Introduction to logic...

- A form of logic...
- Study on reasoning process and production system (also rule) that support the reasoning process.

The idea of Prolog

- A programming language.
- “PROgramming in LOGic” → PROLOG
- Very versatile language – can do or implement all kinds of algorithms.

The idea of Prolog

- Prolog program consist of
  1) a set of facts
  2) a set of conditions

The computer can figure out for itself how to deduce the solution from the facts given.
The idea of Prolog

- Invented by Alain Colmerauer and his colleagues at the University of Aix-Marseille, France in 1972.
- Powerful language for AI and non-numerical programming in general.
- Commercially used in expert systems, intelligent databases, and natural language processing programs.

Varieties of Prolog

- Many Prolog versions are available.
- Standard applied are the same – different in syntax, built-in function, and operating system compatibility.
- Example:
  - Arity Prolog
  - Quintus Prolog
  - Amzi Prolog
  - ALS Prolog
  - SWI Prolog
  - LPA Prolog

How it works?

- Process - Procedural Interpretation of Logic.
  - Knowledge is represented in terms of procedure definitions – clauses.
  - Reasoning – a process of calling the right procedures.

Prolog vs Lisp

- both easy to perform complex computations on complex data.
- both allocate memory dynamically – programmer does not have to declare the size of data structures before creating them.
- both can examine and modify itself.

How it works?

- Example:
  1. For any X, if X is in Kedah, then X is in the Malaysia
  2. Alor Star is in Kedah

Note:

- [1] is a RULE – enables us to infer other info.
- [2] is a fact – does not depend on other

- Both are two types of clauses
How it works?

To know whether Alor Star is in Malaysia – [1] and [2] can be chained together.

In Prolog:

\[
\text{in\_Malaysia}(X) :- \text{in\_Kedah}(X). \\
\text{in\_Kedah}(\text{alor\_star}).
\]

Note:
- \text{in\_Malaysia} and \text{in\_Kedah} are PREDICATES

Declarative and Procedural Meaning

- **Declarative meaning**
  - concerned only with the relations defined in the program.
  - determines what will be the output of the program.

- **Procedural meaning**
  - How the output is obtained – how the relations are actually evaluated by Prolog.

**Terminology**

- **Fact**
- **Rule**
- **Predicate**
- **Argument**
- **Arity**

Consist of predicate with or without argument.

Example:

- \text{state(kedah)}, \text{wan_hussain}.

Consists of name, bracket, and arguments

Example:

- \text{country(malaysia)}.

Element in predicate (written in bracket)

Example:

- For \text{country(malaysia)} the argument is ‘malaysia’.
Terminology
- Fact
- Rule
- Predicate
- Argument
- Arity

Referring to the number of arguments in predicate.
Example:
- state(malaysia).
- father(ahmad, karim).
- foo.

Syntax - Atoms
- Used as names of individuals and predicates.
- Begins with a lowercase letter
- Can contains letters, digits, and the underscore mark (_)
Example:
- kedah
- fatihah1982
- muhamad_shahrul_aiman_rashid
- 'Malaysia'
- '17/6/98'

Syntax - Numbers
- Comprises of integer and real number.
Example:
<table>
<thead>
<tr>
<th>Integer</th>
<th>Real number</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1.234</td>
</tr>
<tr>
<td>12</td>
<td>0.124</td>
</tr>
<tr>
<td>0</td>
<td>0.000009</td>
</tr>
<tr>
<td>100</td>
<td></td>
</tr>
<tr>
<td>789</td>
<td></td>
</tr>
</tbody>
</table>

Syntax - Structure
- Comprises of several components – atom, bracket (open and close), and argument (inside the bracket).
Example:
- on_top(book, table).
- country(malaysia)
- human.
- State(kedah).
- netbook(brand(acer), price(1500)).
Syntax - Structure

- The atom at the beginning is called the **FUNCTOR** of the structure.
- If some of the arguments are also structures, the functor at the beginning of the whole thing is called the **PRINCIPAL FUNCTOR**.

```prolog
netbook(acer, 1500).
netbook(brand(acer), price(1500)).
```

Exercise

- What are these and is it legal?
  - `country(indonesia).`
  - `Book(software_engineering) on_table 123574 nom eat(mahadi, rice, chicken).`
  - `Country('Malaysia'). 5,000.04 78854`

Syntax - Variable

- Contains a string of letters, digit, and underscore.
- Begin with capital letters or the underscore mark.
- Example:
  ```prolog
  A Var1
  Student_Name _name
  _College id_17638
  ```

Exercise

- What is wrong with this one?
  ```prolog
  capital of (alor star, kedah).
  ```

Syntax - Variable

- A special variable - anonymous variable "underscore (_) character".
- Don't care how this variable is instantiated - don't care which term it's bound to, as long as it's bound to something.
- Example:
  ```prolog
  ?- is_in(X, kedah).
  is_in(jitra, kedah).
  is_in(kuantan, pahang).
  is_in(muar, johor).
  ```

Pattern Matching

- `2 = A NUMBER 1 = A NUMBER 2 = 1`
Pattern Matching

- Matching is a process that takes as input two terms and checks whether they match.
- The matching operator is “=”.
- Example:

  \[
  a = a, \\
  \text{state(kedah)} = \text{state(kedah)}. 
  \]

Pattern Matching

- IDENTICAL predicates - the predicate properties must be the same, i.e:
  - The predicate name,
  - Number of argument/arity
  - The sequence/order of arguments in the predicate

Pattern Matching

- Two objects/terms are match if:
  - they are IDENTICAL, or
  - the variables in both terms can be INSTANTIATED to objects.

Pattern Matching

- IDENTICAL object – objects are the same.
Pattern Matching

- In Prolog matching process is also called **UNIFICATION**.
- Unification between:
  - Query with the fact
  - Query with head of a rule

Success or fail

1. point(A,B) = point(1,2).
2. point(A,B) = point(X,Y,Z).
3. plus(2,2) = 4.
4. +(2,D) = +(E,2).
5. triangle(point(-1,0),P2,P3)=triangle(P1,point(1,0), point(0,Y)).

Pattern Matching

- **Variable in**:
  - One side: foo(a)=foo(A).
  - Both side: foo(A)=foo(A).
  - Mix (both contains variable & non-variable)
    foo(a,b)=foo(a,B).
    foo(A,b)=foo(A,B).

Example:

binatang(comel).
binatang(tomok).
binatang(boboy).

makan(comel,ikan).
makan(tomok,ikan).
makan(boboy,jagung).
makan(boboy,rani).

mengiau(comel).
mengiau(tomok).
mengiau(boboy).

binatang(X) :-
  binatang(X),
  makan(X,ikan),
  mengiau(X).

makan(X,ikan) = makan(comel,ikan)

Predicate not defined

6. plus(2,2) = P.

Pattern Matching

- To become identical – the variables will be instantiated – **INSTANTIATION**.
- Assign value to a variable in order to achieve a match.
- Example: foo(a)=foo(A). A=a
  foo(A)=foo(a,B). A=A
  foo(A,b)=foo(a,B). A=a, B=b
  foo(A,b)=foo(A,B). A=A, B=b

Example

binatang(comel).
binatang(tomok).
binatang(boboy).

makan(comel,ikan).
makan(tomok,ikan).
makan(boboy,jagung).
makan(boboy,rani).

mengiau(comel).
mengiau(tomok).
mengiau(boboy).

binatang(X) :-
  binatang(X),
  makan(X,ikan),
  mengiau(X).

makan(X,ikan) = makan(comel,ikan)

mengiau(comel) - ERROR - Predicate not defined
Backtracking

- Prolog will automatically backtrack – for satisfying a goal.

- Useful – relieves the programmer of the burden of programming backtracking explicitly.

- In console, Prolog will backtrack automatically after we press “;”.

```
?- fruit(X), write('I like to eat '), write(X), nl.
I like to eat orange
X = orange;
I like to eat apple
X = apple;
---
```

Backtracking

- This is inconvenience to some problem.

- Example:

```
?- fruit(X), write('I like to eat '), write(X), nl.
I like to eat orange
X = orange;
I like to eat apple
X = apple;
---
```

Backtracking

- To force backtracking use fail/0.

- Example:

```
?- fruit(X), write('I like to eat '), write(X), nl, fail.
I like to eat orange
I like to eat apple
I like to eat banana
```