Script generated by TTT

Title: Seidl: Virtual_Machines (28.05.2013)
Date: Tue May 28 14:04:41 CEST 2013
Duration: 81:08 min
Pages: 94

% linda -> peter
%       [--------]
% john -> mary  fred -> ann
%   |        |
% bob   carol

female(X) :- X=mary ,
female(X) :- X=ann ,
female(X) :- X=linda ,
female(X) :- X=carol .

male(X) :- X=john ,
male(X) :- X=bob ,
male(X) :- X=peter ,
male(X) :- X=fred .
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU 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(Topic).

?- female
| (mary).
ERROR: Syntax error: Operator expected
ERROR: female

ERROR: ** here **
ERROR: (mary).
?- female(mary).
true
Unknown action: h (for help)
Action?
Unknown action: [ (for help)
Action?
Unknown action: A (for help)
Action?.

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

?- female \\
|  (mary).
ERROR: Syntax error: Operator expected 
ERROR: female

ERROR: ** here **
ERROR: (mary).
?- female(mary).
true
Unknown action: H (h for help)
Action?
Unknown action: [ (h for help)
Action?
Unknown action: A (h for help)
Action?.

?- female(mary).
true \\
false.

?- female(X).
X = mary \\
false.

?- female(X).
X = mary \\
X = ann \\
X = linda \\
X = carol.

?-.
female(X) :- X=mary.
female(X) :- X=ann.
female(X) :- X=linda.
female(X) :- X=carol.

male(X) :- X=john.
male(X) :- X=bob.
male(X) :- X=peter.
male(X) :- X=fred.

?- female(mary).
   true.
   false.

?- female(X).
   X = mary ;
   X = ann ;
   X = linda ;
   X = carol.

?- female(bob).
   false.

?- female(mary).
   true.

?- female(mary).
   X = mary ;
   X = ann ;
   X = linda ;
   X = carol.

?- female(bob).
   false.

?-
% halt
kalmera@user:~/vm13$ vi test.pl
kalmera@user:~/vm13$ rlwrap swipl -s test.pl
% /home/kalmera/vm13/test.pl compiled 8.00 sec, 16,720 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
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).

?- parent(bob, mary).
true
?- female(X).
X = Mary ;
X = Ann ;
X = Linda ;
X = Carol.

?- female(bob).
false.

?- halt.
kamera@user:~/vm13$ vi test.pl
kamera@user:~/vm13$ rlwrap swipl -s test.pl
% /home/kamera/vm13/test.pl compiled 0.00 sec, 16720 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU 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).

?- parent(bob, mary).
true
Unknown action: , (h for help)
Action? .

?- parent(bob, P).
P = Mary ;
P = John ;
false.

?- female(bob).
false.

?- halt.
kamera@user:~/vm13$ vi test.pl
kamera@user:~/vm13$ rlwrap swipl -s test.pl
% /home/kamera/vm13/test.pl compiled 0.00 sec, 16720 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU 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).

?- parent(bob, mary).
true
Unknown action: , (h for help)
Action? .

?- parent(bob, P).
P = Mary ;
P = John ;
false.
For help, use ?- help(Topic). or ?- apropos(Word).

?- parent(bob, mary).
true

Unknown action: , (h for help)
Action? .

?- parent(bob, P).
P = mary ;
P = john ;
false.

?- parent(bob, P), male(P).
P = John ;
false.

?- parent(bob, P), male(P).
P = mary ;
false.

?- parent(bob, P), male(P).
P = john ;
false.

?- parent(bob, P); male(P).
P = mary ;
false.

?- parent(bob, P); male(P).
P = john ;
false.

?- parent(C,linda).
C = mary ;
C = fred ;
false.

?- parent(C,P).
C = bob ,
P = mary
female(X) : X=ann.
female(X) : X=linda.
female(X) : X=carol.
male(X) : X=john.
male(X) : X=bob.
male(X) : X=peter.
male(X) : X=fred.

parent(Child,Parent) : Parent=mary.
parent(Child,Parent) : Parent=mary.
parent(Child,Parent) : Parent=peter.
parent(Child,Parent) : Parent=linda.
parent(Child,Parent) : Parent=fred.
parent(Child,Parent) : Parent=carol.
parent(Child,Parent) : Parent=ann.

father(C,F) : parent(C,F), male(F).
mother(C,M) : parent(C,M), female(M).

son(P,S) : parent(P,S), male(S).
daughter(P,D) : parent(P,D), female(D).

grandparent(C,G) : parent(C,P), parent(P,G).

--- INSERT ---

--- INSERT ---

--- INSERT ---

--- INSERT ---
?- halt

?- father(carol,F).
  F = fred ;
  false.

?- mother(carol,F).
  F = ann ;
  false.

?- son(fred,S).
  false.

?- ancestor(carol,X).
  X = fred ;
  X = ann ;
  X = linda ;
  X = peter ;
  false.

Copyright (c) 1990-2011 University of Amsterdam, VU 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.

?- father(carol,F).
  F = fred ;
  false.

?- mother(carol,F).
  F = ann ;
  false.

?- son(fred,S).
  false.

?- ancestor(carol,X).
  X = fred ;
  X = ann ;
  X = linda ;
  X = peter ;
  false.

parent(Child,Parent) :- Child = bob, Parent = mary.
parent(Child,Parent) :- Child = bob, Parent = john.
parent(Child,Parent) :- Child = mary, Parent = linda.
parent(Child,Parent) :- Child = mary, Parent = peter.
parent(Child,Parent) :- Child = fred, Parent = linda.
parent(Child,Parent) :- Child = fred, Parent = peter.
parent(Child,Parent) :- Child = carol, Parent = fred.
parent(Child,Parent) :- Child = carol, Parent = ann.

father(C,F) :- male, parent(C,F).
mother(C,M) :- parent(C,M), female(C).

son(P,S) :- parent(S,P), male(S).
daughter(P,D) :- parent(D,P), female(D).

grandparent(C,G) :- parent(C,P), parent(P,G).

ancestor(X,A) :- parent(X,A).
ancestor(X,A) :- parent(X,B), ancestor(B,A).
sister(X,S) :- parent(X,P), daughter(P,S), X \= S.
### SWI-Prolog Console Output

Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)

Copyright (c) 1980-2011 University of Amsterdam, VU 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).

?- sister(fred,S).
false.

?- sister(fred,S).
S = mary ;
S = mary .
false.

?- halt.

### Handwritten Notes

- \( p = \text{linda} \)
- \( p = \text{peter} \)

### Prolog Definitions

- `parent(Child,Parent) :- Child=mary, Parent=linda.`
- `parent(Child,Parent) :- Child=mary, Parent=peter.`
- `parent(Child,Parent) :- Child=fred, Parent=linda.`
- `parent(Child,Parent) :- Child=carol, Parent=fred.`
- `father(C,F) :- male(F), parent(C,F).`
- `mother(C,M) :- parent(C,M), female(M).`
- `son(P,S) :- parent(S,P), male(S).`
- `daughter(P,D) :- parent(D,P), female(D).`
- `grandparent(C,G) :- parent(C,P), parent(P,G).`
- `ancestor(X,A) :- parent(X,A).`
- `ancestor(X,A) :- parent(X,B), ancestor(B,A).`
- `sister(X,S) :- parent(X,P), daughter(P,S), X \neq S.`

### Example Queries

- `\( [1,2,3] = [1][2][3][\emptyset] \)`
- `\( [1,2,3] = [1][2][3][\emptyset] \)`
- `\( [3,3] = [3][3][\emptyset] \)`
- `\( [1,1,3] = [1][1][3][\emptyset] \)`

- `member(X,Y) :- Y = [X].`
- `member(X,Y) :- Y = [X], member(X,Z).`
- `member([1,2,3], Y).`
- `member([3,3], Y).`
- `member([1], X).`
- `male2(X) :- member(X,[john,bob,peter,fred]).`
\% [1,2,3] = [[1][2][3]]
\% \\
\% 1 [ ]
\% / \ \\
\% 2 [ ]
\% / \ \\
\% 3 [ ]

member(X, Y) :- Y = [X].
member(X, Y) :- Y = [_Z], member(X, Z).

\% member(X, [1,2,3]).
\% member([3,Y], [[1,a],[2,m],[3,z],[4,v],[3,p]]).
\% member([1,X]).

male2(X) :- member(X, [john, bob, peter, fred]).
append(X, Y, Z) :- X=[I], Y=Z.
append(X, Y, Z) :- X=[A|B], Z=[A|D], append(B, Y, D).

-- INSERT --

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).

?- sister(fred, S).
S = mary ;
false.

?-
% /home/kalmera/vml13/test.pl compiled 8.00 sec, 16,720 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU 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).

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

?-
ERROR: Undefined procedure: member/0
ERROR: However, there are definitions for:
ERROR: member/2
false.

?- member(1,X).
X = [1, G297];
X = [G297, [1, G301]];
X = [G297, G300, [1, G304]];
X = [G297, G300, G303, [1, G307]];
X = [G297, G300, G303, G306, [1, G310]];
X = [G297, G300, G303, G306, G309, [1, G313]];
X = [G297, G300, G303, G306, G309, G312, [1, G316]].

ERROR:
% halt
kalmera@user:/vm13$ vi test.pl
kalmera@user:/vm13$ rlwrap swipl -s test.pl
% /home/kalmera/vm13/test.pl compiled 0.00 sec, 16,720 bytes
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU 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).

?- append([1,2],[a,b],X).
X = [1, 2, a, b];
false.

?- append([p,[1,2,3]]).
P = [1];
P = [1];
P = [1, 2];
P = [1, 2, 3];
false.

?- append(\(-T, [1,2,3]).
T = [1, 2, 3];
T = [2, 3];
T = [3];
T = []; false.

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

?- append([1,2],[a,b],X).
X = [1, 2, a, b];
false.

?- append(P,_,[1,2,3]).
P = [1];
P = [1, 2];
P = [1, 2, 3];
false.

?- append(_,T,[1,2,3]).
T = [1, 2, 3];
T = [2, 3];
T = [3];
T = [1];
false.

?- append([1,X],P,[1,2,3]).
X = 2,
P = [3];
false.

?- % halt
kamelra@user:~/vm13$
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).

?- select([1,2,3,4],E,R).
  E = 1,
  R = [2, 3, 4] ;
  E = 2,
  R = [1, 3, 4] ;
  E = 3,
  R = [1, 2, 4] ;
  E = 4,
  R = [1, 2, 3] ;
false.

?- perm([1,2,3],P).
  P = [1, 2, 3] .
member(X, Y) :- Y = [X].
member(X, Y) :- Y = [Z], member(X, Z).

member(X, [1, 2, 3]).
member([3, Y], [[1, a], [2, m], [3, z], [4, v], [3, p]]).
member([1, X]).

male2(X) :- member(X, [john, bob, peter, fred]).

append(X, Y, Z) :- X = [A], Y = Z.
append(X, Y, Z) :- X = [A|B], Z = [A|D], append(B, Y, D).

prefix(Lst, Pre) :- append(Pre, __, Lst).

select(Xs, E, Rst) :- Xs = [E|Rst].
select(Xs, E, Rst) :- Xs = [A|Rst], Rst = [A|Ys], select(Zs, E, Ys).

perm(Xs, Ys) :- Xs = [], Ys = [].
perm(Xs, Ys) :- Xs = [O|W], perm(W, R), select(Ys, O, R).
sudoku(X) :- perm(X, [1, 2, 3, 4]).

all_sudoku(Xs) :- Xs = [1].
all_sudoku(Xs) :- Xs = [Y|Ys], sudoku(Y), all_sudoku(Ys).
list_to_single_lists(Xs, Yss) :- Xs = [], Yss = [].
list_to_single_lists(Xs, Yss) :- Xs = [Xh|Xt], Yss = [[Xh]|Yst], list_to_single_lists(Xt, Yst).

prep_each(Xs, Ys, Zs) :- Ys = [], Zs = [], Xs = [].
prep_each(Xs, Ys, Zs) :- Xs = [[Xh]|Xt], Ys = [H|T], Zs = [[Xh]|T2], prep_each(XT, T, Tz).

transpose(Xs, Ys) :- Xs = [Xh], list_to_single_lists(Xh, Ys).
transpose(Xs, Ys) :- Xs = [Xh|Xt], prep_each(Xh, Zs, Ys), transpose(XT, Zs).

valid_sudoku(Xs) :-
    transpose(Xs, Ys),
    boxes(Xs, Zs),
    all_sudoku(Xs),
    all_sudoku(Ys),
    all_sudoku(Zs).

print_sudoku(Xs) :- Xs = [], write(\n). print_sudoku(Xs).
print_sudoku(Xs) :- Xs = [[Yh]|Yt], write(Yh), write(\n), print_sudoku([Yh|Yt]).
For help, use `?- help(Topic).` or `?- apropos(Word).`

`?- sudoku([2,1,3,4]).`
true
Unknown action: , (h for help)
Action? .

`?- sudoku([2,1,3,_]).`
true ;
false.

`?- sudoku([2,1,3,X]).`
X = 4 ;
false.

`?- sudoku([2,1,Y,X]).`
Y = 3,
X = 4 ;
Y = 4,
X = 3 ;
false.

`?- list_to_single_lists([1,2,3],Z).
Z = [[1], [2], [3]] ;
false.

`?- `
?- list_to_single_lists([1,2,3],Z).
Z = [[1], [2], [3]].
false.

?- prep_each([1,2],[[a],[1]],
          | Z).
Z = [[1], [a], [2]].
false.

?- transpose([[1,2],[a,b]]).
ERROR: Syntax error: Illegal start of term
ERROR: transpose([[1,2],[a,b]]),
ERROR: ** here **
ERROR: )
?- transpose([[1,2],[a,b]],A).
A = [[1, a], [2, b]]
Unknown action: [[h for help]
Action?
Unknown action: [h for help]
Action?
Unknown action: A (h for help)
Action?
false.

?- transpose([[1,2],[a,b,c]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- transpose([[1,2],[a,b]],A).
false.

?- boxes([[1,2,3,4],[a,b,c,d],[9,8,7,6],[1,k,j,h]],A).
A = [[1, 2, a, b], [3, 4, c, d], [9, 8, l, k], [7, 6, j, h]].
Unknown action: [ (h for help)
Action?
Unknown action: A (h for help)
Action?

?- Aw([1, 2, 3, 4], [a, b, c, d], [9, 8, 7, 6], [l, k, j, h]), boxes(A, B), print_sudoku(B).

A = [[1, 2, 3, 4], [a, b, c, d], [9, 8, 7, 6], [l, k, j, h]],
B = [[1, 2, a, b], [3, 4, c, d], [9, 8, l, k], [7, 6, j, h]]
Unknown action: [ (h for help)
Action?
Unknown action: [ (h for help)
Action?
Unknown action: B (h for help)
Action?

?- example(X), print_sudoku(X).

_4343 _4346 _4349 _4352
_G358 G361 3 2
G373 G376 G379 _G382
4 _G391 _G394 1

X = [[G11, G14, G17, G20], [G26, G29, 3, 2], [G41, G44, G47, G50], [4, G59, G62, 1]]

X = [[G11, G14, G17, G20], [G26, G29, 3, 2], [G41, G44, G47, G50], [4, G59, G62, 1]]

Unknown action: [ (h for help)
Action?
Unknown action: [ (h for help)
Action?
Unknown action: A (h for help)
Action?

?- example(X), print_sudoku(X).