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

UNIVERSITY OF
BATH

US
UNIVERSITY
OF SUSSEX

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*

- What do we intuitively mean when we say that a function is computable
- That this relates to topology
- That $R = (\mathcal{P}(\mathbb{N}), Scott)$ 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

What have we learned yesterday?

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

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

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

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

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

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

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

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

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

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

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

$$\textit{representation of } \mathrm{proj} \textit{ in the language}$$

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

Fix a countable set of variables.

Notation: $\underline{\mathtt{x}}$ is a finite set of variables.

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*

Fix a countable set of variables.

Notation: $\underline{\mathbf{x}}$ is a finite set of variables.

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

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

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

Remark: if $\underline{\mathtt{x}} \vdash \mathtt{M}$ then $\underline{\mathtt{x}}$ contains at least the free variables $FV(\mathtt{M})$ of $\mathtt{M}$.

The set Λ of λ-terms is defined as:

$$\mathtt{M} ::= \mathtt{x} \mid \lambda \mathtt{x}.\mathtt{M} \mid \mathtt{M}\,\mathtt{N} \qquad \textit{(for $\mathtt{x}$ a variable)}$$

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...

$R^{\underline{x}} :=$ families of elements of R indexed by $\underline{x}$. I.e. $\underline{a} \in R^{\underline{x}}$ iff $\underline{a} = \{a_{\mathtt{x}} \mid \mathtt{x} \in \underline{x},\, a_{\mathtt{x}} \in R\}$.

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

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

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

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

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

$R^{\underline{x}} :=$ families of elements of R indexed by $\underline{x}$. I.e. $\underline{a} \in R^{\underline{x}}$ iff $\underline{a} = \{a_x \mid x \in \underline{x}, a_x \in R\}$.

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

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

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

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

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

Remember that in R we have:
$$\mathrm{fun} \circ \lambda = \mathrm{id}_{R \Rightarrow R} \qquad\qquad (\beta)$$

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

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

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

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

$\llbracket \underline{y} \vdash (\lambda x.M)\,N \rrbracket$ 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 $M, N \in \Lambda$, $x \in \mathrm{Var}$, define the term $M\{x := N\} \in \Lambda$ by induction:

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

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

$\llbracket \underline{y} \vdash (\lambda x.M)\,N \rrbracket$ 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 $M, N \in \Lambda$, $x \in \mathrm{Var}$, define the term $M\{x := N\} \in \Lambda$ by induction:

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

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

Example

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

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

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

Definition (Substitution)

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

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

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

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

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

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

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

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

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

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

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

Definition

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

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

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
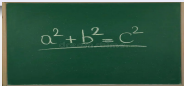Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term M is *normal* if there is no possible $\to_\beta$-reduction from M.
A normal term N is a *normal form of* M if $M \twoheadrightarrow_\beta N$.
A *redex* is any term of shape $(\lambda x. M) N$.

Definition

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

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

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term M is *normal* if there is no possible $\to_\beta$-reduction from M.
A normal term N is a *normal form of* M if M $\twoheadrightarrow_\beta$ N.
A *redex* is any term of shape $(\lambda\texttt{x.M})\,\texttt{N}$.

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

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

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

Definition

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

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

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term M is *normal* if there is no possible $\to_\beta$-reduction from M.
A normal term N is a *normal form of* M if $M \twoheadrightarrow_\beta N$.
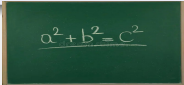A *redex* is any term of shape $(\lambda x.\, M)\, N$.

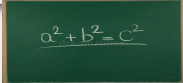Let $I := \lambda x.\, x$, let $\delta := \lambda x.\, x\, x$ and $\Omega := \delta\, \delta$. Show that:

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

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: $\Delta_I \Delta_I \twoheadrightarrow_\beta \Delta_I \Delta_I$

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 $\mathtt{M} \twoheadrightarrow_\beta \mathtt{N}$.
A *redex* is any term of shape $(\lambda \mathtt{x}.\, \mathtt{M})\, \mathtt{N}$.

Let $\mathtt{I} := \lambda \mathtt{x}.\, \mathtt{x}$, 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 $\mathtt{x} \notin FV(\mathtt{M})$.
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:

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

Definition

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

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

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\rightarrow_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term M is *normal* if there is no possible $\rightarrow_\beta$-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 $\texttt{F}\,\texttt{M} =_\beta \texttt{M}\,(\texttt{F}\,\texttt{M})$ for all term M.
There are lots of them!

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!

Example

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

Definition

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

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

Let $\twoheadrightarrow_\beta$ be the reflexive and transitive closure of $\to_\beta$.
Let $=_\beta$ be the symmetric closure of $\twoheadrightarrow_\beta$.
A term M is *normal* if there is no possible $\to_\beta$-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 $F\, M =_\beta M\, (F\, M)$ for all term M.
There are lots of them!

Theorem (Church, Rosser)

The abstract rewriting system $(\Lambda, \to_\beta)$ 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 $(\Lambda, \to_\beta)$ 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_M *on* $\ulcorner A \urcorner$

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

$$\begin{array}{ccc}
\textit{data-type} \;\; \text{A} & & \\
\textit{(that terms computes on)} & \longrightarrow & \mathbb{N} \\
& & \\
\textit{encoding} \; \ulcorner a \urcorner \; \textit{of } a \in A & \longrightarrow & \textit{Church encoding } \ulcorner n \urcorner \textit{ of } n \in \mathbb{N} \\
& & \\
\textit{for each term } \text{M} & & f_{\text{M}}(\ulcorner n \urcorner) := \text{nf}_\beta(\text{M} \ulcorner n \urcorner) \\
\textit{a function (if any) } f_{\text{M}} \textit{ on } \ulcorner A \urcorner & \longrightarrow & \textit{if it is a Church numeral}
\end{array}$$

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

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

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

$$
\begin{array}{ccc}
\textit{data-type } \text{A} & & \\
\textit{(that terms computes on)} & \longrightarrow & \mathbb{N} \\
& & \\
\textit{encoding } \ulcorner a \urcorner \textit{ of } a \in A & \longrightarrow & \textit{Church encoding } \ulcorner n \urcorner \textit{ of } n \in \mathbb{N} \\
& & \\
\textit{for each term } \text{M} & & f_\text{M}(\ulcorner n \urcorner) := \text{nf}_\beta(\text{M}\ulcorner n \urcorner) \\
\textit{a function (if any) } f_\text{M} \textit{ on } \ulcorner A \urcorner & \longrightarrow & \textit{if it is a Church numeral}
\end{array}
$$

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

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

How expressive this is?

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

$$
\begin{array}{ccl}
\textit{data-type} \ \ \mathrm{A} \\
\textit{(that terms computes on)} & \longrightarrow & \mathbb{N} \\[2ex]
\textit{encoding} \ \ulcorner a \urcorner \ \textit{of } a \in A & \longrightarrow & \textit{Church encoding} \ \ulcorner n \urcorner \ \textit{of } n \in \mathbb{N} \\[2ex]
\textit{for each term } \mathtt{M} \\
\textit{a function (if any) } f_\mathtt{M} \ \textit{on} \ \ulcorner A \urcorner & \longrightarrow & f_\mathtt{M}(\ulcorner n \urcorner) := \mathrm{nf}_\beta(\mathtt{M}\ulcorner n \urcorner) \\
& & \textit{if it is a Church numeral}
\end{array}
$$

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

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

How expressive this is?

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

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

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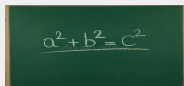
Church encoding of Booleans:

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

✎ Implement the following functions:

$$not : \mathrm{Bool} \rightarrow \mathrm{Bool} \quad and : \mathrm{Bool} \times \mathrm{Bool} \rightarrow \mathrm{Bool}$$

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:

$$successor : \mathrm{nat} \rightarrow \mathrm{nat} \quad addition : \mathrm{nat} \times \mathrm{nat} \rightarrow \mathrm{nat} \quad is\text{-}zero\text{-}test : \mathrm{nat} \rightarrow \mathrm{Bool}$$

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

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

✎ Implement the following functions:

$$subtraction : \mathrm{nat} \times \mathrm{nat} \rightarrow \mathrm{nat} \quad less\text{-}than\text{-}or\text{-}equal\text{-}to\text{-}test : \mathrm{nat} \times \mathrm{nat} \rightarrow \mathrm{Bool}$$

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

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

- 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

What have we learned today?

- 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