Wool Semantic Analysis

1 Overview

This second programming assignment requires more effort than the previous one. We are done with ANTLR at this point. We know that the program is now syntactically correct, or as much as we have checked through the parsing phase. Now we need to make sure that it is semantically correct; in the case of Wool, this means that we are able to assign types to every expression.

You should complete this assignment in four steps (four if you create an AST):

1. **Implement symbol tables.** I provide you with a symbol table implementation that you can use as it is, modify it for your needs, or write your own. This was my first cut at the symbol table for Cool-W so it might need some fixes, but it works in general.

2. **Create the Abstract Syntax Tree.** *This is an optional step.*

3. Complete semantic analysis. Depending upon your design, you may walk the tree several times in order to complete the process. The goal to make sure that every use of an identifier has a binding and that the type checking rules defined for Wool are followed.

4. implement your Woolc tool and test it.

The same constraints and rules about how you are to work are the same as for the previous programming assignment. I again urge you to form study groups, especially to review the code that I provide to you. Also, coming to office hours with questions that we can discuss and clarify any issues should be useful.

2 Expected results

For this assignment, when an error occurs during any pass over the tree, you should print a meaningful error message the standard error stream and continue to the end of the pass. At the end of the pass, if an error has occurred, you should throw a `WoolException` and quit.

3 What to submit and grading information

When you are ready to submit your assignment, you should submit two things. You can submit separate files or bundle them into a single zipped archive. The files to submit are:

- Your Eclipse project for the compiler. Simply export it to a zipped archive. I suggest that once you export it, you use a different workspace and make sure that you are able to import it and have it build and work.

- An executable JAR file with your Woolc ready to run. This should have all of the options except for -o implemented. I will add one or more videos, with starting code to the Wool project module to get you started.

This assignment is worth 200 points. 160 points will be awarded for correctness. For each type of error that shows up in my testing, you will have 10 points deducted from this 160. This means that if, for example, four of my tests fail, but I can determine that two of them are due to the same root cause, then you would receive 130 points for this part, -20 for the two unique failures and -10 for the two failing for the same reason.

The last 40 points will be awarded based upon the quality of your error messages and code organization and readability.

You may make multiple submissions up to the deadline. The last submission you make before the deadline will be the one graded. Each submission supersedes the previous one. Remember that late submissions are not accepted. If you only have one submission and it's late, you will receive a grade of zero. DO NOT wait until the last minute to submit your assignment. Technical difficulties are not an excuse for late submissions. The chance of what you might submit ten minutes before the deadline being so much worse than what you might submit at the deadline is minute and the potential negative consequences are not worth taking the chance. No grading will be done until the deadline has passed.

If you have questions about the starting code or anything else about this assignment, you can add to the **#semantic_analysis** discussion in Slack.