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.

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).

Record Database

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

is_record(planet_waves).

recorded_by(planet_waves, bob_dylan).

recording_year(planet_waves, 1974).

is_record(desire).

recorded_by(desire, bob_dylan).

recording_year(desire, 1975).

is_record(slow_train).

recorded_by(slow_train, bob_dylan).

recording_year(slow_train, 1979).

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

The fact

\[ \text{is_record(slow\_train)} \]

can be interpreted as

\[ \text{slow\_train is-a-record} \]

The fact recording_year(slow\_train, 1979) can be interpreted as the recording year of slow\_train was 1979.

Conditional Relationships

Prolog programs deal with conditional relationships between objects.

\[ \text{likes(christian, X)} :\neg
\text{is_record(X)},
\text{recorded\_by(X, bob\_dylan)},
\text{recording\_year(X, Year)},
\text{Year < 1979}. \]

The rule

\[ \text{likes(christian, X)} :\neg
\text{is_record(X)},
\text{recorded\_by(X, bob\_dylan)},
\text{recording\_year(X, Year)},
\text{Year < 1979}. \]

can be restated as

\[ \text{“Christian likes X, if X is a record, and X is recorded by Bob Dylan, and the recording year is before 1979.”} \]

Variables start with capital letters.
Comma ("," is read as and.
Prolog programs solve problems by asking questions.

"Does Christian like the albums Planet Waves & Slow Train?"

Prolog:

?- likes(christian, planet_waves).
yes
?- likes(christian, slow_train).
no

"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

In Prolog

"," (a comma), means "and"

"Did Bob Dylan record an album in 1974?"

Prolog:

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

---

Sometimes answers have more than one part:

---

**English:**

“List the albums and their artists!”

---

**Prolog:**

?- 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

---

**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(björk, bruce springsteen).

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

English:

"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

Visualizing Logic

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

\[ \text{person}(X) :- \text{has_bellybutton}(X), \text{not_dead}(X). \]

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

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

is read as

"X is a person if X is adam or X is eve or X has a bellybutton."

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.

Here are two examples:

\[ ?- \text{has_bellybutton}(X), \text{not_dead}(X). \]
\[ ?- X=\text{adam} ; X=\text{eve} ; \text{has_bellybutton}(X). \]
and and or can be combined:

?- (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?"

How does Prolog Answer 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?"
(1) scientist(helder).
(2) scientist(ron).
(3) portuguese(helder).
(4) american(ron).
(5) logician(X) :- scientist(X).
(6) ?- logician(X), american(X).

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...

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(björk, bob_dylan).
?- inf_by(björk, woody_guthrie).

Answering Questions...

Map Coloring

"Color a planar map with at most four colors, so that contiguous regions are colored differently."

3 4 2 1
5 6

Color a planar map with at most four colors, so that contiguous regions are colored differently."
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,...

color(R1, R2, R3, R4, R5, R6) :-
diff(R1, R2), diff(R1, R3), diff(R1, R5), diff(R1, R6),
diff(R2, R3), diff(R2, R4), diff(R2, R5), diff(R2, R6),
diff(R3, R4), diff(R3, R6), diff(R5, R6).

diff(red,blue). diff(red,green). diff(red,yellow).
diff(blue,red). diff(blue,green). diff(blue,yellow).
diff(green,red). diff(green,blue). diff(green,yellow).
diff(yellow, red).diff(yellow,blue). diff(yellow,green).

?- color(R1, R2, R3, R4, R5, R6).
R1 = R4 = red, R2 = blue,
R3 = R5 = green, R6 = yellow ;
R1 = red, R2 = blue,
R3 = R5 = green, R4 = R6 = yellow

?- color(R1, R2, R3, R4, R5, R6).
R1 = R4 = red, R2 = blue,
R3 = R5 = green, R6 = yellow

Map Coloring – Backtracking

Map Coloring – Backtracking
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 ...

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.

Read Clocksin-Mellish, Chapter 1-2.
http://dmoz.org/Computers/Programming/Languages/Prolog

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
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

A question consists of one or more goals:
- `?- 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