CS 3300 Compilers Design : JavaCC / JTB Tutorial

Jahnvi Patel

August 27, 2019

Tutorial Goals

We will cover a few examples on using JavaCC / JTB to solve simple tasks on MiniJava programs.

1. Types of generated visitors and how to modify them.
2. Example 1 : How to find number of assignment statements, print statements, etc. for a MiniJava program?
3. Example 2 : For each class, print signatures for all its member functions.

Writing your first visitor

- Several visitors might be generated like GJDepthFirst, GJVisitor, etc. They differ in the way traversal is done, number of arguments to visitor and return value.
- For our lab purposes, these two visitors shall suffice :
  1. GJDepthFirst : Does DFS on the node tree and takes one argument from Main.
  2. GJNoArguDepthFirst : Also does DFS traversal of node tree. Use it when you do not need to pass any argument to visitor.

Note : Do not run jtb command again without using -w flag else your existing visitors might be overwritten.

Node Sub-Classes : Recap

- `NodeToken` : To get the string value for the token, use `tokenImage`.
- `NodeChoice` : For rules of type `A` $\rightarrow$ `B|C`, `which` tells us the index of the choice (for instance, 0 if B was encountered), `choice` points to the node that was chosen.
- `NodeList` : For a list of nodes like `A+`, `elements` returns an enumerator to traverse through the list, `elementAt` can be used to find node at a particular index, `size` can be used to determine the number of nodes present in a list.

TODO : Look at the syntaxtree files for the various sub-classes of node to see what all functionality they provide.

Counting Statements

Task

Given a MiniJava program, find number of assignment statements, print statements, etc.

- Hint : Use `which` on choice node.
- Write a few programs to see how the traversal works, pass arguments to child nodes and return values to parents.

Function Signature

Task

Given a MiniJava program, print the function signatures for all methods for all classes.

- TODO : Use this idea to create a symbol table for each method.

Assignment 2 is up!

- Use a symbol table for each class to store variable names and their types, return types of its functions, arguments that its methods take, a link to parent class and also a list of symbol tables for each of the methods.
- The challenge here lies in the fact that the some symbol used in current scope could have been declared in one of the parent classes. For a variable, check the method symbol table followed by class symbol table and all the way up to root class until we find it.
- A few examples of checks we need to perform : if condition type matches boolean or the actual parameters match formal parameters.
- Can be solved in two passes : first pass constructs the symbol tables and the second pass performs checks and logs out corresponding errors.