CSE 3401 - SWI- Prolog: Getting Started

Updated on January 10, 2013

Homepage, documentation, downloads

SWI Prolog Homepage: http://www.swi-prolog.org/
Documentation: http://www.swi-prolog.org/pldoc/index.html
Installation: http://www.swi-prolog.org/download/stable

Running SWI Prolog in Prism Labs

Type `pl` or `swipl` to run SWI-Prolog.

Getting and Setting the working directory

Get: ?- `working_directory(CWD, CWD).`
Set: ?- `working_directory(_,NewCWD).`

Loading Files

To load a **single file**, for example: family.pl, use

```
?- [‘family.pl’].
```

You may also specify the path, for example:

```
?-[‘C:/PrologFiles/Tutorials/family.pl’]
```

Alternatively use,

```
?- consult(‘family.pl’).
```

It is also possible to load **multiple files** with the following command:

```
?- [‘family1.pl’, ‘familyN.pl’].
```

Alternatively, you may create a single file which you load at the prompt (e.g. myfile1.pl), and which in turn loads all the other files that is required for your application (e.g. myfile2.pl, myfile3.pl). Simply add the following line in the beginning of myfile1.pl:

```
:- ensure_loaded('myfile2.pl').
:- ensure_loaded('myfile3.pl').
```

Executing queries (family.pl)

```
?- male(paul).
True.

?-male(X).
X=john
```

To get additional answers, enter a semicolon (`;`)

```
?-male(X).
X=john;

X=paul;
False.

?- male(_).
True.

?-parent(X,Y).
X= john, Y=paul;
X=mary, Y=paul;
X=john, Y=lisa;
False.
```

Exiting from SWI-Prolog

Type **`halt.`** to exit from SWI-Prolog.

```
?- halt.
```

Alternatively, you may use (CTRL-D) to exit prolog.

Getting Help

`help(+What)` Shows a specific part of the manual., which is one of:

<Name>/<Arity>	Provides help on the specified predicate
<Name>	Provides help on the named predicate with any arity
<Section>	Displays a specified section. Section numbers are separated by dash: 2-3 refers to section 2.3 of the manual. Section numbers are obtained using apropos/1.

Examples:

`?- help(assert).`	Provides help on predicate assert
`?- help(3-5).`	Displays section 3.5 of the manual

`apropos(+Pattern)`
Display all predicates, functions and sections which contain Pattern in their name or summary description.

Example:
`?- apropos(file).` Displays predicates, functions and sections which have `file' (or `File', etc.) in their summary description.

explain(+ToExplain)
Give an explanation of the given 'object'. The argument may be any Prolog data object. If the argument is an atom, a term of the form Name/Arity or a term of the form Module:Name/Arity, explain/1 describes the predicate as well as possible references to it.

Debugging

Chapter 8 of (C & M) covers debugging and common errors in detail. This section provides a brief overview of some of the available options for debugging Prolog programs.

trace

The **`trace`** sets Prolog into a debugging mode (**`notrace`** stops the tracer). The execution of Prolog programs are described in terms of 4 kinds of events that occur:

`Call:` enter the procedure
`Exit:` exit successfully with bindings for variable
`Fail:` exit unsuccessfully
`Redo:` look for an alternative solution

Example:
```
? - trace.
[trace] 6 ?- parentOf(X,Y).
 * Call: (6) parentOf(_G2070, _G2071) ? creep(Press enter to continue to next line)
   Call: (7) fatherOf(_G2070, _G2071) ? creep
   Exit: (7) fatherOf(john, paul) ? creep
 * Exit: (6) parentOf(john, paul) ? creep
X = john,
Y = paul ;
   Redo: (7) fatherOf(_G2070, _G2071) ? creep
   Exit: (7) fatherOf(mary, paul) ? creep
 * Exit: (6) parentOf(mary, paul) ? creep
X = mary,
Y = paul ;
 * Redo: (6) parentOf(_G2070, _G2071) ? creep
   Call: (7) motherOf(_G2070, _G2071) ? creep
   Exit: (7) motherOf(john, lisa) ? creep
 * Exit: (6) parentOf(john, lisa) ? creep
X = john,
Y = lisa.
```

spy

spy allows you to watch exactly how Prolog works to satisfy a goal in your program. You can add a spypoint to a predicate by specifying the predicate's name and arity, like this:

```
?- spy(parentOf).
```

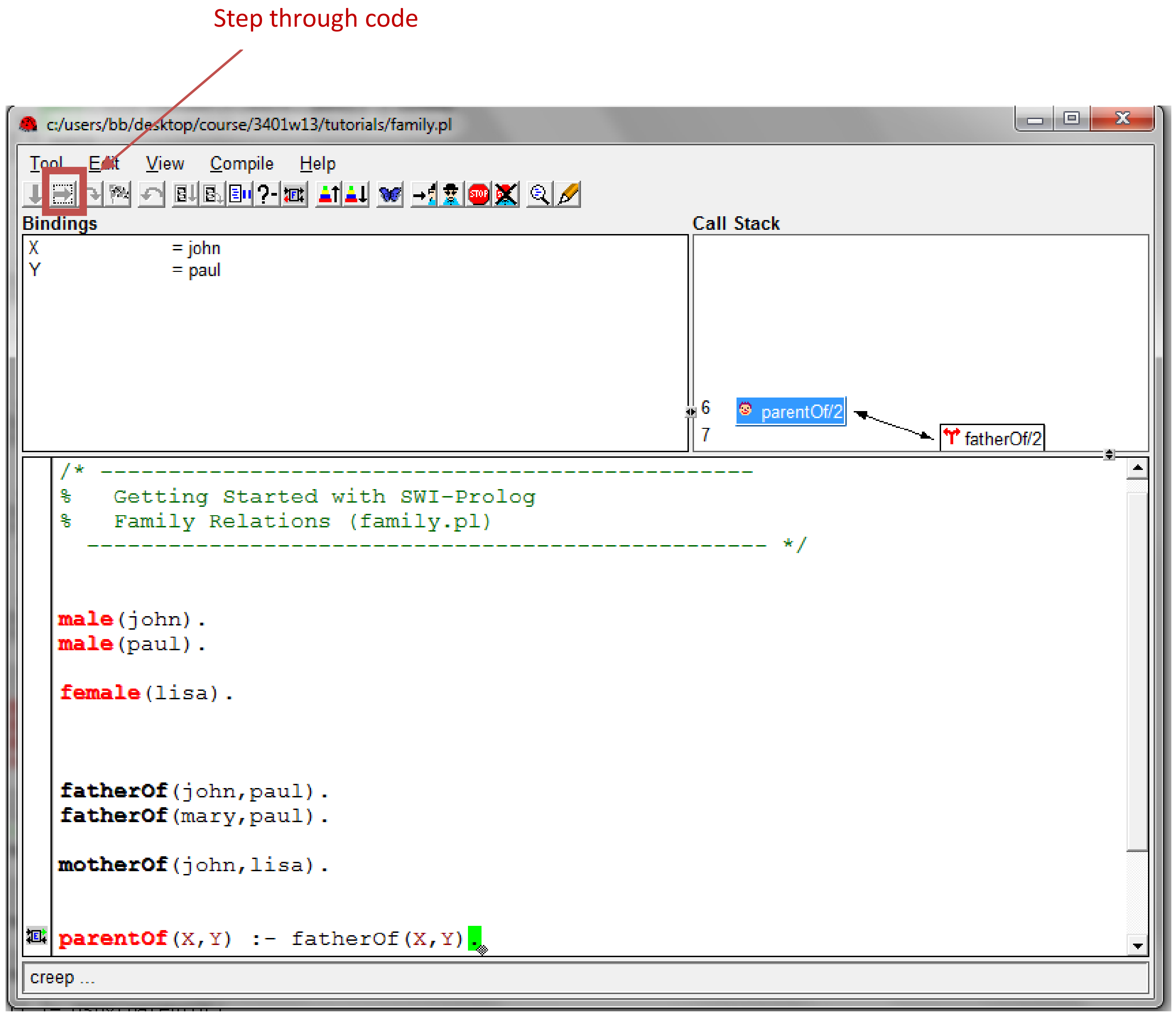
The predicate **nospy(parentOf)** turns off a spypoint.

GUI Debugging

GUI Debugging is also available in Prolog: http://www.swi-prolog.org/gtrace.html .
The procedure is the same, except that instead of trace or spy, you could use gspy or gtrace. For example:

```
?- gtrace, parentOf(X,Y).
```

Figure 1.1 illustrates the window after entering this command.



```
X                = john
Y                = paul
```

```
/* ------------------------------------------------
%   Getting Started with SWI-Prolog
%   Family Relations (family.pl)
  -------------------------------------------------- */


male(john).
male(paul).

female(lisa).


fatherOf(john,paul).
fatherOf(mary,paul).

motherOf(john,lisa).

parentOf(X,Y) :- fatherOf(X,Y).
```

Figure 1.1: GUI Debugging in Prolog

Coding Guidelines

- Comment your code. A % indicates that the rest of the line (up to the end of line) is a comment. Also, everything between /* and */ is considered a comment.
- Indent the code appropriately
- Choose meaningful names for your predicates
- Avoid Singleton variables

  A singleton variable is a variable that only occurs once in the arguments of the head or the body of a predicate and is therefore useless. For example, in the following statements, Y and W are all singleton variables:

  ```
  p(X,Y) :- q(X).
  s(X) :- t(X,W).
  ```

  To make the warnings disappear, use an underscore (_), which stands for an anonymous (new) variable (each time it occurs).

  ```
  p(X,_) :- q(X).
  s(X) :- t(X,_).
  ```

- Keep your code as compact as possible, for example, use
  `equal(X,X)` instead of **`equal(X,Y):-X=Y`**

- Avoid extensive use of semicolon (;) in rules with disjunctions, as it makes debugging harder.