September 1st Session 2014/2015 (A141)

Wan Hussain Wan Ishak

School of Computing
UUM College of Arts and Sciences
Universiti Utara Malaysia

(P) 04-9285150
(E) hussain@uum.edu.my
(U) http://wanhussain.com
Lecture notes

Overview of Prolog

- Main components
- Syntax
- Terms
- Pattern matching, unification and instantiation
- Backtracking
Introduction to logic...

- A form of knowledge representation.

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

<table>
<thead>
<tr>
<th>Premise</th>
<th>Logical Process</th>
<th>Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact</td>
<td></td>
<td>New fact</td>
</tr>
<tr>
<td>Sentence</td>
<td></td>
<td>Action</td>
</tr>
<tr>
<td>Observation</td>
<td></td>
<td>Conclusion</td>
</tr>
<tr>
<td>Information</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
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.
Prolog vs Lisp

What difference?

Prolog has an automated reasoning procedure – an INFERENCE ENGINE.

Programs that perform logical reasoning are much easier to write in 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.
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:
in\_Malaysia and
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:

state(kedah).
wan_hussain.
Terminology

- Fact
- Rule
- Predicate
- Argument
- Arity

Predicate that depend on other predicates/facts or information.

Example:

\[
\text{Is\_in}(X,Y) :- \\
\text{state}(X), \\
\text{country}(Y,X).
\]
Terminology

- Fact
- Rule
- Predicate
- Argument
- Arity

**Consists of name, bracket, and arguments**

**Example:**

```
country(malaysia).
```
Terminology

- Fact
- Rule
- Predicate
- Argument
- Arity

Element in predicate (written in bracket)

Example:

For country(malaysia) the argument is ‘malaysia’.
Terminology

- Fact
- Rule
- Predicate
- Argument
- Arity

Referring to the number of arguments in predicate.

Example:

state(malaysia).
→ state/1

father(ahmad, karim).
→ father/2
foo.
→ foo/0
Arity

- Two distinct predicates can have the same name if they have different arities.

  Example:

  eat(ahmad,rice).
eat(chicken).

- Predicate is identified by its name, a slash, and its arity.

  Example:

  eat(ahmad,nasi). → eat/2
  eat(ahmad).    → eat/1
Syntax

The fundamental units of Prolog syntax are:

- Atoms,
- Numbers,
- Structures, and
- Variables

(Source Bratko, 2001)
Syntax - Atoms

- Used as names of individuals and predicates.
- Begins with a lowercase letter
- Can contain letters, digits, and the underscore mark (_)

Example:

a
kedah
fatihah1982
muhamad_shahrul_aiman_rashid
‘Malaysia’
‘17638’
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>-1.254</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

netbook(acer, 1500).

- The atom at the beginning is called **FUNCTOR** of the structure.

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

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

- Contains of a string of letters, digit, and underscore.

- Begin with capital letters or the underscore mark.

Example:

<table>
<thead>
<tr>
<th>A</th>
<th>Var1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_Name</td>
<td>_name</td>
</tr>
<tr>
<td>_College</td>
<td>Id_17638</td>
</tr>
</tbody>
</table>
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,__).
X = sintok
X = jitra
X = kuantan
X = muar
```

```prolog
is_in(sintok, kedah).
is_in(jitra, kedah).
is_in(kuantan, pahang).
is_in(muar, johor).
```
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
Exercise

What is wrong with this one?

capital of (alor star, kedah).
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:
  
  \[ a = a. \]
  \[ \text{state}(\text{kedah}) = \text{state}(\text{kedah}). \]
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

- 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

Example:

\[
\text{is\_in(ke\text{dah},\text{malaysia})} = \text{is\_in(ke\text{dah},\text{malaysia})}
\]

\[
\rightarrow \text{is\_in/2}
\]

2 arity 2 arity

Same predicate name

Same sequence of arguments
Pattern Matching

Example: `is_in(kedah, Malaysia)`. `is_in(kedah, Malaysia)`. `is_in(kedah, Malaysia)`. `is_in` (kedah, malaysia) .
Pattern Matching

- In Prolog, matching process is also called UNIFICATION.

- Unification between:
  - Query with the fact
  - Query with head of a rule
Pattern Matching

- Variable in:

  - One side \( \text{foo}(a)=\text{foo}(A) \).
  - Both side \( \text{foo}(A)=\text{foo}(A) \).
  - Mix (both contains variable & non-variable)
    \[
    \text{foo}(A,b)=\text{foo}(a,B).
    \text{foo}(A,b)=\text{foo}(A,B).
    \]
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). \quad A=a
  - foo(A)=foo(A). \quad A=A
  - foo(A,b)=foo(a,B). \quad A=a, B=b
  - foo(A,b)=foo(A,B). \quad A=A, B=b
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)).

6. plus(2,2) = P.

7. Siti = penyanyi(siti).

8. eat = makan.
Example

binatang(comel).
binatang(tompok).
binatang(hitam).
binatang(boboy).

makan(comel, ikan).
makan(tompok, ikan).
makan(boboy, jagung).
makan(hitam, nasi).

mengiau(comel).
mengiau(tompok).
mengiau(hitam).
mengiau(boboy).

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

binatang(X) = binatang(comel)
makan(X, ikan) = makan(comel, ikan)
mengiau(X) = mengiau(comel)
Example

\begin{align*}
\text{binatang(comel)}. & \\
\text{binatang(tompok)}. & \\
\text{binatang(hitam)}. & \\
\text{binatang(boboy)}. & \\
\text{makan(comel, ikan)}. & \\
\text{makan(tompok, ikan)}. & \\
\text{makan(boboy, jagung)}. & \\
\text{makan(hitam, nasi)}. & \\
\text{mengiau(cindai)}. & \\
\text{mengiau(tompok)}. & \\
\text{mengiau(hitam)}. & \\
\text{mengiau(boboy)}. & \\
\text{kucing(X)} : - & \\
& \text{binatang(X)}, \\
& \text{makan(X, ikan)}, \\
& \text{mengiau(X)}. & \\
\end{align*}

- ERROR - \textit{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 “;”.
Backtracking

- This is inconvenience to some problem.

- Example:

  ```prolog
  ?- 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:

```prolog
?- 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