Assignment 1: Lambda Calculus Syntax

CSI 3120-B - Fall 2024

School of Electrical Engineering and Computer Science
University of Ottawa

Course Instructor: Prof. Karim Alghoul

Group 28
Ben (300297574)
Shawhin Niktash (300298189)

Submission Date: Oct. 5, 2024

Problem Statement

The task of this assignemnt is to implement a lambda calculus parser based on specified grammer. The parser has the following requirements:

1. 1. Recognize valid expressions
2. 2. Report a detailed syntax error if an invalid expression is found
3. 3. Handle expressions which can include lambda functions, dot operators, variables, and parenthesis
4. 4. Create a parse tree to visualize the broken down expression

The parser can handle strings which consist of variables, brackets, lambda functions, and dot operators. These are valid characters allowed in an expression:

- Variables consist of upper and lower case letters (a-z, A-Z), and digits (0-9). Variables must start with a letter.
- lambda abstractions use a backslash, followed by a variable, a dot, and another expression example: \x. y z
- round brackets ('(', ')')

The parser is whitespace insensitive.

Parsing Technique

The parsing is done as a recursive function `parse_expr(s)`. A top-down method is used because of its better suited for languages with smaller grammar rules (such as this one), and it's simple design, making it easier to implement, maintain, debug, and read. When the string is passed into the function, there can be one of four possible valid expressions: there can be a lambda expression with a slash, a variable, and an expression, an expression inside of brackets, two expressions separated by a space, or a variable. Once a recursive instance has been fully explored, the tokens it parsed is returned and added to the end of the instance above it.

Link to Top-Down and Bottom-Up parsings of each positive and negative example