TPL Final Exam

M. Donica

May 4, 2010

1 The Problem

Two problems were given for the final examination. The first, was to create a finite state machine and the C code for a given regular expression. The second was to create flex and bison files to parse bash command lines. We were told not to worry about capturing every keyword, but to spend more time figuring out how to parse things like pipes and redirects.

2 The First Solution

The trick to this problem was to turn a nondeterministic finite state machine into a deterministic one. Normally, this is done through the method of subset construction, but this one was small enough that I honestly just eyeballed it. I then wrote C based on my finite state diagram and tested it to make sure it was correct.

3 The Second Solution

To accomidate pipes and redirects, I created two rules. One called commandlist, which can either be empty, or a command and then a commandlist, and the rule command which actually accomidates the command and any piping that might be done. A command id either made up of one of a few bash commands, or a command a pipe and a command, or a command, a redirect and another command (or identifier). The command rule itsself cannot be empty. I did that because if it were to be allowed to be empty then something like "_____" would be okay by the rules and woould not throw an error when fed to the parser. As nothing piped with nothing ad naseum isn't even a proposition which makes sense, I though it would be a good idea to avoid, anything which could parse that in a legitimate sense.

One of my statements in the command rule handles functions. A function, is made up of a declaration, a name, an open paran, a commandlist and then a close paran. I did this because it is valid to have an empty function.

Within my .l file, a function's name is defined as being one or more letters, and then an open parenthesis and then a close parenthesis. An identifier is then defined as one or more letters, followed by any number of digits. I checked for them in that order to ensure that function names would not be mistakenly parsed as identifiers.

4 Issues I Faced

I had no issues with the first problem. I drew the finite state machine and then wrote the code for it in about an hour. Other than a missing semicolon at the end of one line, it compiled correctly and ran correctly.

In the second problem, I faced a reduce/reduce error brought on by an out of place instance of the empty rule. As of right now, my code has three shift/reduce errors attached to it. I suspect they arise where I am trying to parse pipes and redirects. If I had more time, I suspect this would be fairly easy to solve.