Student: Ty Davis

Course: CS 4110

Subject: Programming Assignment 2 - Context Free Grammar

Date: November 5, 2025

Balanced Parentheses Discussion

The grammar for the recursive descent algorithm is shown below:

```
S → ^ | SS | (S) | [S] | S
```

The recursive parser mirrors the grammar above by capturing a pair of parentheses/brackets and calling the interior S, then considering the interior as well. To mirror the `S → SS` portion of the grammar, after finding a pair of parentheses/brackets, we also recursively consider the portion of the original S that occurs after the closing bracket.

The stack algorithm works by keeping track of encountered opening bracket symbols and ensuring that the opposite brackets are encountered in the proper order before the string S ends. It is much more efficient than the recursive approach because it simply iterates over the characters in sequence. The recursive will potentially execute many function calls before getting all the way to the deepest descent.

One key difference between the two approaches, is the context of the function seemingly resets with each descent in the recursive approach. This can be useful if you need to execute some function on each level of the descent because the context thereof is exactly the content inside those brackets. That is not the same in the stack algorithm.