

# Project 3: Semantic Analysis

CSC 4351, Spring 2009

Due: 11 March 2009

Add type checking to your compiler as described on page 127. Implement both parts (a) and (b) described in the book. As before, the description of the type rules and semantic rules needs to be extracted from the *Tiger Language Reference Manual* in Appendix A.

## Environment and Support Files

For working on this lab, change the environment variable `PROG` in your `.profile` file to `chap5`. As usual, you can find support code in `/${TIGER}/${PROG}`.

The `Absyn` package contains a few new versions of abstract syntax tree classes. Use `diff` to see the differences with the old versions. The `Translate` package contains the empty abstract class `Exp.java`. This is used as place holder for future parts of the compiler. The `Types` package contains classes representing the data types of the Tiger language.

The main files for you to work with are the files in package `Semant`. In particular, you should modify `Semant/Semant.java`. You may also wish to add additional classes to avoid cluttering `Semant.java`. E.g., the precompiled type checker contains classes `LoopVarEntry` and `LoopSemant`.

## Compilation

Since we no longer use any non-Java source files, it's easiest to manually use `'javac -g */*.java'` for compiling.

For running your type checker on an input file `test.tig`, execute

```
java Semant.Main test.tig
```

in your working directory. The result is a printed representation of the abstract syntax tree with type annotations. You can still run the parser without semantic analysis using

```
java Parse.Main test.tig
```

## Submission

For this lab, you will mainly modify file `Semant/Semant.java` but you might end up adding additional classes. To make sure, you don't forget to submit anything, submit all the Java files in package `Semant`, or the entire directory structure:

```
cd prog3; rm */*.class; ~cs4351_bau/bin/p_copy 3
```

In the `README` file, provide any information that will help the grader to give you partial credit. Explain what's implemented and what's not. Explain important design decisions in your code, e.g., if you add new classes.