University of BRISTOL

Artificial Intelligence: More Debugging Tips

Oliver Ray

Inspecting Terms

- To aid readability, the SWI debugger (and SWI REPL) only print abbreviated terms by default:
  - ?- numlist(1,10,L).
  - L = [1, 2, 3, 4, 5, 6, 7, 8, 9|...].
- This behaviour can be toggled in the debugger (or forced in the REPL) by the following options
  - Hit "**p**" to **portray** terms as abbreviations (default)
  - Hit "**w**" to **write** terms in full
- Note that it is sometimes it is necessary to insert a choice point in order to allow user input:
  - ?- numlist(1,10,L) ; true.
  - L = [1, 2, 3, 4, 5, 6, 7, 8, 9|...] [write]
  - L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] [print]
  - L = [1, 2, 3, 4, 5, 6, 7, 8, 9|...] .

User pressed “w”

User pressed “p”

User pressed <enter>

- https://www.swi-prolog.org/FAQ/AllOutput.htm

Spypoints and Breakpoints

Once you are familiar with the basic tracing options, you can try setting spypoints or break-points to (re)initiate execution tracing on specific program lines or predicate ports:

- In SWISH
  - click on a program line number to set/unset a breakpoint (indicated by a solid circle)
- In the SWIPL top-level
  - Use the commands: **spy(pred/arity)**, **nospy(pred/arity)**, and **nospyall** to set and remove spypoints
- In the SWIPL tracer
  - Hit "**+**" to "**set a spy point**" on the current predicate
  - Hit "**-**" to "**remove spy points**" on the current predicate
  - Hit "**l**" to "**leap**" to the next spy point
- In SWIPL Input-Files
  - Use directives of the form :-**spy(pred/arity)** to set predefined spypoints
  - You can also introduce new predicates to act as conditional spypoints – as illustrated on the next slide

- https://www.swi-prolog.org/pldoc/man?section=debugoverview

Example

SWISH File Edit Examples Help 143 users online Search

lpn-c3-s4-q4

```
35 % ----------------------------------------------------
36
37 %%% PATHS USING LISTS %%%
38
39 leg(From,by_car,To) :- byCar(From,To).
40 leg(From,by_train,To) :- byTrain(From,To).
41 leg(From,by_plane,To) :- byPlane(From,To).
42
43 % ----------------------------------------------------
44 my_spy(_) :- write("start trace"). % :-spy(my_spy/1).
45 journey(A,B,C) :- A==B, \+my_spy(journey(A,B,C)).
46 % ----------------------------------------------------
47 journey(From,To,[From,By,To]) :-
48     leg(From,By,To).
49 journey(From,To,[From,By|Rest]) :-
50     leg(From,By,Next),
51     journey(Next,To,Rest).
52
```

journey(metz,frankfurt,Path).

Breakpoint 1295 in 1-st clause of my_spy/1 at line 44

Path = [metz, by_train, frankfurt]

Call: write("start trace")

start trace

Exit: write("start trace")

Exit: my_spy(journey(frankfurt,frankfurt,_1470))

Call: leg(frankfurt,_1492,frankfurt)

Fail: leg(frankfurt,_1484,frankfurt)

?- journey(metz,frankfurt,Path).

Examples History Solutions table results Run!

Determinism

```
DETERMINISTIC PREDICATE
(well-behaved: closing off the REDO port, i.e. "leaving no choicepoint")

             +---------------------+
     -------▷|Call------▷-----▷Succ|------▷
   from      |                     |      to
   prev      |                     |      next
   pred      |                     |      pred
     ◁---+   |Fail             Redo|   +---
         |   +---------------------+   |
         |                             |
         +---◁----no choicepoint----◁--+
```

```
NONDETERMINISTIC PREDICATE (succeeds maybe 0 times)
(well-behaved if it closes off the REDO port at the last solution, i.e. "leaves no choicepoint")

             +---------------------+
     -------▷|Call---+--▷--+--▷Succ|------▷
   from      |       |     |       |      to
   prev      |       ∇     Δ       |      next
   pred      |       |     |       |      pred
     ◁---+---|Fail◁--+--◁--+---Redo|◁--+---
         |   +---------------------+   |
         |                             |
         +---◁----no choicepoint----◁--+
```

```
SEMI-DETERMINISTIC PREDICATE
(well-behaved: closing off the REDO port, i.e. "leaving no choicepoint")

             +---------------------+
     -------▷|Call---+--▷-----▷Succ|------▷
   from      |       |             |      to
   prev      |       ∇             |      next
   pred      |       |             |      pred
     ◁---+---|Fail◁--+             |   +---
         |   +---------------------+   |
         |                             |
         +---◁----no choicepoint----◁--+
```

```
MULTI PREDICATE (succeeds at least once)
(well-behaved if it closes off the REDO port at the last solution, i.e. "leaves no choicepoint")

             +---------------------+
     -------▷|Call-------▷--+--▷Succ|------▷
   from      |             |       |      to
   prev      |             Δ       |      next
   pred      |             |       |      pred
     ◁---+---|Fail◁-------◁--+---Redo|◁--+---
         |   +---------------------+   |
         |                             |
         +---◁----no choicepoint----◁--+
```

Cuts – the good and the bad

size(X,small) :- X<5, ! . size(X,large) :- X>9, ! . % X>=5 size(X,medium). % X>=5, X=<9	Red cuts can further increase efficiency (and also program compactness) by letting us make some guards completely implicit!

min(X,Y,Y) :- X>=Y, ! . min(X,Y,X). % X>Y	But, be very careful when combining guards with pattern matching in the predicate head: e.g. try running ?- min(5,3,5).

Case Statements / If-then-else

```
% mutually exclusive cases built using( If -> Then ; Else )

diagnosis(Patient,Condition):-
      temperature(Patient,T),
      ( T=<37      -> blood_pressure(Patient,Condition)
      ; T <38      -> Condition=ok
      ; otherwise -> diagnose_fever(Patient,Condition)
      ).
```

Acts as if defined by:

```
(If -> Then; _Else) :- If, !, Then.
(If -> _Then; Else) :- !, Else.
(If -> Then) :- If, !, Then.
```

Note that only one solution is tried for If

Note that (If -> Then) is like (If -> Then ; fail)
(so the whole construct fails if the If fails)

```
% equivalent but ugly version with cuts

diagnosis(Patient,Condition):-
       temperature(Patient,T),
       T=<37,!,
       blood_pressure(Patient,Condition).

diagnosis(Patient,ok):-
       temperature(Patient,T),
       T<38,!.

diagnosis(Patient,Condition):-
       diagnose_fever(Patient,Condition).
```

Hacking the Grid Size

- You can temporarily reduce the grid size to speed up debugging (a lot!)
- …/ailp/library/game_predicates.pl line 52 (spiral) and line 53 (search)

```
51  internal_grid_size(X) :-
52    ( part_module(0) -> X = 10
53    ; part_module(101) -> X = 10
54    ; otherwise       -> X = 20).
55
```

Forall

- forall/2 checks that for all successful instances of a first goal, a second goal succeeds

```
?- forall(member(X,[1, 2, 3, 4, 5]), X>0).
true
```

- This meta-predicate behaves as if defined by:

```
forall( Condition , Action) :-
    \+ (Condition, \+ Action).
```

- For example, the following checks whether all elements of a change-list have the same source and destination banks

```
?- forall(member(X:Y:Z, [man:0:1,fox:0:1,hen:0:1]),(Y=0,Z=1)).
true

?- forall(member(X:Y:Z, [man:0:1,fox:0:1,hen:1:0]),(Y=0,Z=1)).
false
```

Tracing NAF

SWISH File Edit Examples Help 317 users online Search

lpn-c3-s4-q4

```
90
91
92 female(lisa).
93
94 male(peter).
95
96 male(oliver).
97
98 married(peter,lisa).
99
100 bachelor(X) :- male(X), \+married(X,_).
101            %  male(X), (married(X,_)->fail;true).
102
103 surprise(X) :- \+ married(X,_), male(X).
104
105
106
```

trace, (bachelor(X)).

```
Call: bachelor(_5198)
  Call: male(_1430)
  Exit: male(peter)
  Call: married(peter,_1772)
  Exit: married(peter,lisa)
  Redo: male(_1430)
  Exit: male(oliver)
  Call: married(oliver,_1772)
  Fail: married(oliver,_1772)
Redo: bachelor(oliver)
Exit: bachelor(oliver)
X = oliver
```

?- trace, (bachelor(X)).

Examples History Solutions table results Run!

Thank you