

The λ -calculus, from intuitionistic to classical logic

Webpage of the course

Davide Barbarossa

db2437@bath.ac.uk

Dept of Computer Science



Giulio Guerrieri

g.guerrieri@sussex.ac.uk

Dept of Computer Science



ESSLLI Summer School, Bochum (Germany)

28/07/2025 – 01/08/2025

The λ -calculus, from intuitionistic to classical logic

Lecture 2: The λ -calculus

Read the [notes](#): they are full of details, proofs, explanations, exercises, bibliography!

Davide Barbarossa

db2437@bath.ac.uk

Dept of Computer Science



ESSLLI Summer School, Bochum (Germany)

29/07/2025

Previously...

- What do we intuitively mean when we say that a function is **computable**
- That this relates to **topology**
- That $R = (\mathcal{P}(\mathbb{N}), \text{Scott})$ is a good **topological space** for modeling this
- That this topology only depends from the **partial order** \subseteq
 - That actually, Scott-topology and Scott-continuity themselves are a property about **“approximations”** in posets
 - That Scott-continuous functions are given by their restriction on the **finite elements** of R .
 - That continuous function **embed into** their base space. This is done via the **retraction** λ , fun, i.e. they satisfy (β)
 - That all Scott-continuous functions on R have **fixed points**!
 - That λ produces **RE sets** and fun preserves them



- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

Extracting a formal language from R

- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

Extracting a formal language from R

Let's take inspiration by the fact that the set of RE sets is closed wrt the following rules:

$$\overline{\lambda(\lambda \circ \text{curry}(\cdots(\lambda \circ \text{curry}(\text{proj}_i^n)) \cdots))}$$

$$\frac{a \mapsto f(a) \text{ computable}}{\lambda(f)}$$

$$\frac{a \quad b}{@ (a, b)}$$

Extracting a formal language from R

Let's take inspiration by the fact that the set of RE sets is closed wrt the following rules:

$$\overline{\lambda(\lambda \circ \text{curry}(\dots(\lambda \circ \text{curry}(\text{proj}_i^n)) \dots))} \quad \text{encoding of proj in } R$$

$$\frac{a \mapsto f(a) \quad \text{computable}}{\lambda(f)} \quad \begin{array}{c} \text{function over } R \\ \downarrow \\ \text{its encoding in } R \end{array}$$

$$\frac{a \quad b}{@ (a, b)} \quad \text{"application" in } R$$

Extracting a formal language from R

Let's take inspiration by the fact that the set of RE sets is closed wrt the following rules:

$$\overline{\lambda(\lambda \circ \text{curry}(\cdots(\lambda \circ \text{curry}(\text{proj}_i^n)) \cdots))} \quad \text{encoding of proj in } R$$

$$\frac{a \mapsto f(a) \quad \text{computable}}{\lambda(f)} \quad \begin{array}{c} \text{function over } R \\ \downarrow \\ \text{its encoding in } R \end{array} \quad \frac{a \quad b}{@ (a, b)} \quad \text{"application" in } R$$

We would like a functional programming language to be closed wrt the following rules:

representation of proj in the language

$$\begin{array}{c} \text{abstract function} \\ \downarrow \\ \text{its reification in the language} \end{array} \quad \text{"application" in the language}$$

Extracting a formal language from R

The set Λ^+ of λ -terms-with-context-variables is defined as:

$$\begin{array}{c} (x \in \underline{x}) \frac{}{\underline{x} \vdash x} \\[2ex] (y \notin \underline{x}) \frac{\underline{x}, y \vdash M}{\underline{x} \vdash \lambda y. M} \qquad \frac{\underline{x} \vdash M \quad \underline{x} \vdash N}{\underline{x} \vdash MN} \end{array}$$

We would like a functional programming language to be closed wrt the following rules:

representation of proj in the language

abstract function
 \downarrow
its reification in the language

“application” in the language

The set Λ^+ of λ -terms-with-context-variables is defined as:

$$(\mathbf{x} \in \underline{\mathbf{x}}) \frac{}{\underline{\mathbf{x}} \vdash \mathbf{x}}$$

$$(\mathbf{y} \notin \underline{\mathbf{x}}) \frac{\underline{\mathbf{x}}, \mathbf{y} \vdash \mathbf{M}}{\underline{\mathbf{x}} \vdash \lambda \mathbf{y}. \mathbf{M}}$$

$$\frac{\underline{\mathbf{x}} \vdash \mathbf{M} \quad \underline{\mathbf{x}} \vdash \mathbf{N}}{\underline{\mathbf{x}} \vdash \mathbf{MN}}$$

The set Λ of λ -terms is defined as:

$$\mathbf{M} ::= \mathbf{x} \mid \lambda \mathbf{x}. \mathbf{M} \mid \mathbf{MN} \quad (\text{for } \mathbf{x} \in \{\mathbf{x}_1, \mathbf{x}_2, \dots\})$$

What's the actual formal definition of λ -terms?

That requires more carefulness than you think!

The “issue” of α -equivalence

$$\lambda x.M = \lambda y.(M\{x := y\}) \quad \text{whenever } y \notin FV(M)$$

In pen-and-paper research:

- Words quotiented by α -equivalence
- Trees quotiented by α -equivalence

In computer oriented research:

- De-Brujin indices
- Nominal sets
- Abstract syntax
- ...

In the notes:

- Graphs with built-in α -equivalence

For the lectures:

- Informal treatment and hoping all goes well...

- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

Definition (Semantics of Λ^\vdash in R)

We define, by induction on $\underline{x} \vdash M$, its R -interpretation $\llbracket \underline{x} \vdash M \rrbracket : R^{\underline{x}} \rightarrow R$ as:

$$\llbracket \underline{x} \vdash x \rrbracket := \text{proj}_{\underline{x}}^x, \quad \llbracket \underline{x} := \underline{a} \vdash x \rrbracket \quad := \quad a_x$$

$$\llbracket \underline{x} \vdash \lambda y.P \rrbracket, \quad \llbracket \underline{x} := \underline{a} \vdash \lambda y.P \rrbracket \quad := \quad \lambda (\llbracket \underline{x} := \underline{a}, y := (_) \vdash P \rrbracket) \quad \text{if } y \notin \underline{x}$$

$$\llbracket \underline{x} \vdash P Q \rrbracket, \quad \llbracket \underline{x} := \underline{a} \vdash P Q \rrbracket \quad := \quad @(\llbracket \underline{x} := \underline{a} \vdash P \rrbracket, \llbracket \underline{x} := \underline{a} \vdash Q \rrbracket)$$

Definition (Semantics of Λ^\vdash in R)

We define, by induction on $\underline{x} \vdash M$, its R -interpretation $\llbracket \underline{x} \vdash M \rrbracket : R^{\underline{x}} \rightarrow R$ as:

$$\llbracket \underline{x} \vdash x \rrbracket := \text{proj}_{\underline{x}}^x, \quad \llbracket \underline{x} := \underline{a} \vdash x \rrbracket := a_x$$

$$\llbracket \underline{x} \vdash \lambda y.P \rrbracket, \quad \llbracket \underline{x} := \underline{a} \vdash \lambda y.P \rrbracket := \lambda (\llbracket \underline{x} := \underline{a}, y := (_) \vdash P \rrbracket) \text{ if } y \notin \underline{x}$$

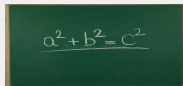
$$\llbracket \underline{x} \vdash P Q \rrbracket, \quad \llbracket \underline{x} := \underline{a} \vdash P Q \rrbracket := @(\llbracket \underline{x} := \underline{a} \vdash P \rrbracket, \llbracket \underline{x} := \underline{a} \vdash Q \rrbracket)$$

Remember that in R we have:

$$\text{fun} \circ \lambda = \text{id}_{R \Rightarrow R} \quad (\beta)$$

Fix $\underline{y}, x \vdash M$ with $x \notin \underline{y}$, and $\underline{y} \vdash N$. Then $\llbracket \underline{y} \vdash (\lambda x.M) N \rrbracket$ is the function:

$$\underline{a} \in R^{\underline{x}} \mapsto \llbracket \underline{y} := \underline{a} \vdash (\lambda x.M) N \rrbracket = \llbracket \underline{y} := \underline{a}, x := \llbracket \underline{y} := \underline{a} \vdash N \rrbracket \vdash M \rrbracket \in R$$



$$a^2 + b^2 = c^2$$

$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. M) N \rrbracket$ passes $\llbracket \underline{y} := \underline{a} \vdash N \rrbracket$ to $\llbracket \underline{y} := \underline{a}, \mathbf{x} \vdash M \rrbracket$ via \mathbf{x} .

Denotational Semantics of Λ^{\vdash} in R

$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. M) N \rrbracket$ passes $\llbracket \underline{y} := \underline{a} \vdash N \rrbracket$ to $\llbracket \underline{y} := \underline{a}, \mathbf{x} \vdash M \rrbracket$ via \mathbf{x} .

Such substitution *is itself the interpretation of a λ -term-with-context-variables!*

Definition (Substitution)

Given $M, N \in \Lambda$, $\mathbf{x} \in \text{Var}$, define the term $M\{\mathbf{x} := N\} \in \Lambda$ by induction:

$$\mathbf{x}\{\mathbf{x} := N\} := N \quad \mathbf{z}\{\mathbf{x} := N\} := \mathbf{z} \quad (P Q)\{\mathbf{x} := N\} := (P\{\mathbf{x} := N\})(Q\{\mathbf{x} := N\})$$

$$(\lambda \mathbf{z}. P)\{\mathbf{x} := N\} := \lambda \mathbf{z}. (P\{\mathbf{x} := N\}), \quad \text{if } \mathbf{z} \notin FV(N) \cup \{\mathbf{x}\}.$$

Denotational Semantics of Λ^{\vdash} in R

$\llbracket \underline{y} \vdash (\lambda x.M) N \rrbracket$ passes $\llbracket \underline{y} := \underline{a} \vdash N \rrbracket$ to $\llbracket \underline{y} := \underline{a}, x \vdash M \rrbracket$ via x .

Such substitution *is itself the interpretation of a λ -term-with-context-variables!*

Definition (Substitution)

Given $M, N \in \Lambda$, $x \in \text{Var}$, define the term $M\{x := N\} \in \Lambda$ by induction:

$$x\{x := N\} := N \quad z\{x := N\} := z \quad (PQ)\{x := N\} := (P\{x := N\})(Q\{x := N\})$$

$$(\lambda z.P)\{x := N\} := \lambda z.(P\{x := N\}), \quad \text{if } z \notin FV(N) \cup \{x\}.$$

Example

$$\begin{aligned} (\lambda y.x)\{x := y\} &= (\lambda v.x)\{x := y\} && \text{for } v \neq x, \text{ by } \alpha \\ &= \lambda v.y && \text{for } v \neq x, y, \text{ by def of substitution} \\ &= \lambda x.y && \text{for } x \neq y, \text{ by } \alpha \end{aligned}$$

Denotational Semantics of Λ^\vdash in R

$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. M) N \rrbracket$ passes $\llbracket \underline{y} := \underline{a} \vdash N \rrbracket$ to $\llbracket \underline{y} := \underline{a}, \mathbf{x} \vdash M \rrbracket$ via \mathbf{x} .

Such substitution *is itself the interpretation of a λ -term-with-context-variables!*

Definition (Substitution)

Given $M, N \in \Lambda$, $\mathbf{x} \in \text{Var}$, define the term $M\{\mathbf{x} := N\} \in \Lambda$ by induction:

$$\begin{aligned} \mathbf{x}\{\mathbf{x} := N\} &:= N & \mathbf{z}\{\mathbf{x} := N\} &:= \mathbf{z} & (P Q)\{\mathbf{x} := N\} &:= (P\{\mathbf{x} := N\})(Q\{\mathbf{x} := N\}) \\ (\lambda \mathbf{z}. P)\{\mathbf{x} := N\} &:= \lambda \mathbf{z}. (P\{\mathbf{x} := N\}), & \text{if } \mathbf{z} \notin FV(N) \cup \{\mathbf{x}\}. \end{aligned}$$

Theorem ($(R, \lambda, @)$ is a model of λ -calculus)

For all $\underline{y}, \mathbf{x} \vdash M$ with $\mathbf{x} \notin \underline{y}$, and $\underline{y} \vdash N$, we have:

$$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. M) N \rrbracket = \llbracket \underline{y} \vdash M\{\mathbf{x} := N\} \rrbracket : R^{\underline{y}} \rightarrow R. \quad (\llbracket \beta \rrbracket)$$

The proof is by induction on $\underline{y}, \mathbf{x} \vdash M$ (but careful with α -equivalence!).

Full β Operational Semantics of Λ

- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

$$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. \mathbf{M}) \mathbf{N} \rrbracket = \llbracket \underline{y} \vdash \mathbf{M}\{\mathbf{x} := \mathbf{N}\} \rrbracket : R^{\underline{y}} \rightarrow R. \quad (\llbracket \beta \rrbracket)$$

$$\llbracket \underline{y} \vdash (\lambda \mathbf{x}. M) N \rrbracket = \llbracket \underline{y} \vdash M\{\mathbf{x} := N\} \rrbracket$$

$$(\lambda x. M) N \quad = \quad M\{x := N\}$$

$$(\lambda \mathbf{x}. M) N \rightarrow_{\beta} M\{\mathbf{x} := N\}$$

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x.M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x.M \rightarrow_\beta \lambda x.N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M' N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta M N'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form of* M if $M \twoheadrightarrow_\beta N$.

Full β Operational Semantics of Λ

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x. M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x. M \rightarrow_\beta \lambda x. N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M' N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta M N'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

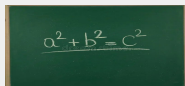
Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form of M* if $M \twoheadrightarrow_\beta N$.

Let $I := \lambda x. x$, let $\delta := \lambda x. x x$ and $\Omega := \delta \delta$. Show that:

$$\Omega \rightarrow_\beta \Omega \quad \text{and} \quad (\lambda x. I) \Omega =_\beta I I$$



Full β Operational Semantics of Λ

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x. M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x. M \rightarrow_\beta \lambda x. N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

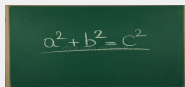
A normal term N is a *normal form of M* if $M \twoheadrightarrow_\beta N$.

Let $I := \lambda x. x$, let $\delta := \lambda x. x x$ and $\Omega := \delta \delta$. Show that:

$$\Omega \rightarrow_\beta \Omega \quad \text{and} \quad (\lambda x. I) \Omega =_\beta II$$

Let $\Delta_M := \lambda x. M(x x)$ for $x \notin FV(M)$.

For f a variable, does $\Delta_f \Delta_f$ have a normal form? Show that: $\Delta_I \Delta_I \twoheadrightarrow_\beta \Delta_I \Delta_I$



Full β Operational Semantics of Λ

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x. M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x. M \rightarrow_\beta \lambda x. N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form of M* if $M \twoheadrightarrow_\beta N$.

Let $I := \lambda x. x$, let $\delta := \lambda x. x x$ and $\Omega := \delta \delta$. Show that:

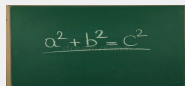
$$\Omega \rightarrow_\beta \Omega \quad \text{and} \quad (\lambda x. I) \Omega =_\beta I I$$

Let $\Delta_M := \lambda x. M (x x)$ for $x \notin FV(M)$.

For f a variable, does $\Delta_f \Delta_f$ have a normal form? Show that: $\Delta_I \Delta_I \twoheadrightarrow_\beta \Delta_I \Delta_I$

Let $\Psi := \lambda x. \lambda y. y (x x)$ and $\chi := \Psi \Psi$. Show that:

$$\chi M \twoheadrightarrow_\beta M \chi \quad \text{and} \quad \chi \twoheadrightarrow_\beta \lambda x. x \chi$$



Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x.M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x.M \rightarrow_\beta \lambda x.N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form of M* if $M \twoheadrightarrow_\beta N$.

Definition

A *fixed point combinator* is a term F such that $FM =_\beta M(FM)$ for all term M .

There are lots of them!

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x. M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x. M \rightarrow_\beta \lambda x. N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M' N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta M N'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form* of M if $M \twoheadrightarrow_\beta N$.

Definition

A *fixed point combinator* is a term F such that $FM =_\beta M(FM)$ for all term M .

There are lots of them!

Example

Remember $\Delta_f = \lambda x. f(xx)$. Show that $Y := \lambda f. \Delta_f \Delta_f$ is a fixed-point combinator

Full β Operational Semantics of Λ

Definition

The *full β -reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda x. M) N \rightarrow_\beta M\{x := N\}} \quad \frac{M \rightarrow_\beta N}{\lambda x. M \rightarrow_\beta \lambda x. N} \quad \frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M' N} \quad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta M N'}$$

Let \twoheadrightarrow_β be the reflexive and transitive closure of \rightarrow_β .

Let $=_\beta$ be the symmetric closure of \twoheadrightarrow_β .

A term M is *normal* if there is no possible \rightarrow_β -reduction from M .

A normal term N is a *normal form* of M if $M \twoheadrightarrow_\beta N$.

Definition

A *fixed point combinator* is a term F such that $FM =_\beta M(FM)$ for all term M .

There are lots of them!

Theorem (Church, Rosser)

The abstract rewriting system $(\Lambda, \rightarrow_\beta)$ is confluent.

\implies If a term has a normal form, then it is unique.

Full β Operational Semantics of Λ as a model of computation

Can we see $(\Lambda, \rightarrow_\beta)$ as a model of computation? Need to choose:

*some data-type A
(that terms computes on)*

some encoding $\ulcorner A \urcorner$ of A

*for each term M
a function (if any) f_M on $\ulcorner A \urcorner$*

Full β Operational Semantics of Λ as a model of computation

Can we see $(\Lambda, \rightarrow_\beta)$ as a model of computation? Yes!

data-type A
(that terms computes on) $\longrightarrow \mathbb{N}$

encoding $\ulcorner a \urcorner$ of $a \in A$ \longrightarrow *Church encoding* $\ulcorner n \urcorner$ of $n \in \mathbb{N}$

for each term M $\longrightarrow f_M(\ulcorner n \urcorner) := \text{nf}_\beta(M \ulcorner n \urcorner)$
a function (if any) f_M on $\ulcorner A \urcorner$ \longrightarrow *if it is a Church numeral*

A function $f : A \rightarrow A$ is implemented by M when

$$f_M \ulcorner a \urcorner = \ulcorner f(a) \urcorner$$

Full β Operational Semantics of Λ as a model of computation

Can we see $(\Lambda, \rightarrow_\beta)$ as a model of computation? Yes!

data-type A
(that terms computes on) $\longrightarrow \mathbb{N}$

encoding $\ulcorner a \urcorner$ of $a \in A$ \longrightarrow *Church encoding* $\ulcorner n \urcorner$ of $n \in \mathbb{N}$

for each term M $\longrightarrow f_M(\ulcorner n \urcorner) := \text{nf}_\beta(M \ulcorner n \urcorner)$
a function (if any) f_M on $\ulcorner A \urcorner$ \longrightarrow *if it is a Church numeral*

A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is implemented by M when

$$M \ulcorner n \urcorner =_\beta \ulcorner f(n) \urcorner$$

How expressive this is?

Full β Operational Semantics of Λ as a model of computation

Can we see $(\Lambda, \rightarrow_\beta)$ as a model of computation? Yes!

data-type A
(that terms computes on) $\longrightarrow \mathbb{N}$

encoding $\ulcorner a \urcorner$ of $a \in A$ \longrightarrow Church encoding $\ulcorner n \urcorner$ of $n \in \mathbb{N}$

for each term M
a function (if any) f_M on $\ulcorner A \urcorner$ \longrightarrow $f_M(\ulcorner n \urcorner) := \text{nf}_\beta(M \ulcorner n \urcorner)$
if it is a Church numeral

A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is implemented by M when

$$M \ulcorner n \urcorner =_\beta \ulcorner f(n) \urcorner$$

How expressive this is?

$(\Lambda, \rightarrow_\beta)$ under Church encoding is Turing-complete!

Under Church encoding, the *partial* functions $\mathbb{N} \rightarrow \mathbb{N}$ which are implementable in $(\Lambda, \rightarrow_\beta)$ are exactly the Turing-computable ones.

We did **not** mention how to handle *partiality* in all this discussion, it would require going deeper!

Basic pen-and-paper fun(ctional) programming together!

- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

Basic pen-and-paper fun(ctional) programming together!

Church encoding of Booleans:

$$\text{TRUE} := \lambda x. \lambda y. x \quad \text{FALSE} := \lambda x. \lambda y. y$$

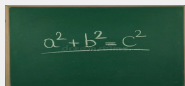
✎ Implement the following functions:

not and

Church encoding of natural numbers – *using notation*: $M^{(n)} x := M (M (\dots (M x)))$

$$\ulcorner n \urcorner := \lambda f. \lambda x. f^{(n)} x$$

✎ Implement the following functions:



successor addition is-zero-test

Suppose to have a term $\text{PRED} \in \Lambda$ that implements the predecessor function:

$$\text{PRED } \ulcorner 0 \urcorner =_{\beta} \ulcorner 0 \urcorner, \quad \text{PRED } \ulcorner n + 1 \urcorner =_{\beta} \ulcorner n \urcorner$$

✎ Implement the following functions:

subtraction less-than-or-equal-to-test

✎ Using a $\text{MULT} \in \Lambda$ implementing multiplication, implement the factorial:

$$(_)! : \mathbb{N} \rightarrow \mathbb{N}, \quad 0! := 1, \quad (n + 1)! := n! (n + 1)$$

- 1 Extracting a formal language from R
- 2 Denotational Semantics of Λ^+ in R
- 3 Full β Operational Semantics of Λ
- 4 Basic pen-and-paper fun(ctional) programming together!
- 5 Summary, exercises, bibliography

Summary, exercises, bibliography

- Inspired by the graph model, we have defined a very basic **functional programming language**
- We have given it a **denotational semantics** in the graph model
- We have given it a **operational semantics**
- Even if minimal, the OPS makes it a **Turing-complete** programming language
- We have **programmed** some basic functions on simple datatypes



Summary, exercises, bibliography

- Do the proofs of the statements in the slides
- Look at my **notes** on the [webpage of the course](#), there are plenty of **exercises**
- The exercises have **solutions** (but try to do them by yourself before looking at them!)

- Chapter 2 of:
Amadio R., Curien P-L, Domains and lambda-calculi, 1996,
[https://www.cambridge.org/core/books/
domains-and-lambdacalculi/4C6AB6938E436CFA8D5A8533B76A7F23](https://www.cambridge.org/core/books/domains-and-lambdacalculi/4C6AB6938E436CFA8D5A8533B76A7F23)
- Chapter 2, 3, 6 of:
Henk P. Barendregt, The lambda-calculus, its syntax and semantics, 1984,
[https://www.sciencedirect.com/bookseries/
studies-in-logic-and-the-foundations-of-mathematics/vol/103](https://www.sciencedirect.com/bookseries/studies-in-logic-and-the-foundations-of-mathematics/vol/103)
- Chapter 1 of:
PhD thesis of Giulio Manzonetto, Models and theories of lambda calculus, 2008,
<https://www.irif.fr/~gmanzone/ManzonettoPhdThesis.pdf>
- To go (much) further:
A Lambda Calculus Satellite, Henk Barendregt and Giulio Manzonetto, 2022,
<https://www.collegepublications.co.uk/logic/mlf/?00035>