Notes

Ernest Kirstein

September 12, 2014

Recursive Descent Parsing

The core rules for my parser are built off of Dr. Lewis's notes [1]. A grammar is define as an ordered collection of production rules. My parser uses context-free grammar rules, which are comprised of a 'head' (the single-symbol left hand side of the production rule), and a 'tail' (one or more symbols comprising the right hand side of the production rule).

These grammars may be 'compiled' using the four procedures: factoring, substitution, removing left recursion, and removing useless rules. Let 'decision list' define an ordered list of production rule choices which produces a parse tree. As each of these four procedures produces a weakly equivalent grammar, there exists a mapping for any decision list in a compiled grammar back into a same-terminal-producing decision list in the pre-compiled (parent) grammer. My parser keeps track of these inverse transformation rules as performs it's compilation procedure so that a compiled grammar's decision list can be easily converted to the initial grammar's equavalent decision list.

Take this simple grammar for example:

$$S \to AB \tag{1}$$
$$A \to a \tag{2}$$
$$A \to SA \tag{3}$$
$$B \to b \tag{4}$$
$$B \to SB \tag{5}$$

It compiles into the weakly equivalent grammar:

$$Z \to \epsilon \tag{1}$$
$$B \to b \tag{2}$$
$$S \to aBS' \tag{3}$$
$$S' \to \epsilon \tag{4}$$
$$A \to aZ \tag{5}$$
$$B \to aBS'B \tag{6}$$
$$S' \to aZBS' \tag{7}$$
$$Z \to bS'A \tag{8}$$
$$Z \to aBS'BS'A \tag{9}$$

So when the terminal stream "aabb" is parsed in the compiled grammar to the decision list $[3, 6, 2, 4, 2, 4]$ it can be transformed into the parent-grammar-equivalent decision list: $[1, 2, 5, 1, 2, 4, 4]$.

References

[1] F. D. Lewis. Recursive descent parsing. `http://www.cs.engr.uky.edu/~lewis/essays/compilers/rec-des.html`, 2002.

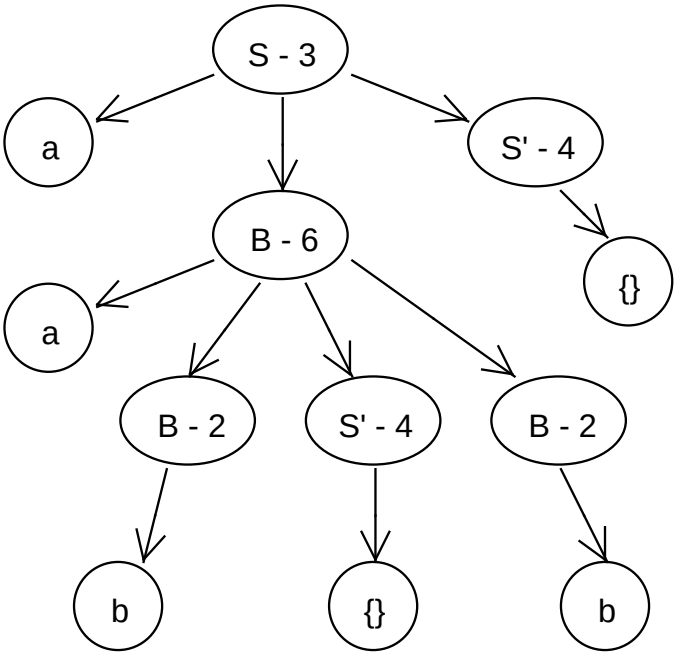

Figure 1: Compiled Grammar Parse Tree

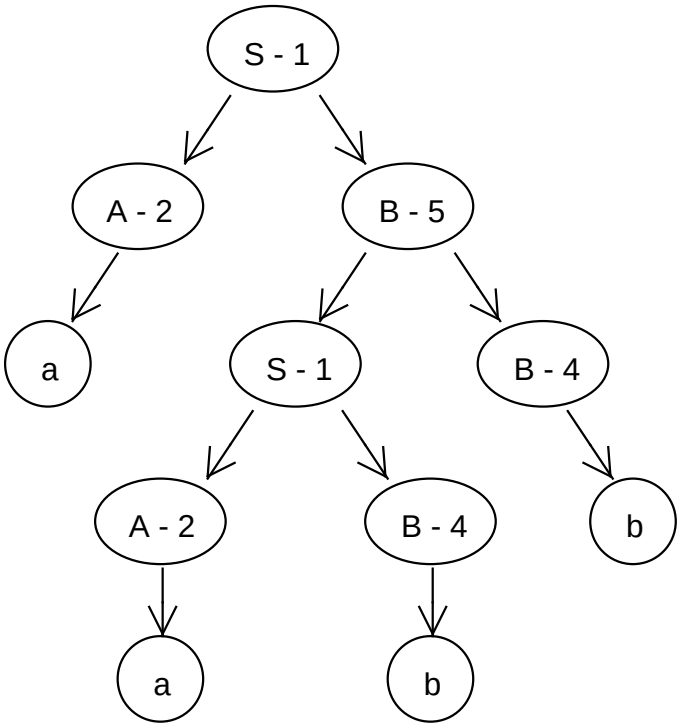

Figure 2: Parent Grammar Parse Tree