Com Sci 221 — Programming Languages

Recording Your CS221 Work on Scheme, ML, Prolog Interpreters

(by Steph Bailey, revised by Mike O'Donnell, Dec. 1993)

For assignments done with interpreters (the Scheme, ML, and Prolog processors used in this course are interpreters), the best presentation of the computations is usually a *typescript* containing a listing of your program, a listing of your test file, and all the output generated by the execution of your test file (see below on how to create a typescript). Once logged in, you will create your source program with the text editor of your choice. If you don't know how to use a Unix text editor, type: `emacs` at the shell prompt, and then type the back-space key followed by `t`, and this will start you on a tutorial about how to use emacs, a popular text editor.

After you have created and debugged your program (more on that below), you must create the *typescript* mentioned above. Before you can create a typescript, you must create a testing file for your program which contains *exactly what you would type to ML to run the test cases for your program.* The reason why you should use a test file, rather than just typing the test cases directly in at the keyboard is that using the *delete* key (a very common occurrence with computers) causes the transcript file to become a horrible, illisible mess. If the commands used to create your typescript come from a file, this problem won't arise.

The test file should cause your program to be read (compiled) in, and run exciting and illustrative test cases, thus really impressing me with how well your program performs. The test file should also contain occasional *output* statements which print a running commentary on what is happening in your test cases, since the contents of the test file are not printed as they are executed. Otherwise I may have a difficult time figuring out which commands produced what output, and I will get annoyed.

As an example, I'll pretend that I'm handing in an ML program called `prop.sml`. My test data file is called `mltestdata`.

```
(* Comments are enclosed in these things, but they don't print out *)
use "prop.sml";   (* Load the program *)
output std_out "\n*** finding cnf(OR(AND(VAR \"p\",VAR \"q\"), VAR \"r\"));\n";
cnf(OR(AND(VAR "p",VAR "q"), VAR "r"));  (* Boring test case! *)
(* Now set the magic system variable to enable printing the whole list *)
System.Control.Print.printDepth := 10;    (* This can be useful *)
output std_out "\n*** finding cnf(OR(AND(VAR \"p\",VAR \"q\"), VAR \"r\"));\n";
cnf(OR(AND(VAR "p",VAR "q"), VAR "r"));  (* I wouldn't get a good grade
                                            for these tests *)
```

After you have created a test data file, you may create the script with the following sequence of commands:

- `script` at the shell prompt. This will start a new shell where everything which appears on your terminal will be recorded.
- `cat progfile` where progfile is the name of your program file (repeat this command several times if you have several source files).
- `cat testfile` where testfile is the name of your testing file.
- `ml <testfile` this invokes ML to execute all the statements that you have in your testfile. It will complete all the commands and then exit back to the shell
- `exit` this exits the script shell and closes the typescript file.