4.5 Logic programming and PROLOG

A key difference between logic and the PROLOG is shown below:

logic	**prolog**
Variables are explicitly quantified .	Quantification is provided implicitly by the way variables are interpreted (variables begin with upper letters, constants with lower letters)
There are explicit symbols for *and* ($\wedge$) & *or* ($\vee$)	There is an explicit symbol for *and* (,) but there is none for *or*.
Negation ($\neg$) represented explicitly. $\forall x$: dog(x) $\rightarrow \neg$ cat(x)	Negation represented implicitly by the lack of assertion (**negation as failure**).
Example: $\forall x: \exists y: Q(x, y) \rightarrow P(X)$	Example: p(X):- q(X, Y).

Clauses (Facts and Rules)

Facts (relations or properties)

Natural language	**Prolog syntax**
Ahmed likes Saly	likes(ahmed, saly)

Natural	**Prolog**
Kermit is	green(kermit)
Hiam is a girl	girl (hiam)

Rules

```
  Saly likes everything that Ayman likes.
  likes(saly, Something):- likes(ayman, Something).
```

The `:-` symbol is simply pronounced "if", and serves to separate the two parts of a rule: the head and the body.

Here is a program designed to find solutions to this car-buying problem:

```
PREDICATES
  nondeterm can_buy(symbol, symbol)
  nondeterm person(symbol)
  nondeterm car(symbol)
  likes(symbol, symbol)
  for_sale(symbol)
CLAUSES
  can_buy(X,Y):-
  person(X),
  car(Y),
  likes(X,Y),
  for_sale(Y).
  person(mohamad).
  person(saher).
  person(samia).
  person(osama).
  car(lemon).
  car(hot_rod).
  likes(kelly, hot_rod).
  likes(judy, pizza).
  likes(ellen, tennis).
  likes(mark, tennis).
  for_sale(pizza).
  for_sale(lemon).
   for_sale(hot_rod).
```

What can Saher and Mohmad buy? Who can buy the hot rod? You can try the following goals:

```
    can_buy(Who, What).
    can_buy(saher, What).
    can_buy(mohamad, What).
    can_buy(Who, hot_rod).
```

Queries

Once we give Prolog a set of facts, we can proceed to ask questions concerning these facts; this is known as *querying the Prolog system*. We can ask Prolog the same type of questions that we would ask you about these relations. Based upon the known facts and rules given earlier, you can answer questions about these relations, just as Prolog can.

In natural language, we ask you:

```
  Does Ayman like Saly?
```

In Prolog syntax, we ask Prolog:

```
  likes(ayman, saly).
```

Given this query, Prolog would answer

```
  yes
```

because Prolog has a fact that says so. As a little more complicated and general question, we could ask you in natural language:

```
  What does ayman like?
  likes(ayman, What).
```

Notice that Prolog syntax does not change when you ask a question: this query looks very similar to a fact. However, it is important to notice that the second object--*What*--begins with a capital letter, while the first object--*ayman*--does not. This is because *ayman* is a fixed, constant object--aconstants known value--but *What* is a variable. Variables always begin with an upper-case letter or an underscore.

Prolog always looks for an answer to a query by starting at the top of the facts. It looks at each fact until it reaches the bottom, where there are no more. Given the query about what Ayman likes, Prolog will return

```
  What=saly
  What=cats
  2 Solutions
```

This is because Prolog knows

```
  likes(ayman, cats).
```

and

```
  likes(ayman, cats).
```