References

Programming in Prolog, 5Ed., Clocksin & Mellish
SWI Prolog Reference Manual
repeat

succeed infinite number of times on backtracking

repeat.
repeat -: repeat.
repeat Example

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

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

related(X,Y) :- father_child(_, X), sibling(X,Y).

sibling(X, Y) :-
    parent_child(Z, X),
    repeat,
    parent_child(Z, Y),
    \+ X = Y.

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

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

new_get(X) :- repeat, get_char(X).

get_non_space(X) :- new_get(X), \+ X = ' '.

?- get_non_space(X).
|: a.
X = a .

?- get_non_space(X).
|: a b c d.
X = a ;
X = b ;
X = c ;
X = d .
if_example(X,Y) :- X > 2 -> Y is 0; Y is 10.

?- if_example(1,Y).
    Y = 10.

?- if_example(5,Y).
    Y = 0.
maplist

maplist(:Pred, +List)
  Apply Pred to elements of List
  Return false when Pred fails
  Return true if Pred is true for all elements

?- maplist(number,[1,2,3]).
  true.

?- maplist(number,[1,g,3]).
  false.
findall

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

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

related(X, Y) :- father_child(_, X), sibling(X, Y).

sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y), \+ X = Y.

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

?- findall(X, sibling(X, sally), Y).
Y = [erica, pete, matt].
Dynamic Predicates

Predicates that can be changed by a program

:- dynamic found/1.
assert & retract

asserta
  Add a clause (dynamic) to the beginning of the database

assertb
  Add a clause (dynamic) to the end of the database

retract
  Remove a clause (dynamic)
compile_predicates(:ListOfNameArity)

Compiles dynamic predicates into static predicated
Find all implementation

:- dynamic found/1.

find_all(X, G, _) :-
    asserta(found(mark)),
    call(G),
    asserta(found(result(X))),
    fail.

find_all(_,_,L) :- collect_found([], M), !, L = M.

collect_found(S, L) :-
    getnext(X),
    !,
    collect_found([X|S],L).

collect_found(L, L).

collect_found(Y) :- retract(found(X)), !, X = result(Y).

?- findall(X,sibling(X,sally),Y).
Y = [erica, pete, matt].
Strings

?- string_to_atom(X,cat).
X = "cat".

?- string_to_atom(X,'cat mat').
X = "cat mat".

?- string_to_atom(X,123).
X = "123".
hanoi(N) :- move(N, left, center, right).

move(0, _, _, _) :- !.
move(N, A, B, C) :-
    M is N - 1,
    move(M, A, C, B), inform(A, B), move(M, C, B, A).

inform(X, Y) :-
    write([move, from, X, to, Y]),
    nl.

?- hanoi(3).
[move, from, left, to, center]
[move, from, left, to, right]
[move, from, center, to, right]
[move, from, left, to, center]
[move, from, right, to, left]
[move, from, right, to, center]
[move, from, left, to, center]
true.
Profiling

?- profile(hanoi(15)).

![Profile Output]

File | Sort | Time | Help
---|---|---|---
Flat profile by time self
- nl/0 | 95.9%
- inform/2 | 1.4%
- $garbage_collect/1 | 1.1%
- move/4 | 1.1%
- write/1 | 0.6%
- is/2 | 0.0%
- hanoi/1 | 0.0%

363 samples in 0.23 sec; 7 predicates; 10 nodes in
Window-based Debugger

?- gspy(sibling).
% The graphical front-end will be used for subsequent tracing
% Spy point on sibling/2
true.

[debug]  ?- sibling(X,Y).
Threads

?- thread_create(hanoi(10),ID,[]).

All internal Prolog operations are thread-safe

mutex
mutex_create(?MutexId)
mutex_lock(+MutexId)
mutex_unlock(+MutexId)
mutex_unlock_all

Message Queues

thread_send_message(+QueueOrThreadId, +Term)
thread_get_message(?Term)
message_queue_create(?Queue)
Foreign Language Interface

SWI Prolog has an interface to C

C can call Prolog predicates
Standalone Executables

qsave_program(+File)

?- [family].
% family compiled 0.00 sec, 2,180 bytes
true.

?- qsave_program(testRun).
true.
Terminal Window Running testRun

Last login: Tue Sep 15 12:17:54 on ttys001
Al pro 11->/Users/whitney/Courses/520/Fall09/prologExamples/testRun ; exit;
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.6.64)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

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

?-
GUI Based Application

How?