An analysis on the invariance of λ-calculus

Haileselassie Gaspar

May 21, 2025

1 Abstract

The notion of lambda calculus has been a part of computational theory ever since Alan Turing proved that it was an equivalent model to Turing machines, and it has had an significant impact on the field of computability and functional programming.

2 Introduction

3 Lambda Calculus

In order to talk about the invariance of λ-calculus it is first necesary to define some notation that will be used in this paper.

Definition. $M, N, P....$ denote arbitrary λ-terms, $x, y, z...$ denote variables and the set of λ-terms Λ is inductively defined as:

$$\begin{aligned} &\text{Variables: } x \in \Lambda \\ &\text{Abstraction: } M \in \Lambda(\lambda x.M) \in \Lambda \\ &\text{Application: } M, N \in \Lambda(MN) \in \Lambda \end{aligned} \tag{1}$$

Definition. $FV(M)$ is the set of free variables in M and it includes every variable in M not bound by an abstraction.

For further reading on the syntax and axioms of the lambda calculus, refer to -Barendegt book-. It is necesary however to introduce the notion of reduction in the lambda calculus.

Definition. Let **R** be a notion of reduction on Λ. Then **R** induces the binary relations:

$$\begin{aligned} &1.\ \to_R \ \textit{one step R-reduction} \\ &2.\ \to_R^* \ \textit{R-reduction} \\ &3.\ =_R \ \textit{R-equality or R-convertibility} \end{aligned} \tag{2}$$

4 Proof Overview

As stated before the measure employed to analyze the time invariance of lambda calculus, or, said differently, its universality, is the number of transitions in a turing machine. If the implementation introduced in **Beta reduction invariance Paper citation** – is correct, then by means of the Linear Substitution Calculus, it is possible to represent even size-exploding terms in Turing machines. The proof will be dividied into two sections, and this paper will focus on the implementation of the first in Haskell.