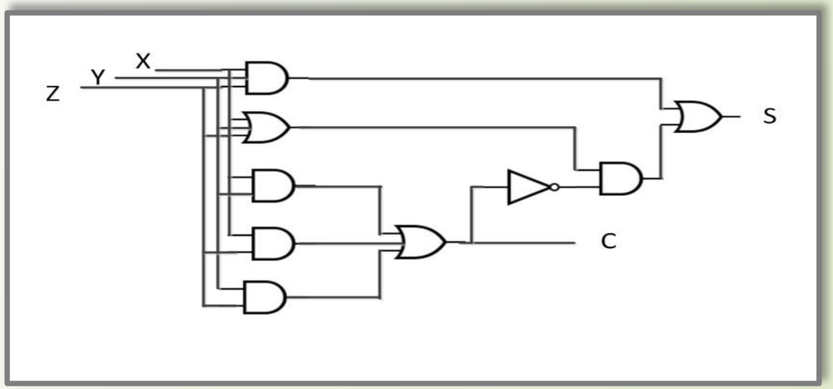
Exercise #1



Solution

```
circuit(A, B, C, Out) :- inv(C, Cinv),and(B, Cinv, BC),
or(BC, A, Out).
```

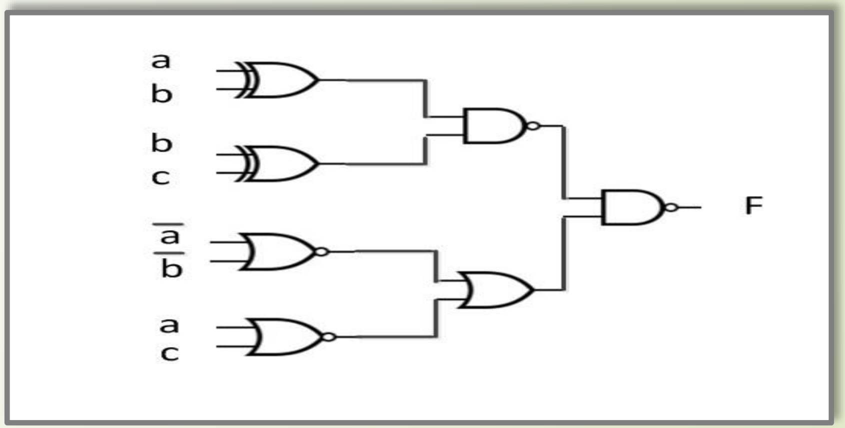
Exercise #2



Solution

```
sum(X, Y, Z, S, C) :- and(X, Y, XY),and(XY, Z, XYZ),or(X, Y, XoY),
or(XoY, Z, XoYoZ),and(X, Z, XZ),and(Y, Z, YZ),
or(XY, XZ, Temp1),or(Temp1, YZ, C),inv(C, CI),
and(XoYoZ, CI, Temp2),or(Temp2, XYZ, S).
```

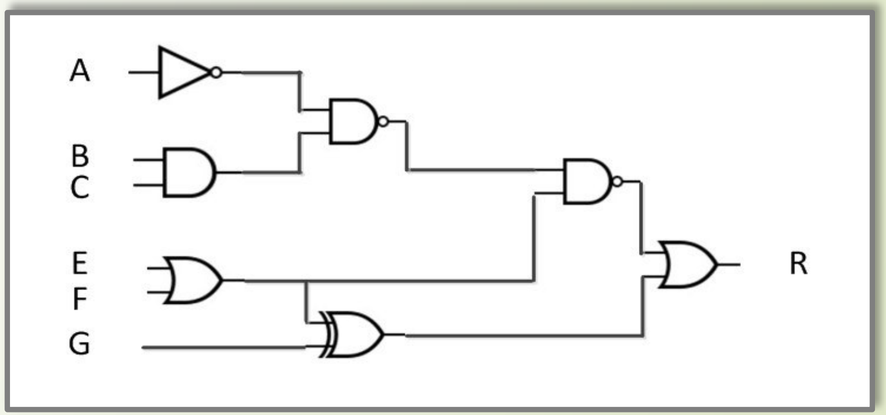
Exercise #3



Solution

result(A,B,C,F) :- xor(A, B, X1),xor(B, C, X2),nand(X1, X2, P1),
inv(A, A1),inv(B, B1),nor(A1, B1, N1),nor(A, C, N2),
or(N1, N2, P2),nand(P1, P2, F).

Exercise #4



Solution

```
final(A, B, C, E, F, G, R) :- inv(A, A1),and(B, C, Cnt1),
nand(A1, Cnt1, Cnt2),or(E, F, Cnt3),nand(Cnt2, Cnt3, T1),
xor(Cnt3, G, T2),or(T1, T2, R).
```