What is Prolog?

What is Prolog?

• Prolog is a language which approaches problem-solving in a declarative manner. The idea is to define what the problem is, rather than how it should be solved.

• In practice, most Prolog programs have a procedural as well as a declarative component — the procedural aspects are often necessary in order to make the programs execute efficiently.

3 What is Prolog?

Algorithm = Logic + Control

Robert A. Kowalski

Prescriptive Languages:

• Describe how to solve problem
• Pascal, C, Ada,...
• Also: Imperative, Procedural

Descriptive Languages:

• Describe what should be done
• Also: Declarative

Kowalski’s equation says that

• Logic – is the specification (what the program should do)

• Control – what we need to do in order to make our logic execute efficiently. This usually includes imposing an execution order on the rules that make up our program.
Objects & Relationships

Prolog programs deal with

• objects, and
• relationships between objects

English: “Christian likes the record”

Prolog: likes(christian, record).

Facts

Record Database

Here’s an excerpt from Christian’s record database:

is_record(planet_waves).
is_record(desire).
is_record(slow_train).

recorded_by(planet_waves, bob_dylan).
recorded_by(desire, bob_dylan).
recorded_by(slow_train, bob_dylan).

recording_year(planet_waves, 1974).
recording_year(desire, 1975).
recording_year(slow_train, 1979).

Record Database...

The data base contains unary facts (is_record) and binary facts (recorded_by, recording_year).

The fact

is_record(slow_train)

can be interpreted as

slow_train is-a-record

The fact recording_year(slow_train, 1979) can be interpreted as the recording year of slow_train was 1979.
Conditional Relationships

10 Conditional Relationships

- Prolog programs deal with conditional relationships between objects.

English: 
“C. likes Bob Dylan records recorded before 1979”

Prolog:
likes(christian, X) :-
   is_record(X),
   recorded_by(X, bob_dylan),
   recording_year(X, Year),
   Year < 1979.

11 Conditional Relationships. . .

- The rule

    likes(christian, X) :-
      is_record(X),
      recorded_by(X, bob_dylan),
      recording_year(X, Year),
      Year < 1979.

    can be restated as

    “Christian likes X, if X is a record, and X is recorded by Bob Dylan, and the recording year is before 1979.”

- Variables start with capital letters.
- Comma (“,”) is read as and.

12 Asking Questions

13 Asking Questions

Prolog programs

- solve problems by asking questions.

English: 

“Does Christian like the albums Planet Waves & Slow Train?”
?- likes(christian, planet_waves).
yes
?- likes(christian, slow_train).
no

14 Asking Questions...

English:

“Was *Planet Waves* recorded by Bob Dylan?”
“When was *Planet Waves* recorded?”
“Which album was recorded in 1974?”

Prolog:

?- recorded_by(planet_waves, bob_dylan).
yes
?- recording_year(planet_waves, X).
X = 1974
?- recording_year(X, 1974).
X = planet_waves

15 Asking Questions...

In Prolog

- "," (a comma), means "and’

English:

“Did Bob Dylan record an album in 1974?”

Prolog:

?- is_record(X),
    recorded_by(X, bob_dylan),
    recording_year(X, 1974).
yes

16 Asking Questions...

Sometimes a query has more than one answer:

- Use ";" to get all answers.

English:

“What does Christian like?”

Prolog: 
?- likes(christian, X).
  X = planet_waves ;
  X = desire ;
no

17 Asking Questions...

Sometimes answers have more than one part:

<table>
<thead>
<tr>
<th>English</th>
<th>Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td>“List the albums and their artists!”</td>
<td></td>
</tr>
</tbody>
</table>

?- is_record(X), recorded_by(X, Y).
X = planet_waves,
Y = bob_dylan ;
X = desire,
Y = bob_dylan ;
X = slow_train,
Y = bob_dylan ;
no

18 Recursive Rules

19 Recursive Rules

“People are influenced by the music they listen to.

People are influenced by the music listened to by the people they listen to.”

listens_to(bob_dylan, woody_guthrie).
listens_to(arlo_guthrie, woody_guthrie).
listens_to(van_morrison, bob_dylan).
listens_to(dire_straits, bob_dylan).
listens_to(bruce_springsteen, bob_dylan).
listens_to(bjork, bruce_springsteen).

influenced_by(X, Y) :- listens_to(X, Y).
influenced_by(X, Y) :- listens_to(X, Z),
                      influenced_by(Z, Y).

20 Asking Questions...
“Is Björk influenced by Bob Dylan?”
“Is Björk influenced by Woody Guthrie?”
“Is Bob Dylan influenced by Bruce Springsteen?”

Prolog: 

?- influenced_by(bjork, bob_dylan).
yes
?- influenced_by(bjork, woody_guthrie).
yes
?- influenced_by(bob_dylan, bruce_s).
no

21 Visualizing Logic

- Comma (,) is read as and in Prolog. Example: The rule

  \[\text{person}(X) :- \text{has}\_\text{bellybutton}(X), \text{not}\_\text{dead}(X).\]

  is read as

  “X is a person if X has a bellybutton and X is not dead.”

- Semicolon (;) is read as or in Prolog. The rule

  \[\text{person}(X) :- X=adam ; X=eve ; \text{has}\_\text{bellybutton}(X).\]

  is read as

  “X is a person if X is adam or X is eve or X has a bellybutton.”

22 Visualizing Logic...

- To visualize what happens when Prolog executes (and this can often be very complicated!) we use the following two notations:

  - For AND, both legs have to succeed.
  - For OR, one of the legs has to succeed.
23 Visualizing Logic...

- Here are two examples:

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>- has_bellybutton(X), not_dead(X).</code></td>
<td><code>- X=adam ; X=eve ; has_bellybutton(X).</code></td>
</tr>
<tr>
<td><code>has_bellybutton(X)</code></td>
<td><code>X=adam X=eve has_bellybutton(X)</code></td>
</tr>
<tr>
<td><code>not_dead(X)</code></td>
<td><code>not_dead(X)</code></td>
</tr>
</tbody>
</table>

24 Visualizing Logic...

- `and` and `or` can be combined:

```prolog
?- (X=adam ; X=eve ; has_bellybutton(X)), not_dead(X).
```

- This query asks
  “Is there a person X who is adam, eve, or who has a bellybutton, and who is also not dead?”

25 How does Prolog Answer Questions?

26 Answering Questions

(1) `scientist(helder).`.
(2) `scientist(ron).`.
(3) `portuguese(helder).`.
(4) `american(ron).`.
(5) `logician(X) :- scientist(X).`.
(6) `?- logician(X), american(X).`.

- The rule (5) states that
  “Every scientist is a logician”

- The question (6) asks
  “Which scientist is a logician and an american?”
27 Answering Questions...

28 Answering Questions...

(1) scientist(helder).
(2) scientist(ron).
(3) portuguese(helder).
(4) american(ron).
(5) logician(X) :- scientist(X).
(6) ?- logician(X), american(X).
Answering Questions...

(?- logician(X), american(X).
  logician(X)
  (6)
  scientist(X)
  (1)
  (2)
  scientist(ron)fail
  scientist(helder)
  american(helder)
  american(ron)
  american(X)
  x=ron

is_record(planet_waves).
is_record(desire).

is_record(slow_train).

recorded_by(planet_waves, bob_dylan).
recorded_by(desire, bob_dylan).
recorded_by(slow_train, bob_dylan).

recording_year(planet_waves, 1974).
recording_year(desire, 1975).
recording_year(slow_train, 1979).

likes(christian, X) :-
  is_record(X), recorded_by(X, bob_dylan),
  recording_year(X, Year), Year < 1979.

Answering Questions...

(?- likes(christian, X)
  is_record(X)artist(X, bob_d)recording_year(X, Y)Y<1979

X = planet_waves Y=1974 succeed
X = desire Y=1975 succeed
X = slow_train Y=1974 fail

Answering Questions...

listens_to(bob_dylan, woody_guthrie).
listens_to(arlo_guthrie, woody_guthrie).
listens_to(van_morrison, bob_dylan).
listens_to(dire_straits, bob_dylan).
listens_to(bruce_springsteen, bob_dylan).
listens_to(björk, bruce_springsteen).

(1) influenced_by(X, Y) :- listens_to(X, Y).
(2) influenced_by(X, Y) :-
    listens_to(X, Z),
    influenced_by(Z, Y).

?- influenced_by(bjork, bob_dylan).
?- inf_by(bjork, woody_guthrie).

33 Answering Questions...

34 Answering Questions...

35 Map Coloring

“Color a planar map with at most four colors, so that contiguous regions are colored differently.”
36  Map Coloring...

A coloring is OK iff

1. The color of Region 1 ≠ the color of Region 2, and
2. The color of Region 1 ≠ the color of Region 3, ...

\[
\text{color}(R_1, R_2, R_3, R_4, R_5, R_6) :-
\text{diff}(R_1, R_2), \text{diff}(R_1, R_3), \text{diff}(R_1, R_5), \text{diff}(R_1, R_6), \text{diff}(R_2, R_3), \text{diff}(R_2, R_4), \text{diff}(R_2, R_5), \text{diff}(R_2, R_6), \text{diff}(R_3, R_4), \text{diff}(R_3, R_6), \text{diff}(R_5, R_6).
\]

\[
\text{diff}(\text{red}, \text{blue}). \quad \text{diff}(\text{red}, \text{green}). \quad \text{diff}(\text{red}, \text{yellow}).
\]
\[
\text{diff}(\text{blue}, \text{red}). \quad \text{diff}(\text{blue}, \text{green}). \quad \text{diff}(\text{blue}, \text{yellow}).
\]
\[
\text{diff}(\text{green}, \text{red}). \quad \text{diff}(\text{green}, \text{blue}). \quad \text{diff}(\text{green}, \text{yellow}).
\]
\[
\text{diff}(\text{yellow}, \text{red}). \text{diff}(\text{yellow}, \text{blue}). \quad \text{diff}(\text{yellow}, \text{green}).
\]

37  Map Coloring...

?- \text{color}(R_1, R_2, R_3, R_4, R_5, R_6).
\text{R}_1 = \text{R}_4 = \text{red}, \text{R}_2 = \text{blue},
\text{R}_3 = \text{R}_5 = \text{green}, \text{R}_6 = \text{yellow} ;

\text{R}_1 = \text{red}, \text{R}_2 = \text{blue},
\text{R}_3 = \text{R}_5 = \text{green}, \text{R}_4 = \text{R}_6 = \text{yellow}

38  Map Coloring – Backtracking
40 Working with gprolog

- gprolog can be downloaded from here: http://gprolog.inria.fr/.
- gprolog is installed on lectura (it’s also on the Windows machines) and is invoked like this:

```
> gprolog
GNU Prolog 1.2.16
| ?- [color].
| ?- listing.
go(A, B, C, D, E, F) :- next(A, B), ...
| ?- go(A,B,C,D,E,F).
A = red ...
```

41 Working with gprolog...

- The command [color] loads the prolog program in the file color.pl.
- You should use the texteditor of your choice (emacs, vi,...) to write your prolog code.
- The command listing lists all the prolog predicates you have loaded.
42 Working with gprolog...

43 Readings and References

- Read Clocksin-Mellish, Chapter 1-2.
- [Prolog by Example](#) Coelho & Cotta
- [Prolog: Programming for AI](#) Bratko
- [Programming in Prolog](#) Clocksin & Mellish
- [The Craft of Prolog](#) O’Keefe
- [Prolog for Programmers](#) Kluzniak & Szpakowicz
- [Prolog](#) Alan G. Hamilton
- [The Art of Prolog](#) Sterling & Shapiro

44 Readings and References...

<table>
<thead>
<tr>
<th>Computing with Logic</th>
<th>Maier &amp; Warren</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Systems Through Prolog</td>
<td>Steven H. Kim</td>
</tr>
<tr>
<td>Natural Language Processing in Prolog</td>
<td>Gazdar &amp; Mellish</td>
</tr>
<tr>
<td>Language as a Cognitive Process</td>
<td>Winograd</td>
</tr>
<tr>
<td>Prolog and Natural Language Analysis</td>
<td>Pereira and Shieber</td>
</tr>
<tr>
<td>Computers and Human Language</td>
<td>George W. Smith</td>
</tr>
<tr>
<td>Introduction to Logic</td>
<td>Irving M. Copi</td>
</tr>
<tr>
<td>Beginning Logic</td>
<td>E.J. Lemmon</td>
</tr>
</tbody>
</table>

45 Prolog So Far

- A Prolog program consists of a number of clauses:
  - Rules: `head + body;`
\[
\text{likes(chris, X) :-}
\text{girl(X), black_hair(X)}
\]

- Can be recursive

Facts
- Head but no body.
- Always true.

46 Prolog So Far...

- A clause consists of
  - atoms Start with lower-case letter.
  - variables Start with upper-case letter.

Prolog programs have a
- Declarative meaning
  * The relations defined by the program
- Procedural meaning
  * The order in which goals are tried

47 Prolog So Far...

- A question consists of one or more goals:
  - \text{?- likes(chris, X), smart(X).}
  - "," means and
  - Use ";" to get all answers
  - Questions are either
    * Satisfiable (the goal succeeds)
    * Unsatisfiable (the goal fails)
  - Prolog answers questions (satisfies goals) by:
    * instantiating variables
    * searching the database sequentially
    * backtracking when a goal fails