Reading

Programming in Prolog

Sept 3 Chapters 1 & 2

Sept 8 & 10 - Chapters 3-5

Sept 15 & 17 - Chapters 6-8
Prolog

1972 - Created by Alain Colmerauer

Abbreviation for "programmation en logique"

Logic programming
Prolog Data Types

Term

Atom
   cat  mom  'Roger'  'help me'

Number
   12  32

Variable
   X  WhyMe  _WhatIsThis

Compound term
   mother_child(sally, tom)
   [1, 2, 3]
Prolog Program

Contains
  Set of facts
  Rules about the facts

We ask questions about rules and facts
Hello World

display('Hello World').

display(helloWorld).
Addition

X is 1 + 2.
Running the Examples

?- display('Hello World').
Hello World
true.

?- display(helloWorld).
helloWorld
true.

?- X is 1 + 2.
X = 3.

?-
Demo Swi-Prolog

?- display('Hello World').
Hello World
true.

?- display(helloWorld).
helloWorld
true.

?- X is 1 + 2.
X = 3.

?-
atoms verses Variables

Atoms
Starts with lower case character
Surround with single quotes if
Contains space
Starts with capital letter

Variable
Starts with capital letter or underscore
What happens here?

?- display(Hello).
More Definitions

functor          arguments

mother_child(sally, tom)

functor & arguments are terms

arity - number of arguments

mother_child/2
% (**) Determine whether a given integer number is prime.

% is_prime(P) :- P is a prime number
%    (integer) (+)

is_prime(2).

is_prime(3).

is_prime(P) :- integer(P), P > 3, P mod 2 =\= 0, \+ has_factor(P,3).

% has_factor(N,L) :- N has an odd factor F >= L.
%    (integer, integer) (+,+)

has_factor(N,L) :- N mod L =:= 0.
has_factor(N,L) :- L * L < N, L2 is L + 2, has_factor(N,L2).

Example from https://prof.ti.bfh.ch/hew1/informatik3/prolog/p-99/
Details

Facts

is_prime(2).
is_prime(3).
Rule

is_prime(P) :- integer(P), P > 3, P mod 2 ≠ 0, \+ has_factor(P,3).

P is prime if P is an integer and P > 3 and P mod(2) ≠ 0 and has no factors between P and 3

:- means if
, means and
\+ means not (at least or now)
More Rules

\[
\text{has\_factor}(N,L) :\text{-} N \mod L =: 0.
\]

\[N\text{ has a factor between } N \text{ and } L \text{ if } N \mod(L) = 0\]

\[
\text{has\_factor}(N,L) :\text{-} L \ast L < N, L_2 \text{ is } L + 2, \text{ has\_factor}(N,L_2).
\]

\[N\text{ has a factor between } N \text{ and } L \text{ if}
L\ast L < N \text{ and has\_factor}(N, L + 2) \text{ is true}\]
Running ?? the Example?

Place source code in a file called "prime.pl"

?- consult(prime).
% prime compiled 0.00 sec, 0 bytes
true.

?- is_prime(101).
true.

?-
Some Swi Prolog Documentation

?- manpce.

Opens the XPCE Manual
Second Example - Family

Facts and Rules

mother_child(susan, sally).
mother_child(susan, matt).

father_child(tom, sally).
father_child(tom, erica).
father_child(tom, pete).
father_child(mike, tom).

sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).
Syntax

mother_child(susan, sally).

What is

  mother_child
  susan
  sally
Semantics

mother_child(susan, sally).

Stating a fact about a relationship between susan & sally

What do we mean when we say

susan is the mother, sally is the child
And Rule

sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).
For now we can consider this an or. We will get a more precise definition later and an or operator.
Some Questions

?- consult(family).
% family compiled 0.00 sec, 64 bytes
true.

?- father_child(tom,sally).
true.

?- father_child(tom,pete).
true.
Asking for More Answers

?- father_child(tom,X).
X = sally.

?- father_child(tom,X).
X = sally ;
X = erica.

?- father_child(tom,X).
X = sally ;
X = erica ;
X = pete.

?-

The ';' means to continue to look for another answer
Order of the Rules Matters

father_child(tom, sally).
father_child(tom, erica).
father_child(tom, pete).

?- father_child(tom,X).
  X = sally.

father_child(tom, pete).
father_child(tom, sally).
father_child(tom, erica).

?- father_child(tom,X).
  X = pete.
More Complex Questions

?- sibling(sally,pete).
true.

?- sibling(sally,matt).
true.

?- sibling(matt,sally).
true.

?- sibling(matt,pete).
false.
Using A Variable

?- sibling(sally,X).
X = sally ;
X = erica ;
X = pete ;
X = sally ;
X = matt .
Using Two Variables

?- sibling(X,Y).
X = sally,
Y = sally ;
X = sally,
Y = erica ;
X = sally,
Y = pete ;
X = erica,
Y = sally ;
X = erica,
Y = erica ;
X = erica,
Y = pete ;
X = pete,
Y = sally ;
X = pete,
Y = erica
trace

Shows what rules/facts are being used

?- trace(father_child).
% father_child/2: [call, redo, exit, fail]
true.

[debug]  ?- sibling(sally,pete).
sibling
  T Call: (10) father_child(_L176, sally)
  T Exit: (10) father_child(tom, sally)
  T Call: (10) father_child(tom, pete)
  T Exit: (10) father_child(tom, pete)
true.
More Trace

?- trace(father_child).
% father_child/2: [call, redo, exit, fail]
true.

[debug]  ?- trace(parent_child).
% parent_child/2: [call, redo, exit, fail]
true.

[debug]  ?- sibling(sally,pete).
sibling
  T Call: (9) parent_child(_L176, sally)
  T Call: (10) father_child(_L176, sally)
  T Exit: (10) father_child(tom, sally)
  T Exit: (9) parent_child(tom, sally)
  T Call: (9) parent_child(tom, pete)
  T Call: (10) father_child(tom, pete)
  T Exit: (10) father_child(tom, pete)
  T Exit: (9) parent_child(tom, pete)
true