An analysis on the Invariance of λ-calculus with respect to Turing machines

Haileselassie Gaspar

May 29, 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

Introduce the idea of computation and the reasoning behind the importance of the invariance thesis and computational equivalence and such.
The three basic models of computation:
- Turing $\rightarrow$ Turing machines
- Church $\rightarrow$ Lambda calculus
- Explain the differences a little
These models were proven to be equivalent by Turing, Kleene, Church and Rosser.
The invariance thesis:
- Every reasonable machine can simulate each other within a polynomial bound on time and constant overhead in space
- Orthodox
- Liberal $\leftarrow$ The one we will be using
We can relate reasonable machines through cost models.

3 Requirements

3.1 Introduction to 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:

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

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

In order to analyze reduction strategies in the lambda calculus, we will introduce the concept of a context. A context is a lambda term with a parameter to be filled. It is defined as:

$$C ::= \langle\cdot\rangle \mid \lambda x.C \mid Ct \mid tC \tag{2}$$

For further reading on the syntax and axioms of the lambda calculus, refer to -Barendegt book-.

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

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

– TODO –

3.1.1 Reduction

- Talk about reduction - Talk about residuals - LO order on reductions

3.1.2 Church Rosser And Standarization

- Diamond property - Is LO CR? - What does it mean to be standard? - Is LO Standard?

3.2 Term Rewriting Systems

- How to analyze the properties of the lambda calculus? - Other TRS and how we use them

4 Proof Overview

As stated before the measure employed to analyze the time invariance of lambda calculus is the number of transitions in a turing machine. By means of the Linear Substitution Calculus, it is possible to represent even size-exploding terms in Turing

machines in polynomial time. It will be shown that by converting the LSC to a NPDA, the lambda calculus is indeed quadraticly bound in time when representing Turing machines.

4.1 High level implementation systems

The purpose of the high level implementation system definition is to provide a rewriting system invariant to lambda calculus. This step is a bridge of sorts in between lambda calculus and turing machines. For this, we need to define this class of rewriting systems, and which properties should they satisfy in order to be invariant to lambda calculus. We want spcifically termintation and polynomial overhead.

$$\rightsquigarrow \text{ terminates iff} \rightsquigarrow_X \text{terminates}$$

Furthermore

$$t \rightsquigarrow_X^k u iff t \rightsquigarrow^h u \downarrow \text{ with } O(h) \in O(k^n) \text{ for some } n \in \mathbb{R}$$

4.1.1 Properties of high level systems

- Normal form - Projection - Trace - Syntactic Bound

4.1.2 Proof of high level properties

- Termination and polynomial overhead for a generic LSC term

4.2 Low level implementation

A high level implementation system is implemented on a turing machine with an overhead in time polynomial to k and the size of the initial term.

4.2.1 Properties of low level systems

- Subterm - Selection

4.2.2 Proof of low level propeties

- Polynomial bound on reductions in LSC strategy.

5 Useful derivations

- What does it mean for a derivation to be useful? - Why do we need useful derivations? - Leftmost Outermost Useful

5.1 Standarization of Useful derivations

- Why do LSC srategies contain the subterm property? - Why does LOU have the subterm property?

6 Comparing LSC terms

- Look into the algorithm to compare them and talk about it - If we can define an equality relation in the LSC, then we can prove basically the same as we can in Lambda calculus.