Imperial College London – Department of Computing
MSc in Computing Science
Module: Logic & AI Programming
Sicstus Prolog Introduction in Linux
October 2015

Objectives

The objective of this document is to get you started with using the programming language Prolog, and the Sicstus Prolog environment in Linux on the lab machines. It is assumed that you have a basic understanding of using a Linux command line (terminal) interface to navigate the file system, create new directories etc. Please ask one of the tutorial helpers if you need assistance with this. Information about completing common tasks using Linux can be found on-line in the CSG (Computing Support Group) web pages (http://www.doc.ic.ac.uk/csg/guides/).

I. Introduction to Sicstus Prolog in Linux

First we will see how to run Sicstus Prolog and enter queries.

- To start the Sicstus Prolog interpreter, open a terminal window and enter the command
  ```
  sicstus
  ```
  The prompt | ?- indicates that the environment is ready to receive commands from you, such as loading a program or executing a query.

- Enter the following query at the prompt
  ```
  write('Hello world.').
  ```
  Do not forget the '.' at the end or Sicstus will just start a new line and wait for you to keep typing. Every query and clause in Prolog must finish with a '.'.

- Sicstus should evaluate the query as soon as you hit Enter. In this case, the built in predicate write/1 is executed. This predicate is 'extra-logical', meaning it has been added to Prolog to turn it into a more useful programming language, but has no logical effect on a program. That is to say write/1 always succeeds, and prints out its argument. You will also see that Sicstus outputs a 'yes' to confirm that the query was successful, and evaluated to true. For comparison, enter the query
  ```
  number(a).
  ```
  which should confirm that 'a' is not a number.

Sicstus has many built in predicates like write and number, but little can be achieved using these alone. To solve problems you must write rules defining the concepts in those problems and put them into a program.

Entering programs interactively

Prolog programs can be entered interactively or loaded from a file.

- To enter a program interactively using the interpreter first type this query
  ```
  consult(user).
  ```
  The ?- disappears from the prompt to show that what is typed now will not be interpreted as a query but will be ‘asserted’ as new clauses.

- Enter the following rule and facts.

  ```
  needs(C, oil) :- needs(C, cars).
  needs(britain, cars).
  needs(japan, cars).
  ```

  followed by end_of_file. to exit back to query entry mode.

- Now execute the query

  ```
  needs(X, oil).
  ```

  to see, according to the program entered, which countries need oil. The interpreter should respond with

  ```
  X = britain ?
  ```

  to show that the query evaluates to true when X has the value britain.

- The ? after the answer is prompting you to decide what to do next. Type ‘h’ to see a list of options. Now type ‘;’ to tell Sicstus to continue executing the query by backtracking to find more answers. Keep typing ‘;’ until Sicstus responds with ‘no’.

When things go wrong

- Enter the following queries and commands

  ```
  consult(user).
  sells(C,X):-sells(C,Y).
  end_of_file.

  Query ?- sells(britain,shoes).
  ```

  Prolog is now in an infinite loop! If this happens Ctrl-C will interrupt execution. Typing ‘h’ at this point will list available options. The simplest of these is ‘a’ for abort.

Loading programs from a file

That was probably the last time you will want to enter a program interactively. It is much more sensible to write your program using a text editor and save it to a file.

- Download the program trade.pl from CATE and save it to a suitable folder within your home directory. If Sicstus is still running, exit using ‘halt.’ or Ctrl-D. Change the working directory of your terminal to the one in which you saved the trade program, using cd.

- Restart sicstus and type this query to load (or ‘consult’) the trade program

  ```
  consult(trade).
  ```

  This will look in the current directory for a program in a file named ‘trade.pl’. If the name of your file has spaces, dots, or other characters then you will need to enclose it in single quotes: [‘myfolder/has my other file.pl’]..

- Consulting a program loads it into memory to be interpreted by Sicstus. The alternative is to compile the program as it is loaded, with one of

  ```
  compile(trade).
  [trade].
  ```

  or by starting Sicstus with the -l option: sicstus -l trade. Compiled code runs more quickly than interpreted code, but you cannot debug compiled code in as much detail as interpreted code.

Debugging

- As always, prevention is better than cure, so if you initially find Prolog a strange experience, don’t let this distract you from the fact that it is just programming. Use good programming practice — solve the problem first then code; code incrementally and test each procedure fully before moving on and using it — and you will at least detect bugs quickly rather than burying them deep in the program.

- Sicstus has a built in debugger which allows you to step through, and inspect, each goal executed as a program runs. To invoke the debugger enter the query

  ```
  trace.
  ```

  If you now type a normal query (try something out of the trade program) the interpreter will stop after every call to a new predicate. Press Enter to move it on to the next one.

- During tracing, to skip the execution of a subquery and go straight to where it exits, type s before pressing Enter.

- During tracing, to zip to the end of the current computation type z before pressing Enter.

- At the top level, use ‘notrace.’ to turn the debugger off before executing the next query.

- Tracing can be very slow. Setting ‘spypoints’ (= breakpoints in gdb) speeds things up if you are only interested in calls to particular predicates. For example,

  ```
  spy(sells/3).
  ```

  adds a spy point for sells, where the ‘3’ is the number of arguments the predicate has. Spypoints can also be set and removed during tracing by typing + and ‘- when a call to the appropriate predicate appears in the trace. To skip intermediate calls and zip to the next one with a spy point attached type z during tracing.

- Even with spypoints set, tracing can still be slow and frustrating once programs become even mildly complex. Often, a better method is to temporarily add write statements to your program to print out the values of variables as it executes. Then you can simply run the program and inspect the output for evidence of what is going wrong.

II. Prolog Tutorials

You are now ready to proceed with the tutorial questions. You can use whichever text editor you like. The graphical editors gedit and kate both have Prolog syntax highlighting, and emacs has a Prolog mode. Remember to save and reconsult the file after each change you make!

III. Some Notes

- The Department of Computing has installed Sicstus Prolog 4 on all lab machines, running under Linux and Windows. The assessed lab exercises will be marked under Linux, so it is essential that you ensure your submissions run properly on the lab Linux installation. You might need to use the **fromdos** utility for converting source files if you develop your programs using the Windows implementation.

- Furthermore, you will be sitting your Prolog driving test in the labs using Linux, so you need to get used to it!

- Sicstus Prolog is available for download from

  https://www.doc.ic.ac.uk/csg-old/software/sicstus/

  under a DoC student licence agreement if you would like to install it on your own computers. You will find the necessary keys and passwords in the readme file for the particular version you select. The current lab version is Sicstus 4.3.2. If you install on a Linux machine then you are also advised to look into the rlwrap program.

- The Sicstus documentation is available at http://www.sics.se/sicstus/docs/latest4/html/sicstus.html

or type sicstus-manual at the command line for a pdf version.

- There are several other Prolog implementations besides Sicstus. A popular and stable alternative, preferred by some users, is SWI-Prolog. It is free, and can be downloaded from www.swi-prolog.org if you would like to install it on your own computers. Please note: there are differences in the Sicstus and SWI-Prolog implementations (e.g. the order of arguments for some built-in predicates). Since all assessed work is marked (automatically) using Sicstus on Linux, you must ensure you test your program on a lab machine before submitting it.