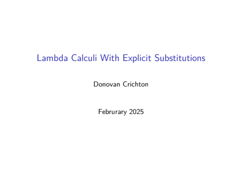
Lambda Calculi With Explicit Substitutions

Donovan Crichton

Februrary 2025

2025-01-30



Lambda Calculi With Explicit Substitutions

Donovan Crichton

Februrary 2025

Preliminaries

- Slides and Examples available at: https://github.com/donovancrichton/Talks
- This talk: BFPG/LambdaCalculiWithExplicitSubstituions

Preliminaries

- Slides and Examples available at: https://github.com/donovancrichton/Talks
- This talk: BFPG/LambdaCalculiWithExplicitSubstituions

Welcome to the talk!

About me

Australian National University

Griffith UNIVERSITY

ASD AUSTRALIAN SIGNALS DIRECTORATE

- PhD Candidate
- Computing Foundations
- School of Computing
- Visiting Scholar
- Trusted Systems Lab
- IIIS
- ASD Co-Lab Scholar

└About me

About me

Australian National University

Griffith UNIVERSITY

ASD AUSTRALIAN SIGNALS DIRECTORATE

- PhD Candidate
- Computing Foundations
- School of Computing
- Visiting Scholar
- Trusted Systems Lab
- IIIS
- ASD Co-Lab Scholar

This is a test defΣ.

A test definition for some concept.

This is a test example.

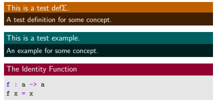
An example for some concept.

The Identity Function

```
f : a -> a
f x = x
```

2025-01-30

Lambda Calculi With Explicit Substitutions



This is a test defΣ.

A test definition for some concept.

This is a test example.

An example for some concept.

The Identity Function

```
f : a -> a
f x = x
```

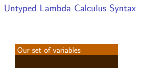
Untyped Lambda Calculus Syntax

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

Untyped Lambda Calculus Syntax

Our set of variables

2025-01-30



Untyped Lambda Calculus Syntax

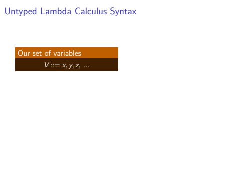
Our set of variables

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

Untyped Lambda Calculus Syntax

Our set of variables

$$V ::= x, y, z, \ \ldots$$



Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \ \ldots$

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

Untyped Lambda Calculus Syntax

Our set of variables

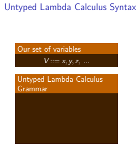
$$V ::= x, y, z, \ \ldots$$

Untyped Lambda Calculus Grammar

2025-01-30

Lambda Calculi With Explicit Substitutions

└Untyped Lambda Calculus Syntax


Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \ \ldots$

Untyped Lambda Calculus Grammar

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

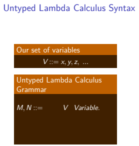
Untyped Lambda Calculus Syntax

Our set of variables

$$V ::= x, y, z, \ \ldots$$

Untyped Lambda Calculus Grammar

$$M, N ::= \qquad V \quad \textit{Variable.}$$



Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \ \ldots$

Untyped Lambda Calculus Grammar

$M, N ::= \quad V \quad \textit{Variable.}$

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

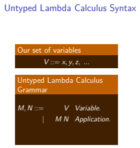
Untyped Lambda Calculus Syntax

Our set of variables

$$V ::= x, y, z, \;\dots$$

Untyped Lambda Calculus Grammar

$$\begin{array}{rcll} M, N ::= & & V & \textit{Variable.} \\ & \mid & M\ N & \textit{Application.} \end{array}$$



Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \dots$

Untyped Lambda Calculus Grammar

$M, N ::= V$ *Variable.* $\mid M\ N$ *Application.*

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

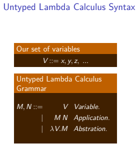
Untyped Lambda Calculus Syntax

Our set of variables

$$V ::= x, y, z, \ \ldots$$

Untyped Lambda Calculus Grammar

$$\begin{array}{rcrl} M, N ::= & & V & \textit{Variable.} \\ & \mid & M\ N & \textit{Application.} \\ & \mid & \lambda V.M & \textit{Abstration.} \end{array}$$



Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \ \ldots$

Untyped Lambda Calculus Grammar

$M, N ::= V$ Variable. $\mid M\ N$ Application. $\mid \lambda V.M$ Abstration.

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

Untyped Lambda Calculus Syntax

Our set of variables

$$V ::= x, y, z, \ \ldots$$

Untyped Lambda Calculus Grammar

$$\begin{array}{lrl} M, N ::= & V & \textit{Variable.} \\ \mid & M\ N & \textit{Application.} \\ \mid & \lambda V.M & \textit{Abstration.} \end{array}$$

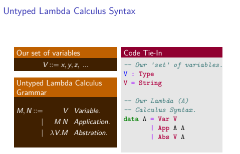
Code Tie-In

```
-- Our 'set' of variables.
V : Type
V = String

-- Our Lambda (Λ)
-- Calculus Syntax.
data Λ = Var V
       | App Λ Λ
       | Abs V Λ
```

2025-01-30

└Untyped Lambda Calculus Syntax



Untyped Lambda Calculus Syntax

Our set of variables

$V ::= x, y, z, \ \ldots$

Untyped Lambda Calculus Grammar

$M, N ::= V$ Variable. $\mid M\ N$ Application. $\mid \lambda V.M$ Abstration.

Code Tie-In

```
-- Our 'set' of variables.
V : Type
V = String

-- Our Lambda (Λ)
-- Calculus Syntax.
data Λ = Var V
       | App Λ Λ
       | Abs V Λ
```

1. A quick refresher on the untyped lambda calculus
2. Smallest turing-complete language.
3. First we need a set of variables.
4. Grammar/Syntax has 3 terms.
5. Looks scary? You can read this already, clearly inspires data declarations in ML languages

Untyped Lambda Calculus Computation

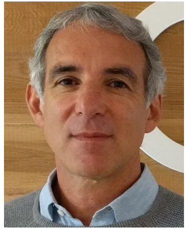
Untyped Lambda Calculus Computation

The Problem

Implementation GapFormal Reasoning

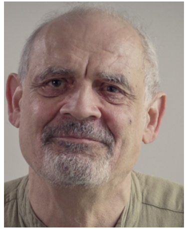
There is usually a non-trivial disconnect between how lambda calculus is presented in mathematics, to how it is implemented in a programming language. Substitution in traditional presentations of lambda calculus is a meta operation, which makes formal reasoning about it difficult.

Explicit Subsitutions (Paper)



Martin Abadi

Luca Cardelli



Pierre-Louis Curien

Jean-Jacques Levy

2025-01-30

Lambda Calculi With Explicit Substitutions

└Explicit Subsitutions (Paper)

References