

How I learnt to love typed λ -calculus

A NewCrafts 2025 Hands-On Lab

Arnaud Bailly Emmanuel Gaillot

2025-11-06

Introduction

An experimental workshop to explore the computational foundation of the universe, and have fun



λ -Calculus Syntax

A term from the λ -Calculus is either:

▶ A *Symbol*: x , y , Foo , $++\dots$

λ -Calculus Syntax

A term from the λ -Calculus is either:

- ▶ A *Symbol*: $x, y, \text{Foo}, ++\dots$
- ▶ An *Abstraction*: $(\text{lam } (x) \ x)$

λ -Calculus Syntax

A term from the λ -Calculus is either:

- ▶ A *Symbol*: $x, y, \text{Foo}, ++\dots$
- ▶ An *Abstraction*: $(\text{lam } (x) \ x)$
- ▶ An *Application*: $(\text{foo } \text{bar})$

λ -Calculus Rules

Terms from the λ -Calculus can be *rewritten* using various rules the two main ones being:

- ▶ α -renaming:

λ -Calculus Rules

Terms from the λ -Calculus can be *rewritten* using various rules the two main ones being:

▶ α -renaming:
 $(\text{lam } (x) \ x) \Longrightarrow (\text{lam } (y) \ y)$

λ -Calculus Rules

Terms from the λ -Calculus can be *rewritten* using various rules the two main ones being:

▶ α -renaming:
 $(\text{lam } (x) \ x) \Longrightarrow (\text{lam } (y) \ y)$

▶ β -reduction:

λ -Calculus Rules

Terms from the λ -Calculus can be *rewritten* using various rules the two main ones being:

- ▶ α -renaming:
 $(\text{lam } (x) \ x) \Rightarrow (\text{lam } (y) \ y)$
- ▶ β -reduction:
 $((\text{lam } (x) \ \text{body}) \ \text{term}) \Rightarrow \text{body}[x \leftarrow \text{term}]$

λ -Calculus Extensions

► Syntactic sugar:

λ -Calculus Extensions

► Syntactic sugar:

► $(\text{lam } (x) (\text{lam } (y) y)) \Rightarrow (\text{lam } (x\ y) y)$

λ -Calculus Extensions

- ▶ Syntactic sugar:
 - ▶ $(\text{lam } (x) (\text{lam } (y) y)) \Rightarrow (\text{lam } (x\ y) y)$
 - ▶ $((\text{foo bar}) \text{baz}) \Rightarrow (\text{foo bar baz})$

λ -Calculus Extensions

- ▶ Syntactic sugar:
 - ▶ $(\text{lam } (x) (\text{lam } (y) y)) \Rightarrow (\text{lam } (x\ y) y)$
 - ▶ $((\text{foo bar}) \text{baz}) \Rightarrow (\text{foo bar baz})$
- ▶ Local definitions: $(\text{let } ((\text{id } (\text{lam } (x) x))) (\text{id foo}))$

λ -Calculus Extensions

- ▶ Syntactic sugar:
 - ▶ $(\text{lam } (x) (\text{lam } (y) y)) \Rightarrow (\text{lam } (x\ y) y)$
 - ▶ $((\text{foo bar}) \text{baz}) \Rightarrow (\text{foo bar baz})$
- ▶ Local definitions: $(\text{let } ((\text{id } (\text{lam } (x) x))) (\text{id foo}))$
- ▶ Global definitions: $(\text{def id } (\text{lam } (x) x))$

λ -Calculus Extensions

► Primitive data:

λ -Calculus Extensions

- ▶ Primitive data:
 - ▶ booleans: true, false

λ -Calculus Extensions

- ▶ Primitive data:
 - ▶ booleans: true, false
 - ▶ numbers: 12, 0.27

λ -Calculus Extensions

- ▶ Primitive data:
 - ▶ booleans: true, false
 - ▶ numbers: 12, 0.27
 - ▶ strings: " λ -calculus rocks"

Typed λ -Calculus

- ▶ The λ -Calculus is the favorite playground of type theorists

Typed λ -Calculus

- ▶ The λ -Calculus is the favorite playground of type theorists
- ▶ There's a whole zoo of type systems we won't dive into (see you next year!)

Typed λ -Calculus

- ▶ The λ -Calculus is the favorite playground of type theorists
- ▶ There's a whole zoo of type systems we won't dive into (see you next year!)
- ▶ Search for `_λ Cube_online...`

Typed λ -Calculus

- ▶ The λ -Calculus is the favorite playground of type theorists
- ▶ There's a whole zoo of type systems we won't dive into (see you next year!)
- ▶ Search for `_λ Cube_` online...
- ▶ We'll introduce *Simply Typed λ -Calculus* aka. $\lambda_{\rightarrow}$

The Code



Action



Conclusion

