CS3410: Software Engineering Lab
Writing LR(1) parser for JavaCC

Aravind S CS11B033
S Akshayaram CS11B057
R Srinivasan CS11B059
S K Ramnanadan CS11B061
Adit Krishnan CS11B063

March 14, 2014

1 JavaCC

JavaCC is a Java parser generator written in the Java programming language. It produces pure Java code. Both JavaCC and the parsers generated by JavaCC have been run on a variety of Java platforms. JavaCC comes with a bunch of grammars including Java 1.0.2, Java 1.1, and Java 2 as well as a couple of HTML grammars. JavaCC is by far the most popular parser generator used with Java applications. They have had over hundreds of thousands of downloads and estimate number of serious users is many thousands. The main highlights of JavaCC are:

- JavaCC generates top-down (recursive descent) parsers as opposed to bottom-up parsers generated by YACC-like tools i.e it implements LL(k) parser. This allows the use of more general grammars (although left-recursion is disallowed). Top-down parsers have a bunch of other advantages (besides more general grammars) such as being easier to debug, having the ability to parse to any non-terminal in the grammar, and also having the ability to pass values (attributes) both up and down the parse tree during parsing.
- The lexical specifications such as regular expressions, strings, etc. and the grammar specifications (the BNF) are both written together in the same file. It makes grammars easier to read (since it is possible to use regular expressions inline in the grammar specification) and also easier to maintain.
- JavaCC comes with JJTree, an extremely powerful tree building preprocessor.

2 LR(1) Grammar

In computer science, a canonical LR parser or LR(1) parser is an LR(k) parser for k=1, i.e. with a single lookahead terminal. The special attribute of this parser is that all LR(k) parsers with k¿1 can be transformed into a LR(1) parser. It can handle all deterministic context-free languages. In the past this LR(k) parser has been avoided because of its huge memory requirements in favor of less powerful alternatives such as the LALR and the LL(1) parser. The name LR is an acronym. The L means that the parser reads input text in one direction without backing up; that direction is typically Left to right within each line, and top to bottom across the lines of the full input file. (This is true for most parsers.) The R means that the parser produces a reversed Rightmost derivation; it does a bottom-up parse, not a top-down LL parse or ad-hoc parse. LR parsers are deterministic; they produce a single correct parse without guesswork or backtracking, in linear time. This is ideal for computer languages.

3 Our Proposal

We propose to add LR(1) parser for the JavaCC program. We propose to comprehensively read up the existing parser in the JavaCC code and make the appropriate changes so that it supports LR(1) parsing as well.