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
UNIVERSITY OF BATH

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 $R = (\mathcal{P}(\mathbb{N}), \mathit{Scott})$ 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

Outline

1. Extracting a formal language from R
2. Denotational Semantics of $\Lambda^{\vdash}$ 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:

$$\frac{}{\lambda(\lambda \circ \mathrm{curry}(\cdots(\lambda \circ \mathrm{curry}(\mathrm{proj}_i^n))\cdots))}$$

$$\frac{a \mapsto f(a) \quad \textit{computable}}{\lambda(f)} \qquad\qquad \frac{a \qquad b}{@(a, b)}$$

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:

$$\frac{}{\lambda(\lambda \circ \mathrm{curry}(\cdots(\lambda \circ \mathrm{curry}(\mathrm{proj}_i^n))\cdots))} \quad \textit{encoding of } \mathrm{proj} \textit{ in } R$$

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

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:

$$\frac{}{\lambda(\lambda \circ \mathrm{curry}(\cdots(\lambda \circ \mathrm{curry}(\mathrm{proj}_i^n))\cdots))} \quad \textit{encoding of } \mathrm{proj} \textit{ in } R$$

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

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

representation of proj *in the language*

abstract function

↓

its reification in the language

“application” in the language

The set $\Lambda^{\vdash}$ of λ-terms-with-context-variables is defined as:

$$(\mathtt{x} \in \underline{\mathtt{x}})\,\frac{}{\underline{\mathtt{x}} \vdash \mathtt{x}}$$

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

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 $\Lambda^{\vdash}$ of λ-terms-with-context-variables is defined as:

$$(\mathtt{x} \in \underline{\mathtt{x}})\,\frac{}{\underline{\mathtt{x}} \vdash \mathtt{x}}$$

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

The set Λ of λ-terms is defined as:

$$\mathtt{M} ::= \mathtt{x} \mid \lambda \mathtt{x}.\mathtt{M} \mid \mathtt{M}\,\mathtt{N} \qquad \textit{(for } \mathtt{x} \in \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}\textit{)}$$

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

That requires more carefulness than you think!

The "issue" of α-equivalence

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

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...

Definition (Semantics of $\Lambda^{\vdash}$ in R)

We define, by induction on $\underline{\mathtt{x}} \vdash \mathtt{M}$, its *$R$-interpretation* $[\![\underline{\mathtt{x}} \vdash \mathtt{M}]\!] : R^{\underline{\mathtt{x}}} \to R$ as:

$$[\![\underline{\mathtt{x}} \vdash \mathtt{x}]\!] := \mathrm{proj}^{\underline{\mathtt{x}}}_{\mathtt{x}}, \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{x}]\!] \quad := \quad a_{\mathtt{x}}$$

$$[\![\underline{\mathtt{x}} \vdash \lambda \mathtt{y}.\mathtt{P}]\!], \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \lambda \mathtt{y}.\mathtt{P}]\!] \quad := \quad \lambda\left([\![\underline{\mathtt{x}} := \underline{a}, \mathtt{y} := (\_) \vdash \mathtt{P}]\!]\right) \quad \textit{if } \mathtt{y} \notin \underline{\mathtt{x}}$$

$$[\![\underline{\mathtt{x}} \vdash \mathtt{P}\,\mathtt{Q}]\!], \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{P}\,\mathtt{Q}]\!] \quad := \quad @([\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{P}]\!], [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{Q}]\!])$$

Definition (Semantics of $\Lambda^{\vdash}$ in R)

We define, by induction on $\underline{\mathtt{x}} \vdash \mathtt{M}$, its *$R$-interpretation* $[\![\underline{\mathtt{x}} \vdash \mathtt{M}]\!] : R^{\underline{\mathtt{x}}} \to R$ as:

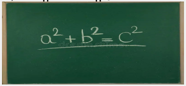
$$
\begin{array}{lll}
[\![\underline{\mathtt{x}} \vdash \mathtt{x}]\!] := \mathrm{proj}^{\underline{\mathtt{x}}}_{\mathtt{x}}, \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{x}]\!] & := & a_{\mathtt{x}} \\
[\![\underline{\mathtt{x}} \vdash \lambda \mathtt{y}.\mathtt{P}]\!]\,, \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \lambda \mathtt{y}.\mathtt{P}]\!] & := & \lambda\left([\![\underline{\mathtt{x}} := \underline{a}, \mathtt{y} := (\_) \vdash \mathtt{P}]\!]\right) \;\; \textit{if} \;\; \mathtt{y} \notin \underline{\mathtt{x}} \\
[\![\underline{\mathtt{x}} \vdash \mathtt{P}\,\mathtt{Q}]\!]\,, \quad [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{P}\,\mathtt{Q}]\!] & := & @\left([\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{P}]\!]\,,\, [\![\underline{\mathtt{x}} := \underline{a} \vdash \mathtt{Q}]\!]\right)
\end{array}
$$

Remember that in R we have:

$$\mathrm{fun} \circ \lambda = \mathrm{id}_{R \Rightarrow R} \qquad (\beta)$$

Fix $\underline{\mathtt{y}}, \mathtt{x} \vdash \mathtt{M}$ with $\mathtt{x} \notin \underline{\mathtt{y}}$, and $\underline{\mathtt{y}} \vdash \mathtt{N}$. Then $\left[\!\!\left[\underline{\mathtt{y}} \vdash (\lambda \mathtt{x}.\mathtt{M})\,\mathtt{N}\right]\!\!\right]$ is the function:

$$\underline{a} \;\in\; R^{\underline{\mathtt{x}}} \;\mapsto\; \left[\!\!\left[\underline{\mathtt{y}} := \underline{a} \vdash (\lambda \mathtt{x}.\mathtt{M})\,\mathtt{N}\right]\!\!\right] = \left[\!\!\left[\underline{\mathtt{y}} := \underline{a}, \mathtt{x} := \left[\!\!\left[\underline{\mathtt{y}} := \underline{a} \vdash \mathtt{N}\right]\!\!\right] \vdash \mathtt{M}\right]\!\!\right] \in R$$



$[\![\underline{\mathtt{y}} \vdash (\lambda \mathtt{x}.\mathtt{M})\,\mathtt{N}]\!]$ passes $[\![\underline{\mathtt{y}} := \underline{a} \vdash \mathtt{N}]\!]$ to $[\![\underline{\mathtt{y}} := \underline{a},\, \mathtt{x} \vdash \mathtt{M}]\!]$ via $\mathtt{x}$.

$[\![\underline{\mathtt{y}} \vdash (\lambda\mathtt{x}.\mathtt{M})\,\mathtt{N}]\!]$ passes $[\![\underline{\mathtt{y}} := \underline{a} \vdash \mathtt{N}]\!]$ to $[\![\underline{\mathtt{y}} := \underline{a}, \mathtt{x} \vdash \mathtt{M}]\!]$ via $\mathtt{x}$.

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

Definition (Substitution)

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

$$\mathtt{x}\{\mathtt{x} := \mathtt{N}\} := \mathtt{N} \qquad \mathtt{z}\{\mathtt{x} := \mathtt{N}\} := \mathtt{z} \qquad (\mathtt{P}\,\mathtt{Q})\{\mathtt{x} := \mathtt{N}\} := (\mathtt{P}\{\mathtt{x} := \mathtt{N}\})(\mathtt{Q}\{\mathtt{x} := \mathtt{N}\})$$

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

$\left[\!\left[\underline{\mathtt{y}} \vdash (\lambda\mathtt{x}.\mathtt{M})\,\mathtt{N}\right]\!\right]$ passes $\left[\!\left[\underline{\mathtt{y}} := \underline{a} \vdash \mathtt{N}\right]\!\right]$ to $\left[\!\left[\underline{\mathtt{y}} := \underline{a},\ \mathtt{x} \vdash \mathtt{M}\right]\!\right]$ via $\mathtt{x}$.

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

Definition (Substitution)

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

$$\mathtt{x}\{\mathtt{x} := \mathtt{N}\} := \mathtt{N} \qquad \mathtt{z}\{\mathtt{x} := \mathtt{N}\} := \mathtt{z} \qquad (\mathtt{P}\,\mathtt{Q})\{\mathtt{x} := \mathtt{N}\} := (\mathtt{P}\{\mathtt{x} := \mathtt{N}\})(\mathtt{Q}\{\mathtt{x} := \mathtt{N}\})$$

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

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

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

$$\left[\!\left[\underline{\mathtt{y}} \vdash (\lambda\mathtt{x}.\,\mathtt{M})\,\mathtt{N}\right]\!\right] = \left[\!\left[\underline{\mathtt{y}} \vdash \mathtt{M}\{\mathtt{x} := \mathtt{N}\}\right]\!\right] : R^{\underline{\mathtt{y}}} \to R. \qquad ([\![\beta]\!])$$

Full β Operational Semantics of Λ

$$\left[\!\!\left[\underline{\mathtt{y}} \vdash (\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N}\right]\!\!\right] = \left[\!\!\left[\underline{\mathtt{y}} \vdash \mathtt{M}\{\mathtt{x} := \mathtt{N}\}\right]\!\!\right] : R^{\underline{\mathtt{y}}} \to R. \tag{$[\![\beta]\!]$}$$

Full β Operational Semantics of Λ

$$\left[\!\left[\underline{\mathtt{y}} \vdash (\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N}\right]\!\right] = \left[\!\left[\underline{\mathtt{y}} \vdash \mathtt{M}\{\mathtt{x} := \mathtt{N}\}\right]\!\right]$$

Full β Operational Semantics of Λ

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

Full β Operational Semantics of Λ

$$(\lambda \mathtt{x}.\, \mathtt{M})\, \mathtt{N} \quad \to_\beta \quad \mathtt{M}\{\mathtt{x} := \mathtt{N}\}$$

Full β Operational Semantics of Λ

Definition

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

$$\frac{}{(\lambda \mathtt{x}.\, \mathtt{M})\, \mathtt{N} \rightarrow_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \rightarrow_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \rightarrow_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\rightarrow_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\rightarrow_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Full β Operational Semantics of Λ

Definition

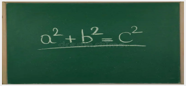
The *full β-reduction* $\to_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N} \to_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \to_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \to_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\to_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Let $\delta := \lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x}$ and $\Omega := \delta\,\delta$. Show that:

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



Full β Operational Semantics of Λ

Definition

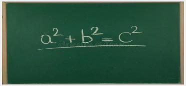
The *full β-reduction* $\rightarrow_\beta \subseteq \Lambda \times \Lambda$ is defined by the following inductive rules:

$$\frac{}{(\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N} \rightarrow_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \rightarrow_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \rightarrow_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\rightarrow_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\rightarrow_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Let $\delta := \lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x}$ and $\Omega := \delta\,\delta$. Show that:

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



Let $\Delta_{\mathtt{M}} := \lambda \mathtt{x}.\,\mathtt{M}\,(\mathtt{x}\,\mathtt{x})$.

For $\mathtt{f}$ a variable, does $\Delta_{\mathtt{f}}\Delta_{\mathtt{f}}$ have a normal form? Show that: $\Delta_{\mathtt{I}}\Delta_{\mathtt{I}} \twoheadrightarrow_\beta \Delta_{\mathtt{I}}\Delta_{\mathtt{I}}$

Full β Operational Semantics of Λ

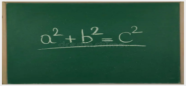
Definition

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

$$\frac{}{(\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N} \to_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \to_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \to_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\to_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Let $\delta := \lambda \mathtt{x}.\,\mathtt{x}\,\mathtt{x}$ and $\Omega := \delta\,\delta$. Show that:



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

Let $\Delta_{\mathtt{M}} := \lambda \mathtt{x}.\,\mathtt{M}\,(\mathtt{x}\,\mathtt{x})$.

For $\mathtt{f}$ a variable, does $\Delta_{\mathtt{f}}\Delta_{\mathtt{f}}$ have a normal form? Show that: $\Delta_{\mathtt{I}}\Delta_{\mathtt{I}} \twoheadrightarrow_\beta \Delta_{\mathtt{I}}\Delta_{\mathtt{I}}$

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

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

Full β Operational Semantics of Λ

Definition

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

$$\frac{}{(\lambda \mathtt{x}.\, \mathtt{M})\, \mathtt{N} \rightarrow_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \rightarrow_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \rightarrow_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\rightarrow_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\rightarrow_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Definition

A *fixed point combinator* is a term $\mathtt{F}$ such that $\mathtt{F}\,\mathtt{M} =_\beta \mathtt{M}\,(\mathtt{F}\,\mathtt{M})$ for all term $\mathtt{M}$.
There are lots of them!

Full β Operational Semantics of Λ

Definition

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

$$\frac{}{(\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N} \rightarrow_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \rightarrow_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \rightarrow_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \rightarrow_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \rightarrow_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\rightarrow_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\rightarrow_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Definition

A *fixed point combinator* is a term $\mathtt{F}$ such that $\mathtt{F}\,\mathtt{M} =_\beta \mathtt{M}\,(\mathtt{F}\,\mathtt{M})$ for all term $\mathtt{M}$.
There are lots of them!

Example

Remember $\Delta_{\mathtt{f}} = \lambda \mathtt{x}.\mathtt{f}(\mathtt{x}\mathtt{x})$. Show that $\mathtt{Y} := \lambda \mathtt{f}.\,\Delta_{\mathtt{f}}\Delta_{\mathtt{f}}$ is a fixed-point combinator

Full β Operational Semantics of Λ

Definition

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

$$\frac{}{(\lambda \mathtt{x}.\,\mathtt{M})\,\mathtt{N} \to_\beta \mathtt{M}\{\mathtt{x} := \mathtt{N}\}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{N}}{\lambda \mathtt{x}.\mathtt{M} \to_\beta \lambda \mathtt{x}.\mathtt{N}} \qquad \frac{\mathtt{M} \to_\beta \mathtt{M}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}'\,\mathtt{N}} \qquad \frac{\mathtt{N} \to_\beta \mathtt{N}'}{\mathtt{M}\,\mathtt{N} \to_\beta \mathtt{M}\,\mathtt{N}'}$$

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term $\mathtt{M}$ is *normal* if there is no possible $\to_\beta$-reduction from $\mathtt{M}$.
A normal term $\mathtt{N}$ is a *normal form of* $\mathtt{M}$ if $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.

Definition

A *fixed point combinator* is a term $\mathtt{F}$ such that $\mathtt{F}\,\mathtt{M} =_\beta \mathtt{M}\,(\mathtt{F}\,\mathtt{M})$ for all term $\mathtt{M}$.
There are lots of them!

Theorem (Church, Rosser)

The abstract rewriting system $(\Lambda, \to_\beta)$ is confluent.
$\Longrightarrow$ *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 $(\Lambda, \to_\beta)$ 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) $f_{\mathtt{M}}$ *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 $\mathtt{M}$ *a function (if any)* $f_{\mathtt{M}}$ *on* $\ulcorner A \urcorner$	$\longrightarrow$	$f_{\mathtt{M}}(\ulcorner n \urcorner) := \mathrm{nf}_\beta(\mathtt{M} \ulcorner n \urcorner)$ *if it is a Church numeral*

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

$$f_{\mathtt{M}} \ulcorner a \urcorner = \ulcorner f(a) \urcorner$$

Full β Operational Semantics of Λ as a model of computation

Can we see $(\Lambda, \to_\beta)$ 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 $\mathtt{M}$ *a function (if any)* $f_{\mathtt{M}}$ *on* $\ulcorner A \urcorner$	$\longrightarrow$	$f_{\mathtt{M}}(\ulcorner n \urcorner) := \mathrm{nf}_\beta(\mathtt{M}\ulcorner n \urcorner)$ *if it is a Church numeral*

A function $f : \mathbb{N} \to \mathbb{N}$ is implemented by $\mathtt{M}$ 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 $\mathtt{M}$ *a function (if any)* $f_\mathtt{M}$ *on* $\ulcorner A \urcorner$	$\longrightarrow$	$f_\mathtt{M}(\ulcorner n \urcorner) := \mathrm{nf}_\beta(\mathtt{M}\ulcorner n \urcorner)$ *if it is a Church numeral*

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

Church encoding of Booleans:

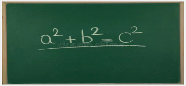
$$\textsc{true} := \lambda \mathtt{x}.\lambda \mathtt{y}.\,\mathtt{x} \qquad \textsc{false} := \lambda \mathtt{x}.\lambda \mathtt{y}.\,\mathtt{y}$$

✎ Implement the following functions:

not *and*

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

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



✎ Implement the following functions:

successor *addition* *is-zero-test*

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

$$\textsc{pred}\,\ulcorner 0 \urcorner \;=_\beta\; \ulcorner 0 \urcorner\,, \qquad \textsc{pred}\,\ulcorner n+1 \urcorner \;=_\beta\; \ulcorner n \urcorner$$

✎ Implement the following functions:

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

✎ Implement the factorial function:

$$(\_)! : \mathbb{N} \to \mathbb{N}, \qquad 0! := 1 \quad \textit{and} \quad (n+1)! := n!\,(n+1)$$

- 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

- 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
- 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
- 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