Tutorial: Getting Started with GNU Prolog on Unix

To do the Prolog assignments, you need to have basic Unix knowledge. If you are not familiar with the Unix commands, you must read text Appendix B.

Enter "general.asu.edu" server using the secure telnet client **SSH** to connect to *general.asu.edu*. You can download a personal version of SSH program from

MyASU → MyApps: Search SSH to download.

Make a new directory by command: *mkdir cse240prolog* and put all your Prolog files and programs in this directory. You may create subdirectories in this directory.

Enter the directory by using Change Directory command: *cd cse240prolog*

Please note that you can use Putty or SSH to connect to the general server from anywhere at any time, that is, you can do the assignment at home. You are required to do the assignment using GNU Prolog on Unix and we will grade your assignment using GNU Prolog on Unix. However, for some reason you want to have a window version of Prolog, you can follow the web links given in textbook section B.4.2 to download a Windows-based Prolog environment. I tested a few programs on SWI-Prolog Version 5.0.9. SWI-Prolog seems to be compatible with GNU Prolog. However, I cannot guarantee that all programs that work in SWI-Prolog will work on GNU Prolog on Unix. It is your responsibility to make sure your program works on GNU Prolog before you submit. Windows-based Prolog environments will not be supported by the TAs and graders. It is your responsibility to read the manual if you want to use them.

To write a GNU Prolog program on Unix, you can either use a Unix editor, e.g., *pico* or *vi* (*pico* is much more convenient) or upload (e.g. using SSH or Putty) a pre-edited file into your Unix directory *cse240prolog*. The name of a Prolog program should have an extension *.pl*

Note, when submit your assignment, you cannot directly submit a file with .pl format, the blackboard may not be able to handle .pl file correctly!

To compile a Prolog program, say myprologprog.pl, type 'gplc myprologprog.pl' at the operating system prompt *general* >

```
general > gplc myprologprog.pl
```

The compiler will generate the machine code program stored under the name `myprologprog`.

To start GNU Prolog, type 'gprolog' at the operating system prompt *general* >

```
general > gprolog
```

To execute a program that has been compiled (machine code of a Prolog program), e.g., `myprologprog`, type

```
|?- [myprologprog].
```

where, `|?-` is the GNU Prolog prompt and you don't enter it. Don't forget the **period** at the end.

To exit from GNU Prolog, type your end-of-file character at the main Prolog prompt:

```
| ?- Ctrl-d
```

You can find a complete set of GNU Prolog commands at http://gnu-prolog.inria.fr/manual/index.html.

1. Try following simple programs (build-in functions). Don't forget the **period** at the end of the statement.

```
| ?- write(hello). /* hello is a constant */
| ?- write(Hello). /* Hello is a variable. Its address will be displayed*/
```

```
|?- write('hello world'). /* a string is printed */
|?- read(Y), write('The variable entered is '), write(Y), nl. /* nl prints a newline */
|?- X is 2+2.
|?- Y is 5*8.
|?- Y is 2**10.
|?- length([a, b, x, y, 2, 45, z], L).
|?- append([a, b, c, d], [4, 6, 8], LL).
|?- append(X,Y,[a,b,c]). Then, type ";" to obtain all possible answers.
|?- X is [1 | [2 | [3 | []]]], write(X). Explain the output.
```

2. Given a Prolog program that stores the weather conditions of some cities:

/* Database for weather. It consists of facts rules. In this example, the rule will cause a number of goals to be called */

```
/* Facts */
weather(phoenix, spring, hot).
weather(phoenix, summer, hot).
weather(phoenix, fall, hot).
weather(phoenix, winter, warm).
weather(wellington, spring, warm).
weather(wellington, summer, warm).
weather(wellington, fall, hot).
weather(wellington, winter, cold).
weather(toronto, spring, cold).
weather(toronto, summer, hot).
weather(toronto, fall, cold).
weather(toronto, winter, cold).
/* Rules */
warmer(C1, C2) :-
weather(C1, spring, hot), weather(C2, spring, warm).
warmer(C1, C2) :-
weather(C1, spring, hot), weather(C2, spring, cold).
/* The following rule is in fact a compound question. It will cause a number of goals (questions) to be called. */
/* It can be considered as the "main" program. Please note, since the questions are connected by "and" */
/* relationship, it stops if a "no" answer is given to any single question. */ Page 3 of 4
/* You could use the "or" relationship to connect questions. The compound question will stop if a "yes" */
/* answer is found. */
weatherquestions :-
warmer(phoenix, X),
write('Phoenix is warmer than '), write(X), nl,
weather(City1, fall , hot),
write('City1 = '), write(City1), nl,
```

```
weather(City2, _ , hot),
write('City2 = '), write(City2), nl,
weather(_, Season , warm),
write('Season = '), write(Season), nl,
weather(C1, summer , hot),
weather(C1, fall , hot),
write('C1 = '), write(C1), nl,
weather(C2, spring, warm),
weather(C2, fall , warm),
write('C2 = '), write(C2), nl, nl.
```

2.1 Enter the program using *pico* editor and save the file as *weather.pl*;

You may enter the program on your PC and upload the program into your cse240prolog directory in *general* server.

2.2 Compile the program by the prolog command:

```
general > gplc weather.pl
```

Make sure you have changed directory (*cd*) into the cse240prolog directory before you call the compile.

2.3 Enter GNU Prolog by calling `gprolog` and then execute the program `weather` by type `[weather].`, and then ask following questions:

```
| ?- warmer(phoenix, X).
| ?- weather(City, fall , hot).
| ?- weather(City, _ , hot). /* To see more answers, enter ";" after each answer. */
| ?- weather(_, Season , warm).
| ?- weatherquestions. /* this will call all questions in the rule. */
```

2.4 Change some of the "and" operator "," to the "or" operator ";" in the weatherquestions and observe what difference is made in the output.

2.5 Manually trace the execution of the goal:

```
| ?- warmer(phoenix, X).
```

FQA

1. The read(X) does not take my input.

   Input a period (.) at the end of your input data.

2. What are the differences between the output true, false, yes, and no?

GNU prolog will output a

- "true", if a match is found, and the search has not completed. Type a ; the search will continue
- "yes" if a match is found in this period (after you have typed a ;), and the search is completed.
- "no", if no match is found in this period (after you have typed a ;), and the search is completed.