1 Introduction

Lilac stands for Leftmost-Innermost LAmbda Calculus. The core notions of lambda calculus are very simple, yet it is still Turing complete. Lilac aims to create lambda calculus as a language. It will follow the same rules as applicative order (leftmost-innermost) lambda calculus. It will implement the integer type, the operations + − ∗ / , and a print functionality.

The motivation for Lilac is to aid the exploration of lambda calculus. Its features, although basic, should improve the usability of Lilac in this exploration.

2 Syntax

The variable names in will be given by the user. \ and . in Lilac corresponds to the abstraction symbols \ and . in lambda calculus. ( and ) are also available to specify the order of operation. The syntax for the variables, abstraction symbols, and parentheses follow the same rules as lambda calculus.

3 $\beta$-reduction

Lilac uses applicative order evaluation strategy for $\beta$-reduction. Reduction steps are not shown, and only what is explicitly printed by the Print functionality will display on the console.

4 Features

4.1 Integer Type

The only data type is the 16-bit integer; all further references to integer will mean 16-bit integer. Four operations are available to integers. Given an integer $n$, Lilac will support abstractions

\[(+n), (-n), (*n), (/n).\]

When applied to another integer $m$, the results will respectively be

\[m + n, m - n, m * n, m / n.\]

Note that integer division truncates towards zero.
4.2 Print

Lilac features two abstractions for printing: \textit{prtc} and \textit{prti}. When applied to an integer \( n \), \textit{prtc} \( n \) will print \( n \) to the console as a character, and \textit{prti} \( n \) will print \( n \) to the console as an integer.

5 Example Code

5.1 Calculator

\texttt{prti((\x.\y. + x y)3 4)}

Output: 7

5.2 Hello World

\texttt{prtc 72 prtc 101 prtc 108 prtc 108 prtc 111 prtc 32}
\texttt{prtc 87 prtc 111 prtc 114 prtc 108 prtc 100}

Output: Hello World