Invariance in the Lambda Calculus through Explicit Substitutions

Haileselassie Gaspar

Vrije Universiteit Amsterdam

14-06-2025

2025-07-10

Invariance in the Lambda Calculus through Explicit Substitutions

Invariance in the Lambda Calculus through Explicit Substitutions

Haileselassie Gaspar

Vrije Universiteit Amsterdam

14-06-2025

Contents

Invariance in the Lambda Calculus through Explicit Substitutions

2025-07-10

└─Contents

Contents

1. Introduction and Background
2. The Size-Explosion Problem
3. Linear Substitution Calculus
4. High-Level Implementation Systems
5. Examples

Equivalence and Invariance of Models

- The Church-Turing Thesis - Turing, Kleene, Church, Rosser
- The Invariance Thesis - Van Embde Boas

Equivalence and Invariance of Models
- The Church-Turing Thesis - Turing, Kleene, Church, Rosser
- The Invariance Thesis - Van Embde Boas

Turing machines are the foundational measure of computational complexity so when we talk about equivalence and invariance we refer to it WITH RESPECT TO TURING MACHINES. We will study INVARIANCE through a cost model. We will only talk about time invariance in this presentation.

Size exploding Family of λ-terms

$$
\begin{aligned}
t_0 &\equiv yxx \\
t_{n+1} &\equiv (\lambda x.t_n)(yxx)
\end{aligned} \tag{1}
$$

We will refer to the normal form of this terms by r_n. By induction we can see that $t_n \xrightarrow[1]{\beta} (\lambda x.t_{n-2})y(yxx)(yxx) \xrightarrow[n-1]{\beta} r_n$, and that $|r_n| \in O(2^n)$.

Size exploding Family of λ-terms

$t_0 \equiv yxx$
$t_{n+1} \equiv (\lambda x.t_n)(yxx)$ (1)

We will refer to the normal form of this terms by r_n. By induction we can see that $t_n \xrightarrow[1]{\beta} (\lambda x.t_{n-2})y(yxx)(yxx) \xrightarrow[n-1]{\beta} r_n$, and that $|r_n| \in O(2^n)$.

When we talk about the size explosion we refer to it in terms of how a Turing machine would represent this term, which obviously takes 2^n steps since space complexity is a lower bound for time complexity on Turing machines

Syntax and operational semantics

- Syntax

$$
\begin{aligned}
t, u &::= x \mid \lambda_{lsc} x.t \mid tu \mid t[x \leftarrow u] \\
S &::= \langle \cdot \rangle \mid \lambda_{lsc} x.S \mid St \mid tS \mid S[x \leftarrow t] \\
L &::= \langle \cdot \rangle \mid L[x \leftarrow t]
\end{aligned} \tag{2}
$$

- Operational Semantics

$$
\begin{aligned}
L\langle \lambda_{lsc} x.t \rangle u &\to_{dB} L\langle t[x \leftarrow u] \rangle \\
S\langle x \rangle [x \leftarrow u] &\to_{ls} S\langle u \rangle [x \leftarrow u]
\end{aligned} \tag{3}
$$

Syntax and operational semantics

- Syntax

$$
\begin{aligned}
t, u &::= x \mid \lambda_{lsc} x.t \mid tu \mid t[x \leftarrow u] \\
S &::= \langle \cdot \rangle \mid \lambda_{lsc} x.S \mid St \mid tS \mid S[x \leftarrow t] \\
L &::= \langle \cdot \rangle \mid L[x \leftarrow t]
\end{aligned} \tag{2}
$$

- Operational Semantics

$$
\begin{aligned}
L\langle \lambda_{lsc} x.t \rangle u &\to_{dB} L\langle t[x \leftarrow u] \rangle \\
S\langle x \rangle [x \leftarrow u] &\to_{ls} S\langle u \rangle [x \leftarrow u]
\end{aligned} \tag{3}
$$

Unfolding of Shared terms

We introduce the operation $\downarrow$ in order to convert λ_{LSC}-terms to regular λ-terms.

$$t[x \leftarrow u]\downarrow = t\downarrow \{x \leftarrow u\downarrow\} \tag{4}$$

And the contextual unfolding of a term:

$$t \downarrow_{S[x \to u]} = t \downarrow_S \{x \to u \downarrow\} \tag{5}$$

Unfolding of Shared terms

We introduce the operation $\downarrow$ in order to convert λ_{LSC}-terms to regular λ-terms.

$$t[x \leftarrow u]\downarrow = t\downarrow \{x \leftarrow u\downarrow\} \tag{4}$$

And the contextual unfolding of a term:

$$t \downarrow_{S[x \to u]} = t \downarrow_S \{x \to u \downarrow\} \tag{5}$$

Let $\rightsquigarrow$ be a deterministic λ-strategy and $\rightsquigarrow_X$ be a strategy on the LSC. The pair $(\rightsquigarrow, \rightsquigarrow_X)$ is called a high-level implementation system if it has the following properties:

1. Normal Form Equality
2. Projection
3. Trace
4. Syntactic Bound

Let $\rightsquigarrow$ be a deterministic λ-strategy and $\rightsquigarrow_X$ be a strategy on the LSC. The pair $(\rightsquigarrow, \rightsquigarrow_X)$ is called a high-level implementation system if it has the following properties:

1. Normal Form Equality
2. Projection
3. Trace
4. Syntactic Bound

Useful Derivations

An applicative context is defined by $A = S\langle Lt\rangle$.
A useful step is either a dB-step or a ls-step $S\langle x\rangle \to S\langle r\rangle$ so that the unfolding $r\downarrow_S$:

1. Either ontains a β-redex
2. Or is an abstraction and S is an applicative context.

Useful Derivations
An applicative context is defined by $A = S\langle Lt\rangle$.
A useful step is either a dB-step or a ls-step $S\langle x\rangle \to S\langle r\rangle$ so that the unfolding $r\downarrow_S$:
1 Either ontains a β-redex
2 Or is an abstraction and S is an applicative context.

Size exploding Terms revisited

Taking $u = yxx$ for readability:

$$
\begin{aligned}
& t_2 \equiv (\lambda x.(\lambda x.(yxx))(yxx))(yxx) \xrightarrow[2]{\beta} y(yuu)(yuu) \equiv r_2 \\
& t_2 \xrightarrow[2]{dB} (yxx)[x \leftarrow yxx][x \leftarrow yxx]
\end{aligned}
\tag{6}
$$

Size exploding Terms revisited

Taking $u = yxx$ for readability:

$$
\begin{aligned}
& t_2 \equiv (\lambda x.(\lambda x.(yxx))(yxx))(yxx) \xrightarrow[2]{\beta} y(yuu)(yuu) \equiv r_2 \\
& t_2 \xrightarrow[2]{dB} (yxx)[x \leftarrow yxx][x \leftarrow yxx]
\end{aligned}
\tag{6}
$$