The λ-calculus, typed and untyped

Lecture 4: The untyped λ-calculus

Giulio Guerrieri

✉ giulio.guerrieri@uibk.ac.at 🌍 https://cl-teaching.uibk.ac.at/26w/lc

Institut für Informatik, Universität Innsbruck

Innsbruck, Austria – 6 November 2026

Outline

1. The syntax and the operational semantics of the untyped λ-calculus
2. The proof of confluence of β-reduction
3. The notions of β-equivalence and $\beta\eta$-equivalence
4. Conclusion, exercises and bibliography

Outline

1. The syntax and the operational semantics of the untyped λ-calculus
2. The proof of confluence of β-reduction
3. The notions of β-equivalence and $\beta\eta$-equivalence
4. Conclusion, exercises and bibliography

The λ-calculus beyond simple types

Term and β-reduction of the simply typed λ-calculus can be defined without types.
$\rightsquigarrow$ Let us explore the word of the λ-calculus without types.

1. What do we gain?
2. What do we lose?

We can freely apply s to t to get st, without requiring $s : A \Rightarrow B$ or $t : A$.

Consider the term $\delta = \lambda x.xx$. It not a term for the simply typed λ-calculus.

- Why is there no A such that $\vdash \lambda x.xx : A$ is derivable?
- $\delta\delta = (\lambda x.xx)\delta \to_\beta (xx)\{\delta/x\} = \delta\delta \to_\beta \dots$ $\rightsquigarrow$ The untyped λ-calculus is not SN.
- $\delta\delta \to_\beta \delta\delta$ is the only possible β-reduction from $\delta\delta$ $\rightsquigarrow$ it is not even normalizing!

The λ-calculus beyond simple types

Term and β-reduction of the simply typed λ-calculus can be defined without types.
$\rightsquigarrow$ Let us explore the word of the λ-calculus without types.

1. What do we gain?
2. What do we lose?

We can freely apply s to t to get st, without requiring $s : A \Rightarrow B$ or $t : A$.

Consider the term $\delta = \lambda x.xx$. It not a term for the simply typed λ-calculus.

- Why is there no A such that $\vdash \lambda x.xx : A$ is derivable?
- $\delta\delta = (\lambda x.xx)\delta \to_\beta (xx)\{\delta/x\} = \delta\delta \to_\beta \dots$ $\rightsquigarrow$ The untyped λ-calculus is not SN.
- $\delta\delta \to_\beta \delta\delta$ is the only possible β-reduction from $\delta\delta$ $\rightsquigarrow$ it is not even normalizing!

The λ-calculus beyond simple types

Term and β-reduction of the simply typed λ-calculus can be defined without types.
$\rightsquigarrow$ Let us explore the word of the λ-calculus without types.

1. What do we gain?
2. What do we lose?

We can freely apply s to t to get st, without requiring $s : A \Rightarrow B$ or $t : A$.

Consider the term $\delta = \lambda x.xx$. It not a term for the simply typed λ-calculus.

- Why is there no A such that $\vdash \lambda x.xx : A$ is derivable?
- $\delta\delta = (\lambda x.xx)\delta \to_\beta (xx)\{\delta/x\} = \delta\delta \to_\beta \dots$ $\rightsquigarrow$ The untyped λ-calculus is not SN.
- $\delta\delta \to_\beta \delta\delta$ is the only possible β-reduction from $\delta\delta$ $\rightsquigarrow$ it is not even normalizing!

The λ-calculus beyond simple types

Term and β-reduction of the simply typed λ-calculus can be defined without types.
$\rightsquigarrow$ Let us explore the word of the λ-calculus without types.

1. What do we gain?
2. What do we lose?

We can freely apply s to t to get st, without requiring $s : A \Rightarrow B$ or $t : A$.

Consider the term $\delta = \lambda x.xx$. It not a term for the simply typed λ-calculus.

- Why is there no A such that $\vdash \lambda x.xx : A$ is derivable?
- $\delta\delta = (\lambda x.xx)\delta \to_\beta (xx)\{\delta/x\} = \delta\delta \to_\beta \dots$ $\rightsquigarrow$ The untyped λ-calculus is not SN.
- $\delta\delta \to_\beta \delta\delta$ is the only possible β-reduction from $\delta\delta$ $\rightsquigarrow$ it is not even normalizing!

The untyped λ-calculus: the philosophy

Functions are treated anonymously, that is, without giving them a name:

$$\mathrm{id}(x) = x \quad \rightsquigarrow \quad x \mapsto x \qquad\qquad \mathrm{sq_sum}(x, y) = x^2 + y^2 \quad \rightsquigarrow \quad (x, y) \mapsto x^2 + y^2$$

Functions of several arguments can be transformed into function of a single argument:

$$(x, y) \mapsto x^2 + y^2 \quad \rightsquigarrow \quad x \mapsto (y \mapsto x^2 + y^2) \qquad \text{(currying)}$$

Functions can be applied to functions and can return functions (higher-order):

$$(x \mapsto x)5 = 5 \qquad\qquad (x \mapsto x)(y \mapsto y^2) = y \mapsto y^2$$

The untyped λ-calculus performs higher-order computation:

- everything is an anonymous function with a single argument (λ-calculus);
- functions can be applied to other functions without any restriction (untyped).

The untyped λ-calculus: the philosophy

Functions are treated anonymously, that is, without giving them a name:

$$\mathrm{id}(x) = x \quad \rightsquigarrow \quad x \mapsto x \qquad\qquad \mathrm{sq_sum}(x, y) = x^2 + y^2 \quad \rightsquigarrow \quad (x, y) \mapsto x^2 + y^2$$

Functions of several arguments can be transformed into function of a single argument:

$$(x, y) \mapsto x^2 + y^2 \qquad \rightsquigarrow \qquad x \mapsto (y \mapsto x^2 + y^2) \qquad \text{(currying)}$$

Functions can be applied to functions and can return functions (higher-order):

$$(x \mapsto x)5 = 5 \qquad\qquad (x \mapsto x)(y \mapsto y^2) = y \mapsto y^2$$

The untyped λ-calculus performs higher-order computation:

- everything is an anonymous function with a single argument (λ-calculus);
- functions can be applied to other functions without any restriction (untyped).

The untyped λ-calculus: the philosophy

Functions are treated anonymously, that is, without giving them a name:

$$\mathrm{id}(x) = x \quad \rightsquigarrow \quad x \mapsto x \qquad\qquad \mathrm{sq_sum}(x, y) = x^2 + y^2 \quad \rightsquigarrow \quad (x, y) \mapsto x^2 + y^2$$

Functions of several arguments can be transformed into function of a single argument:

$$(x, y) \mapsto x^2 + y^2 \qquad \rightsquigarrow \qquad x \mapsto (y \mapsto x^2 + y^2) \qquad \text{(currying)}$$

Functions can be applied to functions and can return functions (higher-order):

$$(x \mapsto x)5 = 5 \qquad\qquad (x \mapsto x)(y \mapsto y^2) = y \mapsto y^2$$

The untyped λ-calculus performs higher-order computation:

- everything is an anonymous function with a single argument (λ-calculus);
- functions can be applied to other functions without any restriction (untyped).

The untyped λ-calculus: the philosophy

Functions are treated anonymously, that is, without giving them a name:

$$\mathrm{id}(x) = x \quad \rightsquigarrow \quad x \mapsto x \qquad\qquad \mathrm{sq_sum}(x, y) = x^2 + y^2 \quad \rightsquigarrow \quad (x, y) \mapsto x^2 + y^2$$

Functions of several arguments can be transformed into function of a single argument:

$$(x, y) \mapsto x^2 + y^2 \qquad \rightsquigarrow \qquad x \mapsto (y \mapsto x^2 + y^2) \qquad\qquad \text{(currying)}$$

Functions can be applied to functions and can return functions (higher-order):

$$(x \mapsto x)5 = 5 \qquad\qquad (x \mapsto x)(y \mapsto y^2) = y \mapsto y^2$$

The untyped λ-calculus performs higher-order computation:

- everything is an anonymous function with a single argument (λ-calculus);
- functions can be applied to other functions without any restriction (untyped).

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $\quad (\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $(\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $(\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $(\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $(\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The untyped λ-calculus, formally

(λ-)terms: $s, t ::= x$ (variable) $\mid \lambda x.t$ (abstraction) $\mid st$ (application) Rmk: stu stands for $(st)u$.

The free variables of a term t are the variables that are not bound to a λ. Formally,

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(st) = \mathrm{fv}(s) \cup \mathrm{fv}(t) \qquad \mathrm{fv}(\lambda x.t) = \mathrm{fv}(t) \setminus \{x\}$$

Terms are identified up to renaming of bound variables (α-equivalence), e.g. $\lambda x.x = \lambda y.y$

β-reduction ($t\{s/x\}$ is the capture-avoiding substitution of s for the free occurrences of x in t):

(the term on the left is a β-redex) $(\lambda x.t)s \to_\beta t\{s/x\}$

Substitution $t\{s/x\}$ should be defined carefully to avoid capture of variables.

$$(\lambda x.yx)\{x/y\} \neq \lambda x.xx \qquad \text{but} \qquad (\lambda x.yx)\{x/y\} = (\lambda z.yz)\{x/y\} = \lambda z.xz$$

To write $t\{s/x\}$, first take t such that its bound variables are not in $\mathrm{fv}(s)$ then substitute:

$$x\{s/x\} := s \quad y\{s/x\} := y \quad (\lambda y.t)\{s/x\} := \lambda y.(t\{s/x\}) \quad (t't)\{s/x\} := t'\{s/x\}t\{s/x\}$$

The structure of a term

Rmk. Every term s can be written in a unique way as

$$s = \lambda x_1 \dots \lambda x_n . h t_1 \dots t_m \qquad \text{with } m, n \in \mathbb{N}$$

where h (the head of s) is either a variable (head variable) or a β-redex (head β-redex).

In a tree representation (abstract syntax tree):

Compare this tree with a derivation in natural deduction. Similarities? Differences?

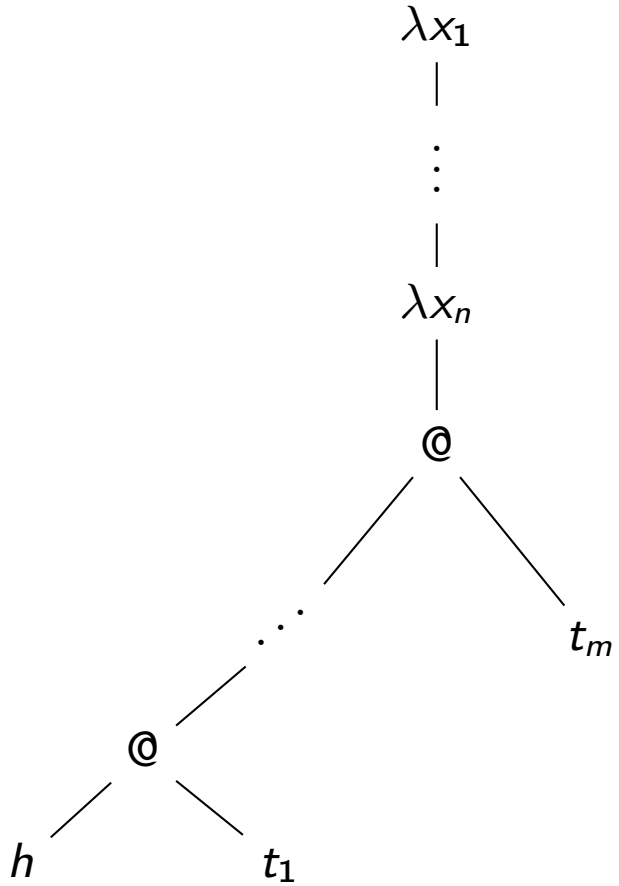
The structure of a term

Rmk. Every term s can be written in a unique way as

$$s = \lambda x_1 \dots \lambda x_n . h t_1 \dots t_m \qquad \text{with } m, n \in \mathbb{N}$$

where h (the head of s) is either a variable (head variable) or a β-redex (head β-redex).

In a tree representation (abstract syntax tree):



Compare this tree with a derivation in natural deduction. Similarities? Differences?

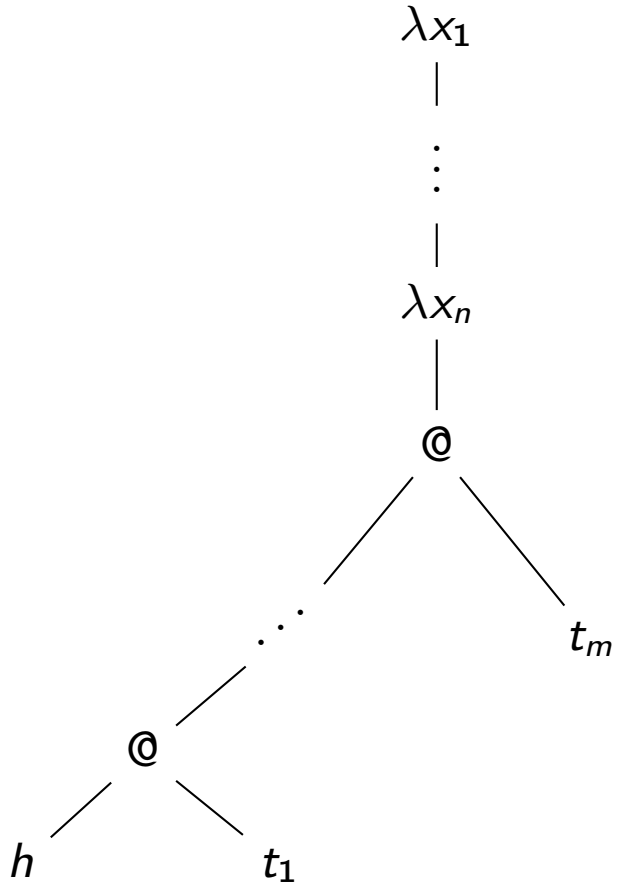
The structure of a term

Rmk. Every term s can be written in a unique way as

$$s = \lambda x_1 \dots \lambda x_n . h t_1 \dots t_m \qquad \text{with } m, n \in \mathbb{N}$$

where h (the head of s) is either a variable (head variable) or a β-redex (head β-redex).

In a tree representation (abstract syntax tree):



Compare this tree with a derivation in natural deduction. Similarities? Differences?

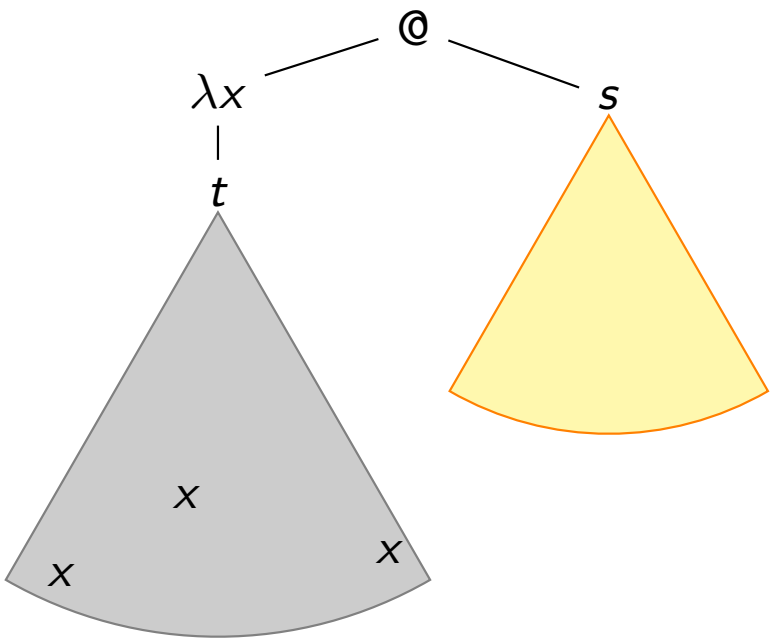
β-reduction from a graphical point of view

$$(\lambda x.t)s \to_\beta t\{s/x\}$$

Compare this figure with the cut-elimination step for natural deduction (see Day 1).

β-reduction from a graphical point of view

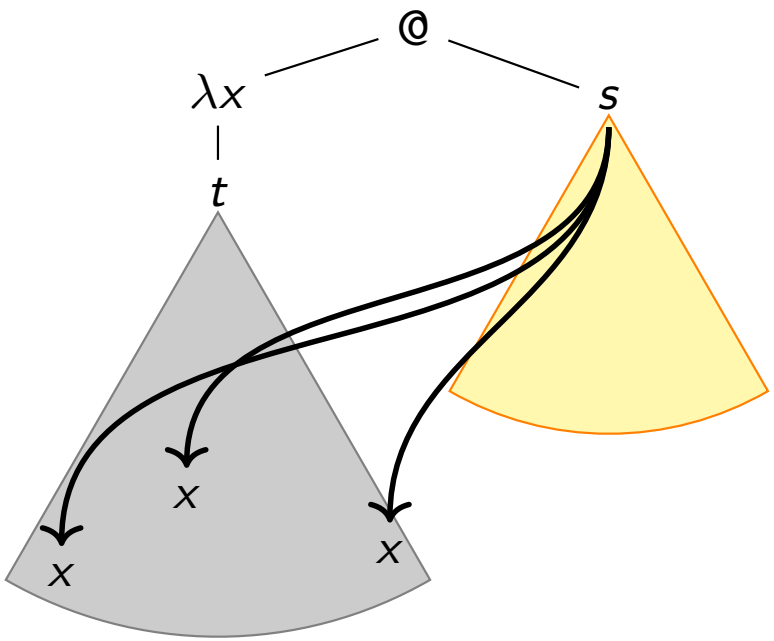
$$(\lambda x.t)s \to_\beta t\{s/x\}$$



Compare this figure with the cut-elimination step for natural deduction (see Day 1).

β-reduction from a graphical point of view

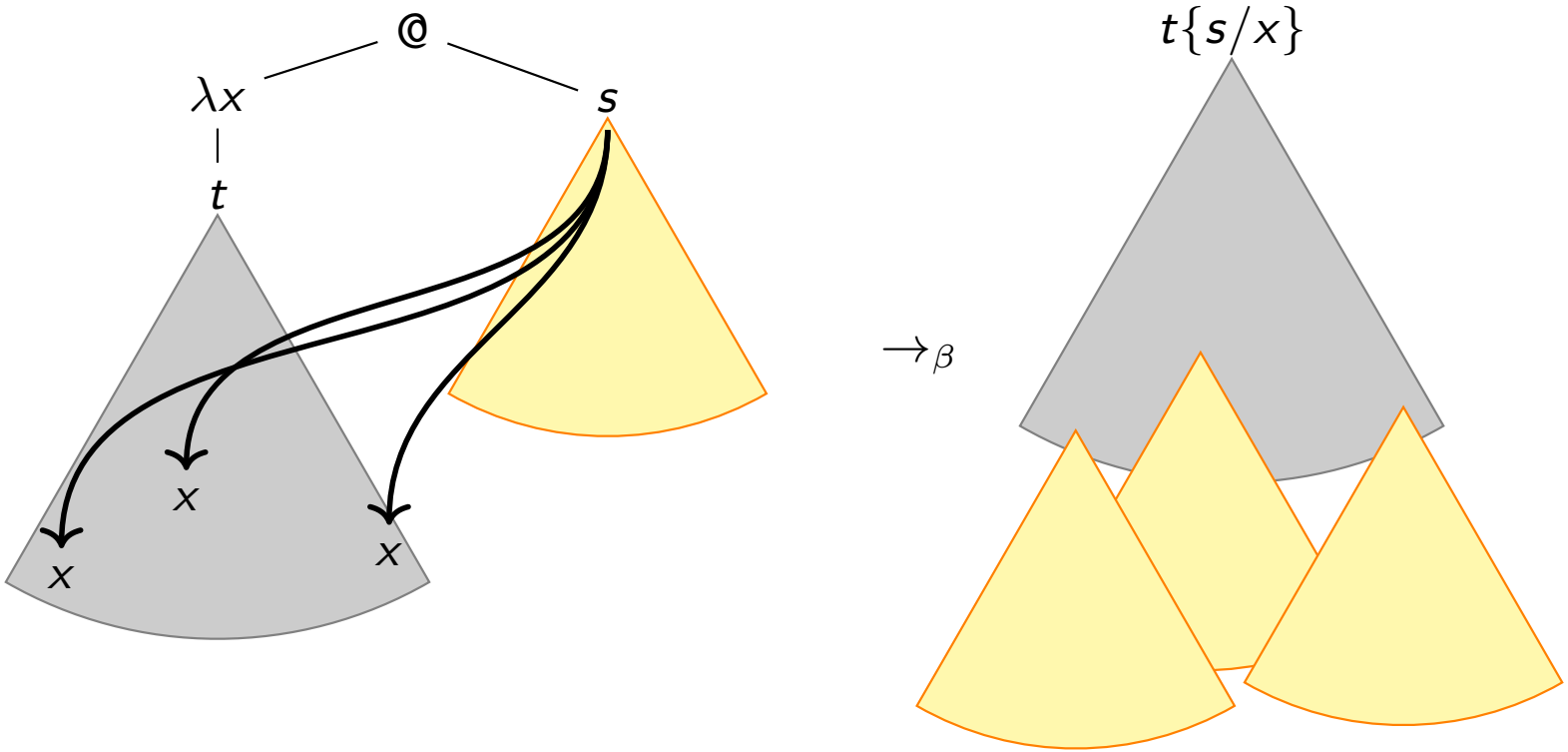
$$(\lambda x.t)s \to_\beta t\{s/x\}$$



Compare this figure with the cut-elimination step for natural deduction (see Day 1).

β-reduction from a graphical point of view

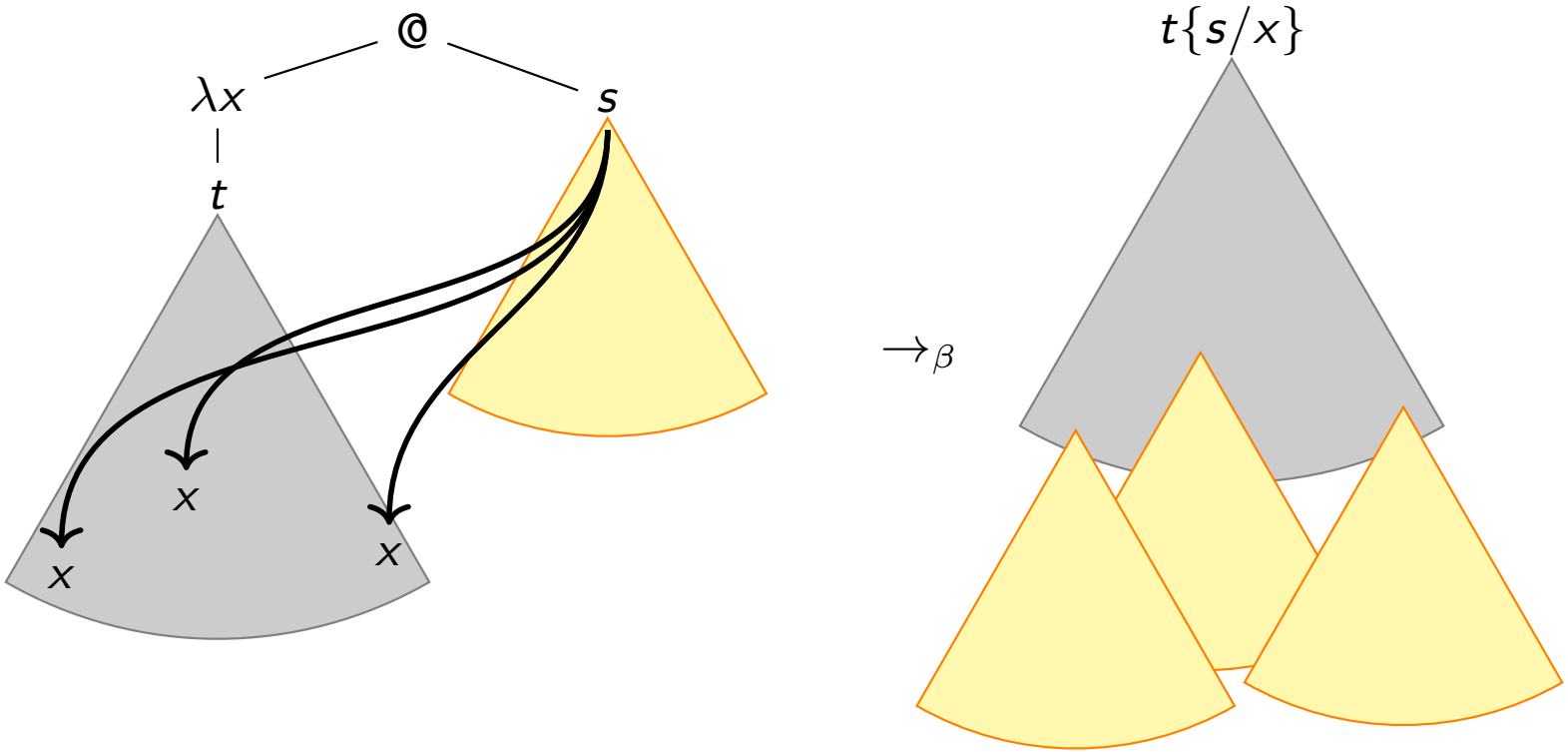
$$(\lambda x.t)s \to_\beta t\{s/x\}$$



Compare this figure with the cut-elimination step for natural deduction (see Day 1).

β-reduction from a graphical point of view

$$(\lambda x.t)s \to_\beta t\{s/x\}$$



Compare this figure with the cut-elimination step for natural deduction (see Day 1).

Different notions of reduction

(Full) β-reduction $\to_\beta$ fires a β-redex anywhere in a term. Formally,

$$\frac{}{(\lambda x.t)s \to_\beta t\{s/x\}} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{t \to_\beta t'}{ts \to_\beta t's} \qquad \frac{t \to_\beta t'}{st \to_\beta st'}$$

Head β-reduction $\to_{h\beta}$ fires a β-redex only in the "head" of a term. Formally,

$$\frac{}{(\lambda x.t)s \to_{h\beta} t\{s/x\}} \qquad \frac{t \to_{h\beta} t'}{\lambda x.t \to_{h\beta} \lambda x.t'} \qquad \frac{t \to_{h\beta} t' \qquad t \neq \lambda x.r}{ts \to_{h\beta} t's}$$

Leftmost-outermost β-reduction $\to_{\ell\beta}$ fires the leftmost-outermost β-redex in a term.

$$\frac{}{(\lambda x.t)s \to_{\ell\beta} t\{s/x\}} \qquad \frac{t \to_{\ell\beta} t'}{\lambda x.t \to_{\ell\beta} \lambda x.t'} \qquad \frac{t \to_{\ell\beta} t' \qquad t \neq \lambda x.r}{ts \to_{\ell\beta} t's} \qquad \frac{t \to_{\ell\beta} t' \qquad s \text{ neutral}}{st \to_{\ell\beta} st'}$$

where neutral means $s = xs_1 \dots s_n$ and $s_1, \dots, s_n$ normal, for some $n \in \mathbb{N}$.

Rmk. $\to_{h\beta} \subsetneq \to_{I\beta} \subsetneq \to_\beta$. For strictness, consider $I = \lambda x.x$ and $t = (Ix)(Iy)(Iz)$. Then,

- $t \to_{h\beta} x(Iy)(Iz)$ but $t \not\to_{h\beta} (Ix)y(Iz)$ and $t \not\to_{h\beta} (Ix)(Iy)z$;
- $x(Iy)(Iz) \to_{\ell\beta} xy(Iz)$ but $x(Iy)(Iz) \not\to_{\ell\beta} x(Iy)z$;
- $t \to_\beta (Ix)(Iy)z$ and $x(Iy)(Iz) \to_\beta x(Iy)z$.

Different notions of reduction

(Full) β-reduction $\to_\beta$ fires a β-redex anywhere in a term. Formally,

$$\frac{}{(\lambda x.t)s \to_\beta t\{s/x\}} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{t \to_\beta t'}{ts \to_\beta t's} \qquad \frac{t \to_\beta t'}{st \to_\beta st'}$$

Head β-reduction $\to_{h\beta}$ fires a β-redex only in the "head" of a term. Formally,

$$\frac{}{(\lambda x.t)s \to_{h\beta} t\{s/x\}} \qquad \frac{t \to_{h\beta} t'}{\lambda x.t \to_{h\beta} \lambda x.t'} \qquad \frac{t \to_{h\beta} t' \qquad t \neq \lambda x.r}{ts \to_{h\beta} t's}$$

Leftmost-outermost β-reduction $\to_{\ell\beta}$ fires the leftmost-outermost β-redex in a term.

$$\frac{}{(\lambda x.t)s \to_{\ell\beta} t\{s/x\}} \qquad \frac{t \to_{\ell\beta} t'}{\lambda x.t \to_{\ell\beta} \lambda x.t'} \qquad \frac{t \to_{\ell\beta} t' \qquad t \neq \lambda x.r}{ts \to_{\ell\beta} t's} \qquad \frac{t \to_{\ell\beta} t' \qquad s \text{ neutral}}{st \to_{\ell\beta} st'}$$

where neutral means $s = xs_1 \dots s_n$ and $s_1, \dots, s_n$ normal, for some $n \in \mathbb{N}$.

Rmk. $\to_{h\beta} \subsetneq \to_{I\beta} \subsetneq \to_\beta$. For strictness, consider $I = \lambda x.x$ and $t = (Ix)(Iy)(Iz)$. Then,

- $t \to_{h\beta} x(Iy)(Iz)$ but $t \not\to_{h\beta} (Ix)y(Iz)$ and $t \not\to_{h\beta} (Ix)(Iy)z$;
- $x(Iy)(Iz) \to_{\ell\beta} xy(Iz)$ but $x(Iy)(Iz) \not\to_{\ell\beta} x(Iy)z$;
- $t \to_\beta (Ix)(Iy)z$ and $x(Iy)(Iz) \to_\beta x(Iy)z$.

Different notions of reduction

(Full) β-reduction $\to_\beta$ fires a β-redex anywhere in a term. Formally,

$$\frac{}{(\lambda x.t)s \to_\beta t\{s/x\}} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{t \to_\beta t'}{ts \to_\beta t's} \qquad \frac{t \to_\beta t'}{st \to_\beta st'}$$

Head β-reduction $\to_{h\beta}$ fires a β-redex only in the "head" of a term. Formally,

$$\frac{}{(\lambda x.t)s \to_{h\beta} t\{s/x\}} \qquad \frac{t \to_{h\beta} t'}{\lambda x.t \to_{h\beta} \lambda x.t'} \qquad \frac{t \to_{h\beta} t' \qquad t \neq \lambda x.r}{ts \to_{h\beta} t's}$$

Leftmost-outermost β-reduction $\to_{\ell\beta}$ fires the leftmost-outermost β-redex in a term.

$$\frac{}{(\lambda x.t)s \to_{\ell\beta} t\{s/x\}} \qquad \frac{t \to_{\ell\beta} t'}{\lambda x.t \to_{\ell\beta} \lambda x.t'} \qquad \frac{t \to_{\ell\beta} t' \qquad t \neq \lambda x.r}{ts \to_{\ell\beta} t's} \qquad \frac{t \to_{\ell\beta} t' \qquad s \text{ neutral}}{st \to_{\ell\beta} st'}$$

where neutral means $s = xs_1 \dots s_n$ and $s_1, \dots, s_n$ normal, for some $n \in \mathbb{N}$.

Rmk. $\to_{h\beta} \subsetneq \to_{l\beta} \subsetneq \to_\beta$. For strictness, consider $I = \lambda x.x$ and $t = (Ix)(Iy)(Iz)$. Then,

- $t \to_{h\beta} x(Iy)(Iz)$ but $t \not\to_{h\beta} (Ix)y(Iz)$ and $t \not\to_{h\beta} (Ix)(Iy)z$;
- $x(Iy)(Iz) \to_{\ell\beta} xy(Iz)$ but $x(Iy)(Iz) \not\to_{\ell\beta} x(Iy)z$;
- $t \to_\beta (Ix)(Iy)z$ and $x(Iy)(Iz) \to_\beta x(Iy)z$.

Different notions of reduction

(Full) β-reduction $\to_\beta$ fires a β-redex anywhere in a term. Formally,

$$\frac{}{(\lambda x.t)s \to_\beta t\{s/x\}} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{t \to_\beta t'}{ts \to_\beta t's} \qquad \frac{t \to_\beta t'}{st \to_\beta st'}$$

Head β-reduction $\to_{h\beta}$ fires a β-redex only in the "head" of a term. Formally,

$$\frac{}{(\lambda x.t)s \to_{h\beta} t\{s/x\}} \qquad \frac{t \to_{h\beta} t'}{\lambda x.t \to_{h\beta} \lambda x.t'} \qquad \frac{t \to_{h\beta} t' \qquad t \neq \lambda x.r}{ts \to_{h\beta} t's}$$

Leftmost-outermost β-reduction $\to_{\ell\beta}$ fires the leftmost-outermost β-redex in a term.

$$\frac{}{(\lambda x.t)s \to_{\ell\beta} t\{s/x\}} \qquad \frac{t \to_{\ell\beta} t'}{\lambda x.t \to_{\ell\beta} \lambda x.t'} \qquad \frac{t \to_{\ell\beta} t' \qquad t \neq \lambda x.r}{ts \to_{\ell\beta} t's} \qquad \frac{t \to_{\ell\beta} t' \quad s \text{ neutral}}{st \to_{\ell\beta} st'}$$

where neutral means $s = xs_1 \dots s_n$ and $s_1, \dots, s_n$ normal, for some $n \in \mathbb{N}$.

Rmk. $\to_{h\beta} \subsetneq \to_{I\beta} \subsetneq \to_\beta$. For strictness, consider $I = \lambda x.x$ and $t = (Ix)(Iy)(Iz)$. Then,

- $t \to_{h\beta} x(Iy)(Iz)$ but $t \not\to_{h\beta} (Ix)y(Iz)$ and $t \not\to_{h\beta} (Ix)(Iy)z$;
- $x(Iy)(Iz) \to_{\ell\beta} xy(Iz)$ but $x(Iy)(Iz) \not\to_{\ell\beta} x(Iy)z$;
- $t \to_\beta (Ix)(Iy)z$ and $x(Iy)(Iz) \to_\beta x(Iy)z$.

Different notions of reduction

(Full) β-reduction $\to_\beta$ fires a β-redex anywhere in a term. Formally,

$$\frac{}{(\lambda x.t)s \to_\beta t\{s/x\}} \qquad \frac{t \to_\beta t'}{\lambda x.t \to_\beta \lambda x.t'} \qquad \frac{t \to_\beta t'}{ts \to_\beta t's} \qquad \frac{t \to_\beta t'}{st \to_\beta st'}$$

Head β-reduction $\to_{h\beta}$ fires a β-redex only in the "head" of a term. Formally,

$$\frac{}{(\lambda x.t)s \to_{h\beta} t\{s/x\}} \qquad \frac{t \to_{h\beta} t'}{\lambda x.t \to_{h\beta} \lambda x.t'} \qquad \frac{t \to_{h\beta} t' \qquad t \neq \lambda x.r}{ts \to_{h\beta} t's}$$

Leftmost-outermost β-reduction $\to_{\ell\beta}$ fires the leftmost-outermost β-redex in a term.

$$\frac{}{(\lambda x.t)s \to_{\ell\beta} t\{s/x\}} \qquad \frac{t \to_{\ell\beta} t'}{\lambda x.t \to_{\ell\beta} \lambda x.t'} \qquad \frac{t \to_{\ell\beta} t' \qquad t \neq \lambda x.r}{ts \to_{\ell\beta} t's} \qquad \frac{t \to_{\ell\beta} t' \quad s \text{ neutral}}{st \to_{\ell\beta} st'}$$

where neutral means $s = xs_1 \dots s_n$ and $s_1, \dots, s_n$ normal, for some $n \in \mathbb{N}$.

Rmk. $\to_{h\beta} \subsetneq \to_{I\beta} \subsetneq \to_\beta$. For strictness, consider $I = \lambda x.x$ and $t = (Ix)(Iy)(Iz)$. Then,

- $t \to_{h\beta} x(Iy)(Iz)$ but $t \not\to_{h\beta} (Ix)y(Iz)$ and $t \not\to_{h\beta} (Ix)(Iy)z$;
- $x(Iy)(Iz) \to_{\ell\beta} xy(Iz)$ but $x(Iy)(Iz) \not\to_{\ell\beta} x(Iy)z$;
- $t \to_\beta (Ix)(Iy)z$ and $x(Iy)(Iz) \to_\beta x(Iy)z$.

Properties of different reductions

Rmk. Reductions $\to_{h\beta}$ and $\to_{\ell\beta}$ are deterministic (they can fire at most one redex). So:

$$\text{If } t \to_r s_1 \text{ and } t \to_r s_2 \text{ then } s_1 = s_2, \text{ for } r \in \{h\beta, l\beta\}.$$

Reduction $\to_\beta$ is not deterministic, it chooses among several β-redexes to fire in a term.

Notation. $t \to^* s$ means that $t = t_0 \overbrace{\to t_1 \to \cdots \to} t_n = s$ (in particular, $t = s$ for $n = 0$).

for some $n \in \mathbb{N}$

Theorem (Confluence)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$, then there is a term r such that $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$.

Def. Let $r \in \{\beta, \ell\beta, h\beta\}$. A term t is r-normal if there is no s such that $t \to_r s$.

Corollary (Uniqueness of normal form)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$ where s_1 and s_2 are β-normal, then $s_1 = s_2$.

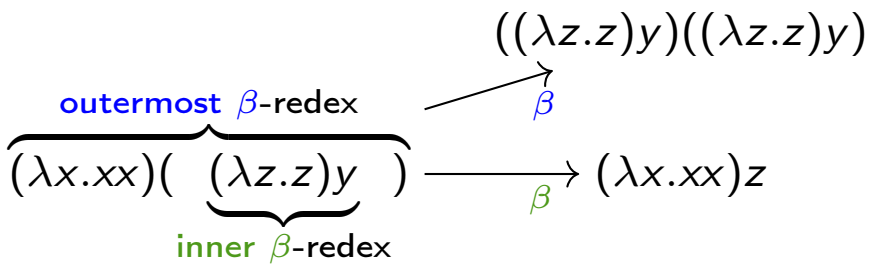
Proof. By confluence, $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$ for some r. By normality, $s_1 = r = s_2$. □

Properties of different reductions

Rmk. Reductions $\to_{h\beta}$ and $\to_{\ell\beta}$ are deterministic (they can fire at most one redex). So:

$$\text{If } t \to_r s_1 \text{ and } t \to_r s_2 \text{ then } s_1 = s_2, \text{ for } r \in \{h\beta, l\beta\}.$$

Reduction $\to_\beta$ is not deterministic, it chooses among several β-redexes to fire in a term.



Notation. $t \to^* s$ means that $t = t_0 \overbrace{\to t_1 \to \cdots \to}^{\text{for some } n\in\mathbb{N}} t_n = s$ (in particular, $t = s$ for $n = 0$).

Theorem (Confluence)

If $t \to^*_\beta s_1$ and $t \to^*_\beta s_2$, then there is a term r such that $s_1 \to^*_\beta r$ and $s_2 \to^*_\beta r$.

Def. Let $r \in \{\beta, \ell\beta, h\beta\}$. A term t is r-normal if there is no s such that $t \to_r s$.

Corollary (Uniqueness of normal form)

If $t \to^*_\beta s_1$ and $t \to^*_\beta s_2$ where s_1 and s_2 are β-normal, then $s_1 = s_2$.

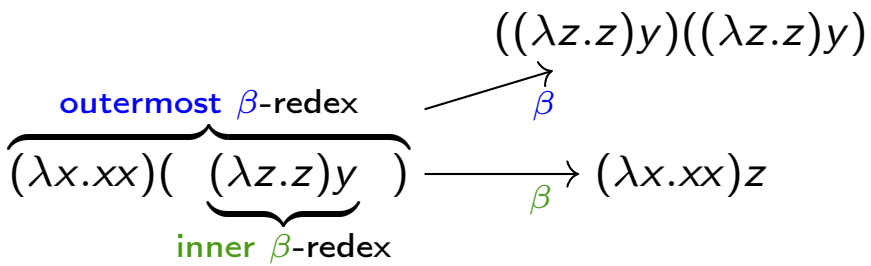
Proof. By confluence, $s_1 \to^*_\beta r$ and $s_2 \to^*_\beta r$ for some r. By normality, $s_1 = r = s_2$. □

Properties of different reductions

Rmk. Reductions $\to_{h\beta}$ and $\to_{\ell\beta}$ are deterministic (they can fire at most one redex). So:

$$\text{If } t \to_r s_1 \text{ and } t \to_r s_2 \text{ then } s_1 = s_2, \text{ for } r \in \{h\beta, l\beta\}.$$

Reduction $\to_\beta$ is not deterministic, it chooses among several β-redexes to fire in a term.



Notation. $t \to^* s$ means that $t = t_0 \overbrace{\to t_1 \to \cdots \to} t_n = s$ (in particular, $t = s$ for $n = 0$).

for some $n \in \mathbb{N}$

Theorem (Confluence)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$, then there is a term r such that $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$.

Def. Let $r \in \{\beta, \ell\beta, h\beta\}$. A term t is r-normal if there is no s such that $t \to_r s$.

Corollary (Uniqueness of normal form)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$ where s_1 and s_2 are β-normal, then $s_1 = s_2$.

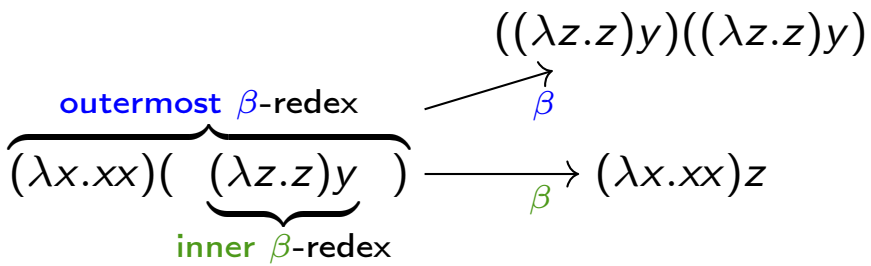
Proof. By confluence, $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$ for some r. By normality, $s_1 = r = s_2$. □

Properties of different reductions

Rmk. Reductions $\to_{h\beta}$ and $\to_{\ell\beta}$ are deterministic (they can fire at most one redex). So:

$$\text{If } t \to_r s_1 \text{ and } t \to_r s_2 \text{ then } s_1 = s_2, \text{ for } r \in \{h\beta, l\beta\}.$$

Reduction $\to_\beta$ is not deterministic, it chooses among several β-redexes to fire in a term.



Notation. $t \to^* s$ means that $t = t_0 \overbrace{\to t_1 \to \cdots \to} t_n = s$ (in particular, $t = s$ for $n = 0$).

for some $n \in \mathbb{N}$

Theorem (Confluence)

If $t \to^*_\beta s_1$ and $t \to^*_\beta s_2$, then there is a term r such that $s_1 \to^*_\beta r$ and $s_2 \to^*_\beta r$.

Def. Let $r \in \{\beta, \ell\beta, h\beta\}$. A term t is r-normal if there is no s such that $t \to_r s$.

Corollary (Uniqueness of normal form)

If $t \to^*_\beta s_1$ and $t \to^*_\beta s_2$ where s_1 and s_2 are β-normal, then $s_1 = s_2$.

Proof. By confluence, $s_1 \to^*_\beta r$ and $s_2 \to^*_\beta r$ for some r. By normality, $s_1 = r = s_2$. □

Normalization, strong normalization and divergence

Def. Let t be a term and $r \in \{\beta, \ell\beta, h\beta\}$.

1. t is *r*-normalizing if there is a r-normal term s such that $t \to_r^* s$.
2. t is strongly *r*-normalizing if there is no $(t_i)_{i\in\mathbb{N}}$ such that $t = t_0$ and $t_i \to_r t_{i+1}$.

Ex. Every β-normal form is strongly β-normalizing. Let $\delta = \lambda x.xx$.

- $\delta\delta$ is not β-normalizing: if $\delta\delta \to_\beta t$ then $t = \delta\delta$.
- $(\lambda x.y)(\delta\delta)$ is β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta y$ which is β-normal) but not strongly β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta (\lambda x.y)(\delta\delta) \to_\beta \dots$).

Rmk. Strong normalization implies normalization, but the converse fails, see above.

Rmk. Strong normalization and normalization coincide for $\to_{h\beta}$ and $\to_{I\beta}$, not for $\to_\beta$.

Rmk. In the simply typed λ-calculus, every term is β-normalizing (actually, strongly).

Normalization, strong normalization and divergence

Def. Let t be a term and $r \in \{\beta, \ell\beta, h\beta\}$.

1. t is *r*-normalizing if there is a r-normal term s such that $t \to_r^* s$.
2. t is strongly *r*-normalizing if there is no $(t_i)_{i\in\mathbb{N}}$ such that $t = t_0$ and $t_i \to_r t_{i+1}$.

Ex. Every β-normal form is strongly β-normalizing. Let $\delta = \lambda x.xx$.

- $\delta\delta$ is not β-normalizing: if $\delta\delta \to_\beta t$ then $t = \delta\delta$.
- $(\lambda x.y)(\delta\delta)$ is β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta y$ which is β-normal) but not strongly β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta (\lambda x.y)(\delta\delta) \to_\beta \dots$).

Rmk. Strong normalization implies normalization, but the converse fails, see above.

Rmk. Strong normalization and normalization coincide for $\to_{h\beta}$ and $\to_{l\beta}$, not for $\to_\beta$.

Rmk. In the simply typed λ-calculus, every term is β-normalizing (actually, strongly).

Normalization, strong normalization and divergence

Def. Let t be a term and $r \in \{\beta, \ell\beta, h\beta\}$.

1. t is *r*-normalizing if there is a r-normal term s such that $t \to_r^* s$.
2. t is strongly *r*-normalizing if there is no $(t_i)_{i\in\mathbb{N}}$ such that $t = t_0$ and $t_i \to_r t_{i+1}$.

Ex. Every β-normal form is strongly β-normalizing. Let $\delta = \lambda x.xx$.

- $\delta\delta$ is not β-normalizing: if $\delta\delta \to_\beta t$ then $t = \delta\delta$.
- $(\lambda x.y)(\delta\delta)$ is β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta y$ which is β-normal) but not strongly β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta (\lambda x.y)(\delta\delta) \to_\beta \dots$).

Rmk. Strong normalization implies normalization, but the converse fails, see above.

Rmk. Strong normalization and normalization coincide for $\to_{h\beta}$ and $\to_{I\beta}$, not for $\to_\beta$.

Rmk. In the simply typed λ-calculus, every term is β-normalizing (actually, strongly).

Normalization, strong normalization and divergence

Def. Let t be a term and $r \in \{\beta, \ell\beta, h\beta\}$.

1. t is *r*-normalizing if there is a r-normal term s such that $t \to_r^* s$.
2. t is strongly *r*-normalizing if there is no $(t_i)_{i\in\mathbb{N}}$ such that $t = t_0$ and $t_i \to_r t_{i+1}$.

Ex. Every β-normal form is strongly β-normalizing. Let $\delta = \lambda x.xx$.

- $\delta\delta$ is not β-normalizing: if $\delta\delta \to_\beta t$ then $t = \delta\delta$.
- $(\lambda x.y)(\delta\delta)$ is β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta y$ which is β-normal) but not strongly β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta (\lambda x.y)(\delta\delta) \to_\beta \dots$).

Rmk. Strong normalization implies normalization, but the converse fails, see above.

Rmk. Strong normalization and normalization coincide for $\to_{h\beta}$ and $\to_{I\beta}$, not for $\to_\beta$.

Rmk. In the simply typed λ-calculus, every term is β-normalizing (actually, strongly).

Normalization, strong normalization and divergence

Def. Let t be a term and $r \in \{\beta, \ell\beta, h\beta\}$.

1. t is *r*-normalizing if there is a r-normal term s such that $t \to_r^* s$.
2. t is strongly *r*-normalizing if there is no $(t_i)_{i\in\mathbb{N}}$ such that $t = t_0$ and $t_i \to_r t_{i+1}$.

Ex. Every β-normal form is strongly β-normalizing. Let $\delta = \lambda x.xx$.

- $\delta\delta$ is not β-normalizing: if $\delta\delta \to_\beta t$ then $t = \delta\delta$.
- $(\lambda x.y)(\delta\delta)$ is β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta y$ which is β-normal) but not strongly β-normalizing (indeed $(\lambda x.y)(\delta\delta) \to_\beta (\lambda x.y)(\delta\delta) \to_\beta \dots$).

Rmk. Strong normalization implies normalization, but the converse fails, see above.

Rmk. Strong normalization and normalization coincide for $\to_{h\beta}$ and $\to_{I\beta}$, not for $\to_\beta$.

Rmk. In the simply typed λ-calculus, every term is β-normalizing (actually, strongly).

Fixed point combinator

Def. A fixed point of a term t is a term s such that $s \to_\beta^* ts$.
A fixed point combinator is a term Y such that Yt is a fixed point of t, for every term t.

Proposition (Fixed point combinator)

Let $A = \lambda a.\lambda f.f(aaf)$ and $\Theta = AA$. Then, Θ is a fixed point combinator.

Proof. $\Theta = (\lambda a.\lambda f.f(aaf))A \to_{h\beta} \lambda f.f(AAf) = \lambda f.f(\Theta f)$. Therefore, for every term t,

$$\Theta t \to_{h\beta} (\lambda f.f(\Theta f))t \to_{h\beta} t(\Theta t). \qquad \square$$

Rmk. Θ is $h\beta$-normalizing but not β-normalizing: $\Theta \to_{h\beta} \lambda f.f(\Theta f)$ which is $h\beta$-normal, but $\Theta \to_\beta \lambda f.f(\Theta f) \to_\beta \lambda f.f(f(\Theta f)) \to_\beta \dots$ and there are not other β-reduction paths.

Rmk. Θ is not a term of the simply typed λ-calculus, because of the subterm aa.

Rmk. Fixed point combinators such has Θ are crucial to represent recursive functions.

Fixed point combinator

Def. A fixed point of a term t is a term s such that $s \to_\beta^* ts$.
A fixed point combinator is a term Y such that Yt is a fixed point of t, for every term t.

Proposition (Fixed point combinator)

Let $A = \lambda a.\lambda f.f(aaf)$ and $\Theta = AA$. Then, Θ is a fixed point combinator.

Proof. $\Theta = (\lambda a.\lambda f.f(aaf))A \to_{h\beta} \lambda f.f(AAf) = \lambda f.f(\Theta f)$. Therefore, for every term t,

$$\Theta t \to_{h\beta} (\lambda f.f(\Theta f))t \to_{h\beta} t(\Theta t).$$ □

Rmk. Θ is $h\beta$-normalizing but not β-normalizing: $\Theta \to_{h\beta} \lambda f.f(\Theta f)$ which is $h\beta$-normal, but $\Theta \to_\beta \lambda f.f(\Theta f) \to_\beta \lambda f.f(f(\Theta f)) \to_\beta \dots$ and there are not other β-reduction paths.

Rmk. Θ is not a term of the simply typed λ-calculus, because of the subterm aa.

Rmk. Fixed point combinators such has Θ are crucial to represent recursive functions.

Fixed point combinator

Def. A fixed point of a term t is a term s such that $s \to_\beta^* ts$.
A fixed point combinator is a term Y such that Yt is a fixed point of t, for every term t.

Proposition (Fixed point combinator)

Let $A = \lambda a.\lambda f.f(aaf)$ and $\Theta = AA$. Then, Θ is a fixed point combinator.

Proof. $\Theta = (\lambda a.\lambda f.f(aaf))A \to_{h\beta} \lambda f.f(AAf) = \lambda f.f(\Theta f)$. Therefore, for every term t,

$$\Theta t \to_{h\beta} (\lambda f.f(\Theta f))t \to_{h\beta} t(\Theta t).$$ □

Rmk. Θ is $h\beta$-normalizing but not β-normalizing: $\Theta \to_{h\beta} \lambda f.f(\Theta f)$ which is $h\beta$-normal, but $\Theta \to_\beta \lambda f.f(\Theta f) \to_\beta \lambda f.f(f(\Theta f)) \to_\beta \dots$ and there are not other β-reduction paths.

Rmk. Θ is not a term of the simply typed λ-calculus, because of the subterm aa.

Rmk. Fixed point combinators such has Θ are crucial to represent recursive functions.

Simply typed versus untyped

The simply typed λ-calculus (in Curry-style) is a restriction of the untyped λ-calculus
$\rightsquigarrow$ the latter just take terms and β-reduction from the former without checking typability.

But the untyped λ-calculus can also be seen as a "special case" of the simply type one.
Consider that the simple types are generated by only one ground type X.

Def. Let $\equiv$ be the least congruence on simple types generated by $X \equiv X \Rightarrow X$, that is:

$$\frac{}{X \equiv X} \qquad \frac{A \equiv B}{B \equiv A} \qquad \frac{A \equiv B \quad B \equiv C}{A \equiv C} \qquad \frac{}{X \equiv X \Rightarrow X} \qquad \frac{A \equiv A' \quad B \equiv B'}{A \Rightarrow B \equiv A' \Rightarrow B'}$$

Rmk. $A \equiv X$ for every simple type A (proof by induction on A) $\rightsquigarrow$ All types are the same!

Proposition (Untyped = simply typed + recursive type identity $\equiv$)

Every untyped term is typable in Curry's simply typed λ-calculus extended with the rule:

$$\frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B} \equiv$$

Proof. By straightforward induction on t (exercise!). □

Simply typed versus untyped

The simply typed λ-calculus (in Curry-style) is a restriction of the untyped λ-calculus $\rightsquigarrow$ the latter just take terms and β-reduction from the former without checking typability.

But the untyped λ-calculus can also be seen as a "special case" of the simply type one. Consider that the simple types are generated by only one ground type X.

Def. Let $\equiv$ be the least congruence on simple types generated by $X \equiv X \Rightarrow X$, that is:

$$\frac{}{X \equiv X} \qquad \frac{A \equiv B}{B \equiv A} \qquad \frac{A \equiv B \quad B \equiv C}{A \equiv C} \qquad \frac{}{X \equiv X \Rightarrow X} \qquad \frac{A \equiv A' \quad B \equiv B'}{A \Rightarrow B \equiv A' \Rightarrow B'}$$

Rmk. $A \equiv X$ for every simple type A (proof by induction on A) $\rightsquigarrow$ All types are the same!

Proposition (Untyped = simply typed + recursive type identity $\equiv$)

Every untyped term is typable in Curry's simply typed λ-calculus extended with the rule:

$$\frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B} \equiv$$

Proof. By straightforward induction on t (exercise!). □

Simply typed versus untyped

The simply typed λ-calculus (in Curry-style) is a restriction of the untyped λ-calculus $\rightsquigarrow$ the latter just take terms and β-reduction from the former without checking typability.

But the untyped λ-calculus can also be seen as a "special case" of the simply type one. Consider that the simple types are generated by only one ground type X.

Def. Let $\equiv$ be the least congruence on simple types generated by $X \equiv X \Rightarrow X$, that is:

$$\frac{}{X \equiv X} \qquad \frac{A \equiv B}{B \equiv A} \qquad \frac{A \equiv B \quad B \equiv C}{A \equiv C} \qquad \frac{}{X \equiv X \Rightarrow X} \qquad \frac{A \equiv A' \quad B \equiv B'}{A \Rightarrow B \equiv A' \Rightarrow B'}$$

Rmk. $A \equiv X$ for every simple type A (proof by induction on A) $\rightsquigarrow$ All types are the same!

Proposition (Untyped = simply typed + recursive type identity $\equiv$)

Every untyped term is typable in Curry's simply typed λ-calculus extended with the rule:

$$\frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B} \equiv$$

Proof. By straightforward induction on t (exercise!). □

Outline

1. The syntax and the operational semantics of the untyped λ-calculus
2. The proof of confluence of β-reduction
3. The notions of β-equivalence and $\beta\eta$-equivalence
4. Conclusion, exercises and bibliography

The challenges of proving confluence of β-reduction

An ARS $(A, \to)$ is confluent when for all r_1, r_2, if $r_1 \,{}^*_\beta\!\leftarrow \cdot \to^*_\beta r_2$ then $r_1 \to^*_\beta \cdot \,{}^*_\beta\!\leftarrow r_2$.

In Lecture 3, we proved that $(\mathrm{STLC}, \to_\beta)$ is confluent using Newman's lemma, as it is:

- strongly normalizing (SN), and
- weakly confluent (for all t, r_1, r_2, if $r_1 \,{}_\beta\!\leftarrow t \to_\beta r_2$ then $r_1 \to^*_\beta s \,{}^*_\beta\!\leftarrow r_2$ for some s).

Weak confluence is a local property, which is easy to prove by just inspecting all possible cases where $r_1 \leftarrow t \to r_2$: it could be long and tedious, but there are finitely many cases.

SN is crucial, as there are weakly confluent but non-confluent ARSs (Ex. 2, Lecture 3).

Unfortunately, the untyped λ-calculus is not SN! $\rightsquigarrow$ Another proof strategy is needed.

Confluence is a global property with infinitely many cases to check! $\rightsquigarrow$ A difficult proof.

The challenges of proving confluence of β-reduction

An ARS $(A, \to)$ is confluent when for all r_1, r_2, if $r_1 \,{}^*_\beta\!\leftarrow \cdot \to^*_\beta r_2$ then $r_1 \to^*_\beta \cdot \,{}^*_\beta\!\leftarrow r_2$.

In Lecture 3, we proved that $(\mathrm{STLC}, \to_\beta)$ is confluent using Newman's lemma, as it is:

- strongly normalizing (SN), and
- weakly confluent (for all t, r_1, r_2, if $r_1 \,{}_\beta\!\leftarrow t \to_\beta r_2$ then $r_1 \to^*_\beta s \,{}^*_\beta\!\leftarrow r_2$ for some s).

Weak confluence is a local property, which is easy to prove by just inspecting all possible cases where $r_1 \leftarrow t \to r_2$: it could be long and tedious, but there are finitely many cases.

SN is crucial, as there are weakly confluent but non-confluent ARSs (Ex. 2, Lecture 3).

Unfortunately, the untyped λ-calculus is not SN! $\rightsquigarrow$ Another proof strategy is needed.

Confluence is a global property with infinitely many cases to check! $\rightsquigarrow$ A difficult proof.

The challenges of proving confluence of β-reduction

An ARS $(A, \to)$ is confluent when for all r_1, r_2, if $r_1 \,{}^*_\beta\!\leftarrow \cdot \to^*_\beta r_2$ then $r_1 \to^*_\beta \cdot \,{}^*_\beta\!\leftarrow r_2$.

In Lecture 3, we proved that $(\mathrm{STLC}, \to_\beta)$ is confluent using Newman's lemma, as it is:

- strongly normalizing (SN), and
- weakly confluent (for all t, r_1, r_2, if $r_1 \,{}_\beta\!\leftarrow t \to_\beta r_2$ then $r_1 \to^*_\beta s \,{}^*_\beta\!\leftarrow r_2$ for some s).

Weak confluence is a local property, which is easy to prove by just inspecting all possible cases where $r_1 \leftarrow t \to r_2$: it could be long and tedious, but there are finitely many cases.

SN is crucial, as there are weakly confluent but non-confluent ARSs (Ex. 2, Lecture 3).

Unfortunately, the untyped λ-calculus is not SN! $\rightsquigarrow$ Another proof strategy is needed.

Confluence is a global property with infinitely many cases to check! $\rightsquigarrow$ A difficult proof.

A tool to prove confluence: parallel β-reduction

Def. The parallel β-reduction $\Rightarrow_\beta$ on λ-terms is defined inductively by the rules below.

$$\frac{}{x \Rightarrow_\beta x} \qquad \frac{t \Rightarrow_\beta t'}{\lambda x.t \Rightarrow_\beta \lambda x.t'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{st \Rightarrow_\beta s't'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{(\lambda x.s)t \Rightarrow_\beta s'\{t'/x\}}$$

Idea. $t \Rightarrow_\beta t'$ means t' is obtained from t by firing simultaneously a number of β-redexes.

Ex. $t = (\lambda x.xx)((\lambda z.z)y) \Rightarrow_\beta yy$. Note that $t \to_\beta (\lambda x.xx)y \to_\beta yy$ but $t \not\to_\beta yy$.

Lemma (properties of $\Rightarrow_\beta$)

1. (Reflexivity) $t \Rightarrow_\beta t$ for every λ-term t.
2. (Containing $\to_\beta$) If $t \to_\beta t'$ then $t \Rightarrow_\beta t'$.
3. (Contained in $\to_\beta^*$) If $t \Rightarrow_\beta t'$ then $t \to_\beta^* t'$.
4. (Closure under substitution) If $s \Rightarrow_\beta s'$ and $t \Rightarrow_\beta t'$ then $s\{t/x\} \Rightarrow_\beta s'\{t'/x\}$.
5. (Reflexive-transitive closure) $\to_\beta^* = \Rightarrow_\beta^*$.

Proof. 1) By induction on t. 2) By induction on the definition of $t \to_\beta t'$, using Point 1. 3)–4) By induction on t. 5) $\to_\beta \subseteq \Rightarrow$ by Point 2, hence $\to_\beta^* \subseteq \Rightarrow^*$; on the other hand, $\to_\beta^* \supseteq \Rightarrow$ by Point 3, hence $\to_\beta^* = \to_\beta^{**} \subseteq \Rightarrow^*$. □

A tool to prove confluence: parallel β-reduction

Def. The parallel β-reduction $\Rightarrow_\beta$ on λ-terms is defined inductively by the rules below.

$$\frac{}{x \Rightarrow_\beta x} \qquad \frac{t \Rightarrow_\beta t'}{\lambda x.t \Rightarrow_\beta \lambda x.t'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{st \Rightarrow_\beta s't'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{(\lambda x.s)t \Rightarrow_\beta s'\{t'/x\}}$$

Idea. $t \Rightarrow_\beta t'$ means t' is obtained from t by firing simultaneously a number of β-redexes.

Ex. $t = (\lambda x.xx)((\lambda z.z)y) \Rightarrow_\beta yy$. Note that $t \to_\beta (\lambda x.xx)y \to_\beta yy$ but $t \not\to_\beta yy$.

Lemma (properties of $\Rightarrow_\beta$)

1. (Reflexivity) $t \Rightarrow_\beta t$ for every λ-term t.
2. (Containing $\to_\beta$) If $t \to_\beta t'$ then $t \Rightarrow_\beta t'$.
3. (Contained in $\to_\beta^*$) If $t \Rightarrow_\beta t'$ then $t \to_\beta^* t'$.
4. (Closure under substitution) If $s \Rightarrow_\beta s'$ and $t \Rightarrow_\beta t'$ then $s\{t/x\} \Rightarrow_\beta s'\{t'/x\}$.
5. (Reflexive-transitive closure) $\to_\beta^* = \Rightarrow_\beta^*$.

Proof. 1) By induction on t. 2) By induction on the definition of $t \to_\beta t'$, using Point 1. 3)–4) By induction on t. 5) $\to_\beta \subseteq \Rightarrow$ by Point 2, hence $\to_\beta^* \subseteq \Rightarrow^*$; on the other hand, $\to_\beta^* \supseteq \Rightarrow$ by Point 3, hence $\to_\beta^* = \to_\beta^{**} \subseteq \Rightarrow^*$. □

A tool to prove confluence: parallel β-reduction

Def. The parallel β-reduction $\Rightarrow_\beta$ on λ-terms is defined inductively by the rules below.

$$\frac{}{x \Rightarrow_\beta x} \qquad \frac{t \Rightarrow_\beta t'}{\lambda x.t \Rightarrow_\beta \lambda x.t'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{st \Rightarrow_\beta s't'} \qquad \frac{s \Rightarrow_\beta s' \quad t \Rightarrow_\beta t'}{(\lambda x.s)t \Rightarrow_\beta s'\{t'/x\}}$$

Idea. $t \Rightarrow_\beta t'$ means t' is obtained from t by firing simultaneously a number of β-redexes.

Ex. $t = (\lambda x.xx)((\lambda z.z)y) \Rightarrow_\beta yy$. Note that $t \to_\beta (\lambda x.xx)y \to_\beta yy$ but $t \not\to_\beta yy$.

Lemma (properties of $\Rightarrow_\beta$)

1. (Reflexivity) $t \Rightarrow_\beta t$ for every λ-term t.
2. (Containing $\to_\beta$) If $t \to_\beta t'$ then $t \Rightarrow_\beta t'$.
3. (Contained in $\to_\beta^*$) If $t \Rightarrow_\beta t'$ then $t \to_\beta^* t'$.
4. (Closure under substitution) If $s \Rightarrow_\beta s'$ and $t \Rightarrow_\beta t'$ then $s\{t/x\} \Rightarrow_\beta s'\{t'/x\}$.
5. (Reflexive-transitive closure) $\to_\beta^* = \Rightarrow_\beta^*$.

Proof. 1) By induction on t. 2) By induction on the definition of $t \to_\beta t'$, using Point 1. 3)–4) By induction on t. 5) $\to_\beta \subseteq \Rightarrow$ by Point 2, hence $\to_\beta^* \subseteq \Rightarrow^*$; on the other hand, $\to_\beta^* \supseteq \Rightarrow$ by Point 3, hence $\to_\beta^* = \to_\beta^{**} \subseteq \Rightarrow^*$. □

Complete development via parallel β-reduction

Def. The complete development t^* of a λ-term t is defined by induction on t as follows:

$$x^* = x \qquad (st)^* = s^* t^* \quad \text{if } s \neq \lambda x.r$$
$$(\lambda x.t)^* = \lambda x.t^* \qquad ((\lambda x.s)t)^* = s^*\{t^*/x\}$$

Idea. t^* is obtained from t by firing *all* β-redexes in t simultaneously.

Lemma (complete development)

If $t \Rightarrow_\beta s$ then $s \Rightarrow_\beta t^*$.

Proof. By induction on t. Cases:

- If $t = x \Rightarrow_\beta s$ then $s = x \Rightarrow_\beta x = t^*$.
- If $t = \lambda x.t_1 \Rightarrow_\beta s$ then $s = \lambda x.s_1$ with $t_1 \Rightarrow_\beta s_1$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and hence $s = \lambda x.s_1 \Rightarrow_\beta \lambda x.t_1^* = t^*$.
- If $t = t_1 t_2 \Rightarrow_\beta s$ with $t_1 \neq \lambda x.r$, then $s = s_1 s_2$ with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$, hence $s = s_1 s_2 \Rightarrow_\beta t_1^* t_2^* = t^*$.
- If $t = (\lambda x.t_1)t_2$ then either $s = (\lambda x.s_1)s_2$ or $s = s_1\{s_2/x\}$, both with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$. There are two cases:
  1. if $s = (\lambda x.s_1)s_2$ then $s \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$;
  2. otherwise $s = s_1\{s_2/x\} \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$ by Point 4 in the lemma on p. 16. □

Complete development via parallel β-reduction

Def. The complete development t^* of a λ-term t is defined by induction on t as follows:

$$x^* = x \qquad (st)^* = s^* t^* \quad \text{if } s \neq \lambda x.r$$
$$(\lambda x.t)^* = \lambda x.t^* \qquad ((\lambda x.s)t)^* = s^*\{t^*/x\}$$

Idea. t^* is obtained from t by firing *all* β-redexes in t simultaneously.

Lemma (complete development)

If $t \Rightarrow_\beta s$ then $s \Rightarrow_\beta t^*$.

Proof. By induction on t. Cases:

- If $t = x \Rightarrow_\beta s$ then $s = x \Rightarrow_\beta x = t^*$.
- If $t = \lambda x.t_1 \Rightarrow_\beta s$ then $s = \lambda x.s_1$ with $t_1 \Rightarrow_\beta s_1$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and hence $s = \lambda x.s_1 \Rightarrow_\beta \lambda x.t_1^* = t^*$.
- If $t = t_1 t_2 \Rightarrow_\beta s$ with $t_1 \neq \lambda x.r$, then $s = s_1 s_2$ with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$, hence $s = s_1 s_2 \Rightarrow_\beta t_1^* t_2^* = t^*$.
- If $t = (\lambda x.t_1)t_2$ then either $s = (\lambda x.s_1)s_2$ or $s = s_1\{s_2/x\}$, both with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$. There are two cases:
  1. if $s = (\lambda x.s_1)s_2$ then $s \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$;
  2. otherwise $s = s_1\{s_2/x\} \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$ by Point 4 in the lemma on p. 16. □

Complete development via parallel β-reduction

Def. The complete development t^* of a λ-term t is defined by induction on t as follows:

$$x^* = x \qquad (st)^* = s^* t^* \quad \text{if } s \neq \lambda x.r$$
$$(\lambda x.t)^* = \lambda x.t^* \qquad ((\lambda x.s)t)^* = s^*\{t^*/x\}$$

Idea. t^* is obtained from t by firing *all* β-redexes in t simultaneously.

Lemma (complete development)

If $t \Rightarrow_\beta s$ then $s \Rightarrow_\beta t^*$.

Proof. By induction on t. Cases:

- If $t = x \Rightarrow_\beta s$ then $s = x \Rightarrow_\beta x = t^*$.
- If $t = \lambda x.t_1 \Rightarrow_\beta s$ then $s = \lambda x.s_1$ with $t_1 \Rightarrow_\beta s_1$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and hence $s = \lambda x.s_1 \Rightarrow_\beta \lambda x.t_1^* = t^*$.
- If $t = t_1 t_2 \Rightarrow_\beta s$ with $t_1 \neq \lambda x.r$, then $s = s_1 s_2$ with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$, hence $s = s_1 s_2 \Rightarrow_\beta t_1^* t_2^* = t^*$.
- If $t = (\lambda x.t_1)t_2$ then either $s = (\lambda x.s_1)s_2$ or $s = s_1\{s_2/x\}$, both with $t_1 \Rightarrow_\beta s_1$ and $t_2 \Rightarrow_\beta s_2$. By induction hypothesis, $s_1 \Rightarrow_\beta t_1^*$ and $s_2 \Rightarrow_\beta t_2^*$. There are two cases:
  1. if $s = (\lambda x.s_1)s_2$ then $s \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$;
  2. otherwise $s = s_1\{s_2/x\} \Rightarrow_\beta t_1^*\{t_2^*/x\} = t^*$ by Point 4 in the lemma on p. 16. □

Proof of confluence of $\to_\beta$ via parallel β-reduction

Corollary

1. ($\Rightarrow_\beta$ is diamond) If $t \Rightarrow_\beta s_1$ and $t \Rightarrow_\beta s_2$ then $s_1 \Rightarrow_\beta r$ and $s_2 \Rightarrow_\beta r$ for some r.
2. ($\Rightarrow_\beta$ is confluent) If $t \Rightarrow_\beta^* s_1$ and $t \Rightarrow_\beta^* s_2$ then $s_1 \Rightarrow_\beta^* r$ and $s_2 \Rightarrow_\beta^* r$ for some r.

Proof. 1) By the complete development lemma, $s_1 \;{}_\beta\!\Leftarrow t \Rightarrow_\beta s_2$ implies $s_1 \Rightarrow_\beta t^* \;{}_\beta\!\Leftarrow s_2$.
2) It is a good exercise! □

Theorem (confluence of $\to_\beta$)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$, then there is a term r such that $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$.

Proof. By Point 5 in the lemma on p. 16, the hypothesis $s_1 \;{}^*_\beta\!\leftarrow t \to_\beta^* s_2$ means $s_1 \;{}^*_\beta\!\Leftarrow t \Rightarrow_\beta^* s_2$. By confluence of $\Rightarrow_\beta$, there is r such that $s_1 \Rightarrow_\beta^* r \;{}^*_\beta\!\Leftarrow s_2$, which means $s_1 \to_\beta^* r \;{}^*_\beta\!\leftarrow s_2$ by Point 5 again. □

Proof of confluence of $\to_\beta$ via parallel β-reduction

Corollary

1. ($\Rightarrow_\beta$ is diamond) If $t \Rightarrow_\beta s_1$ and $t \Rightarrow_\beta s_2$ then $s_1 \Rightarrow_\beta r$ and $s_2 \Rightarrow_\beta r$ for some r.
2. ($\Rightarrow_\beta$ is confluent) If $t \Rightarrow_\beta^* s_1$ and $t \Rightarrow_\beta^* s_2$ then $s_1 \Rightarrow_\beta^* r$ and $s_2 \Rightarrow_\beta^* r$ for some r.

Proof. 1) By the complete development lemma, $s_1 \;{}_\beta\!\Leftarrow t \Rightarrow_\beta s_2$ implies $s_1 \Rightarrow_\beta t^* \;{}_\beta\!\Leftarrow s_2$.
2) It is a good exercise! □

Theorem (confluence of $\to_\beta$)

If $t \to_\beta^* s_1$ and $t \to_\beta^* s_2$, then there is a term r such that $s_1 \to_\beta^* r$ and $s_2 \to_\beta^* r$.

Proof. By Point 5 in the lemma on p. 16, the hypothesis $s_1 \;{}^*_\beta\!\leftarrow t \to_\beta^* s_2$ means $s_1 \;{}^*_\beta\!\Leftarrow t \Rightarrow_\beta^* s_2$. By confluence of $\Rightarrow_\beta$, there is r such that $s_1 \Rightarrow_\beta^* r \;{}^*_\beta\!\Leftarrow s_2$, which means $s_1 \to_\beta^* r \;{}^*_\beta\!\leftarrow s_2$ by Point 5 again. □

Outline

1. The syntax and the operational semantics of the untyped λ-calculus
2. The proof of confluence of β-reduction
3. The notions of β-equivalence and $\beta\eta$-equivalence
4. Conclusion, exercises and bibliography

λ-theories and the β-equivalence

Especially in semantic studies, equational theories (or λ-theories) on λ-terms are studied.
$\rightsquigarrow$ Focus on equivalences between λ-terms, their extensional properties.

Idea. Terms belonging to the same equivalence class have the same "meaning".

Def. A λ-theory is a binary relation $\equiv$ on λ-terms closed under the clauses below.

$$\frac{}{(\lambda x.s)t \equiv s\{t/x\}}\,\beta \qquad \overbrace{\underbrace{\frac{}{t \equiv t}\,\text{refl} \quad \frac{s \equiv t}{t \equiv s}\,\text{sym} \quad \frac{s \equiv t \quad t \equiv u}{s \equiv u}\,\text{trans}}_{\text{equivalence}} \quad \frac{s \equiv t}{\lambda x.s \equiv \lambda x.t}\,\lambda \quad \frac{s \equiv s' \quad t \equiv t'}{st \equiv s't'}\,@}^{\text{congruence}}$$

Def. β-equivalence $=_\beta$ is the smallest λ-theory.

Proposition

β-equivalence coincides with the reflexive-transitive, symmetric closure of β-reduction, i.e. $=_\beta = (\to_\beta \cup \beta\leftarrow)^*$, i.e. $t =_\beta s$ iff $\exists\,(r_i)_{0\le i\le n} : t = r_0,\ s = r_n,\ r_i \to_\beta \cup\, \beta\leftarrow r_{i+1}\ \forall\, 1 \le i < n$.

Proof. It is a good exercise! □

λ-theories and the β-equivalence

Especially in semantic studies, equational theories (or λ-theories) on λ-terms are studied.
$\rightsquigarrow$ Focus on equivalences between λ-terms, their extensional properties.

Idea. Terms belonging to the same equivalence class have the same "meaning".

Def. A λ-theory is a binary relation $\equiv$ on λ-terms closed under the clauses below.

$$\frac{}{(\lambda x.s)t \equiv s\{t/x\}}\,\beta \qquad \overbrace{\underbrace{\frac{}{t \equiv t}\,\text{refl} \quad \frac{s \equiv t}{t \equiv s}\,\text{sym} \quad \frac{s \equiv t \quad t \equiv u}{s \equiv u}\,\text{trans}}_{\text{equivalence}} \quad \frac{s \equiv t}{\lambda x.s \equiv \lambda x.t}\,\lambda \quad \frac{s \equiv s' \quad t \equiv t'}{st \equiv s't'}\,@}^{\text{congruence}}$$

Def. β-equivalence $=_\beta$ is the smallest λ-theory.

Proposition

β-equivalence coincides with the reflexive-transitive, symmetric closure of β-reduction, i.e. $=_\beta = (\to_\beta \cup \beta\leftarrow)^*$, i.e. $t =_\beta s$ iff $\exists\,(r_i)_{0\le i\le n} : t = r_0,\ s = r_n,\ r_i \to_\beta \cup\, \beta\leftarrow r_{i+1}\ \forall\, 1 \le i < n$.

Proof. It is a good exercise! □

λ-theories and the β-equivalence

Especially in semantic studies, equational theories (or λ-theories) on λ-terms are studied.
$\rightsquigarrow$ Focus on equivalences between λ-terms, their extensional properties.

Idea. Terms belonging to the same equivalence class have the same "meaning".

Def. A λ-theory is a binary relation $\equiv$ on λ-terms closed under the clauses below.

$$\frac{}{(\lambda x.s)t \equiv s\{t/x\}}\,\beta \qquad \overbrace{\underbrace{\frac{}{t \equiv t}\,\text{refl} \quad \frac{s \equiv t}{t \equiv s}\,\text{sym} \quad \frac{s \equiv t \quad t \equiv u}{s \equiv u}\,\text{trans}}_{\text{equivalence}} \quad \frac{s \equiv t}{\lambda x.s \equiv \lambda x.t}\,\lambda \quad \frac{s \equiv s' \quad t \equiv t'}{st \equiv s't'}\,@}^{\text{congruence}}$$

Def. β-equivalence $=_\beta$ is the smallest λ-theory.

Proposition

β-equivalence coincides with the reflexive-transitive, symmetric closure of β-reduction, i.e. $=_\beta = (\to_\beta \cup \beta\leftarrow)^*$, i.e. $t =_\beta s$ iff $\exists\,(r_i)_{0\le i\le n} : t = r_0,\ s = r_n,\ r_i \to_\beta \cup\, \beta\leftarrow r_{i+1}\ \forall\, 1 \le i < n$.

Proof. It is a good exercise! □

Some properties of β-equivalence

As it is confluent, β-reduction can be used as a tool to investigate β-equivalence.

Theorem (Church–Rosser)

1. If $s =_\beta t$ then $s \to_\beta^* u$ and $t \to_\beta^* u$ for some u.
2. If $s =_\beta t$ and t is β-normal, then $s \to_\beta^* t$.

Proof. 1) Since $s =_\beta t$, there is $(r_i)_{0 \leq i \leq n}$ for some $n \in \mathbb{N}$ such that $r_0 = s$ and $r_n = t$, and $r_i \to_\beta r_{i+1}$ or $r_{i+1} \to_\beta r_i$ for all $0 \leq i < n$. We proceed by induction on $n \in \mathbb{N}$. Cases:

- $n = 0$: Then, $s = t$ and the statement to prove holds by taking $u = t$.
- $n > 0$: Thus, $s =_\beta r_{n-1}$ and ($r_{n-1} \to_\beta t$ or $t \to_\beta r_{n-1}$). By induction hypothesis applied to $s =_\beta r_{n-1}$ (as $n - 1 < n$), $s \to_\beta^* u' \,{}_\beta^*\!\leftarrow r_{n-1}$ for some u'. Two cases:
  1. if $t = r_n \to_\beta r_{n-1}$, then $t \to_\beta^* u'$ and the statement to prove holds by taking $u = u'$;
  2. if $r_{n-1} \to_\beta r_n = t$, then $u' \,{}_\beta^*\!\leftarrow r_{n-1} \to_\beta t$ and hence, by confluence of $\to_\beta$, there is u such that $u' \to_\beta^* u \,{}_\beta^*\!\leftarrow t$. As $s \to_\beta^* u'$, then $s \to_\beta^* u$ and we are done.

2) By Point 1, $s \to_\beta^* u \,{}_\beta^*\!\leftarrow t$ for some u. As t is β-normal, $t = u$ and so $s \to_\beta^* t$. □

Theorem (consistency)

The β-equivalence is consistent, that is, there are some λ-terms s, t such that $s \neq_\beta t$.

Proof. Let $s = \lambda x.x$ and $t = \lambda y.\lambda x.x$. If $s =_\beta t$ then $s \to_\beta^* t$ by Church–Rosser, since t is β-normal. As s is β-normal, then $s = t$, but $s \neq t$. Therefore, $\lambda x.x \neq_\beta \lambda y.\lambda x.x$. □

Some properties of β-equivalence

As it is confluent, β-reduction can be used as a tool to investigate β-equivalence.

Theorem (Church–Rosser)

1. If $s =_\beta t$ then $s \to_\beta^* u$ and $t \to_\beta^* u$ for some u.
2. If $s =_\beta t$ and t is β-normal, then $s \to_\beta^* t$.

Proof. 1) Since $s =_\beta t$, there is $(r_i)_{0 \leq i \leq n}$ for some $n \in \mathbb{N}$ such that $r_0 = s$ and $r_n = t$, and $r_i \to_\beta r_{i+1}$ or $r_{i+1} \to_\beta r_i$ for all $0 \leq i < n$. We proceed by induction on $n \in \mathbb{N}$. Cases:

- $n = 0$: Then, $s = t$ and the statement to prove holds by taking $u = t$.
- $n > 0$: Thus, $s =_\beta r_{n-1}$ and ($r_{n-1} \to_\beta t$ or $t \to_\beta r_{n-1}$). By induction hypothesis applied to $s =_\beta r_{n-1}$ (as $n - 1 < n$), $s \to_\beta^* u' {}_\beta^*\!\leftarrow r_{n-1}$ for some u'. Two cases:
  1. if $t = r_n \to_\beta r_{n-1}$, then $t \to_\beta^* u'$ and the statement to prove holds by taking $u = u'$;
  2. if $r_{n-1} \to_\beta r_n = t$, then $u' {}_\beta^*\!\leftarrow r_{n-1} \to_\beta t$ and hence, by confluence of $\to_\beta$, there is u such that $u' \to_\beta^* u {}_\beta^*\!\leftarrow t$. As $s \to_\beta^* u'$, then $s \to_\beta^* u$ and we are done.

2) By Point 1, $s \to_\beta^* u {}_\beta^*\!\leftarrow t$ for some u. As t is β-normal, $t = u$ and so $s \to_\beta^* t$. □

Theorem (consistency)

The β-equivalence is consistent, that is, there are some λ-terms s, t such that $s \neq_\beta t$.

Proof. Let $s = \lambda x.x$ and $t = \lambda y.\lambda x.x$. If $s =_\beta t$ then $s \to_\beta^* t$ by Church–Rosser, since t is β-normal. As s is β-normal, then $s = t$, but $s \neq t$. Therefore, $\lambda x.x \neq_\beta \lambda y.\lambda x.x$. □

Some properties of β-equivalence

As it is confluent, β-reduction can be used as a tool to investigate β-equivalence.

Theorem (Church–Rosser)

1. If $s =_\beta t$ then $s \to_\beta^* u$ and $t \to_\beta^* u$ for some u.
2. If $s =_\beta t$ and t is β-normal, then $s \to_\beta^* t$.

Proof. 1) Since $s =_\beta t$, there is $(r_i)_{0 \le i \le n}$ for some $n \in \mathbb{N}$ such that $r_0 = s$ and $r_n = t$, and $r_i \to_\beta r_{i+1}$ or $r_{i+1} \to_\beta r_i$ for all $0 \le i < n$. We proceed by induction on $n \in \mathbb{N}$. Cases:

- $n = 0$: Then, $s = t$ and the statement to prove holds by taking $u = t$.
- $n > 0$: Thus, $s =_\beta r_{n-1}$ and ($r_{n-1} \to_\beta t$ or $t \to_\beta r_{n-1}$). By induction hypothesis applied to $s =_\beta r_{n-1}$ (as $n - 1 < n$), $s \to_\beta^* u' {}_\beta^*\!\leftarrow r_{n-1}$ for some u'. Two cases:
  1. if $t = r_n \to_\beta r_{n-1}$, then $t \to_\beta^* u'$ and the statement to prove holds by taking $u = u'$;
  2. if $r_{n-1} \to_\beta r_n = t$, then $u' {}_\beta^*\!\leftarrow r_{n-1} \to_\beta t$ and hence, by confluence of $\to_\beta$, there is u such that $u' \to_\beta^* u {}_\beta^*\!\leftarrow t$. As $s \to_\beta^* u'$, then $s \to_\beta^* u$ and we are done.

2) By Point 1, $s \to_\beta^* u {}_\beta^*\!\leftarrow t$ for some u. As t is β-normal, $t = u$ and so $s \to_\beta^* t$. □

Theorem (consistency)

The β-equivalence is consistent, that is, there are some λ-terms s, t such that $s \neq_\beta t$.

Proof. Let $s = \lambda x.x$ and $t = \lambda y.\lambda x.x$. If $s =_\beta t$ then $s \to_\beta^* t$ by Church–Rosser, since t is β-normal. As s is β-normal, then $s = t$, but $s \neq t$. Therefore, $\lambda x.x \neq_\beta \lambda y.\lambda x.x$. □

$\beta\eta$-equivalence and extensionality

In mathematics, the principle of functional extensionality states that:

given two functions $f, g \colon X \Rightarrow Y$, if $f(x) = g(x)$ for all $x \in X$ then $f = g$.

Question. As the λ-calculus is a theory of functions, does extensionality hold there?

If we take $=_\beta$ as the equality, the answer is negative. Take $s = \lambda x.yx$ and $t = y$:

- $su =_\beta tu$ for every term u, indeed $su = (\lambda x.yx)u \to_\beta yu = tu$;
- but $s \neq_\beta t$, for if $s =_\beta t$ then by Church–Rosser $s = \lambda x.yx \to_\beta^* y = t$, impossible.

Def. $\beta\eta$-equivalence $=_{\beta\eta}$ is the least λ-theory with the axiom $\dfrac{}{\lambda x.tx =_{\beta\eta} t}\,\eta$ if $x \notin \mathrm{fv}(t)$.

Theorem (Extensionality)

If $su =_{\beta\eta} tu$ for every λ-term u, then $s =_{\beta\eta} t$.

Proof. By hypothesis, $sx =_{\beta\eta} tx$ with $x \notin \mathrm{fv}(s) \cup \mathrm{fv}(t)$. As $=_{\beta\eta}$ is a congruence, $\lambda x.sx =_{\beta\eta} \lambda x.tx$. By the η-axiom, $\lambda x.sx =_{\beta\eta} s$ and $\lambda x.tx =_{\beta\eta} t$. By transitivity, $s =_{\beta\eta} t$. □

η-reduction $\to_\eta$ is generated by the rule $\lambda x.tx \to_\eta t$ if $x \notin \mathrm{fv}(t)$, fired anywhere in a term.

Rmk. $=_{\beta\eta} = (\to_{\beta\eta} \cup {}_{\beta\eta}\!\leftarrow)^*$ where $\to_{\beta\eta} = \to_\beta \cup \to_\eta$ (exercise!).

$\beta\eta$-equivalence and extensionality

In mathematics, the principle of functional extensionality states that:

$$\text{given two functions } f, g \colon X \Rightarrow Y, \text{ if } f(x) = g(x) \text{ for all } x \in X \text{ then } f = g.$$

Question. As the λ-calculus is a theory of functions, does extensionality hold there?

If we take $=_\beta$ as the equality, the answer is negative. Take $s = \lambda x.yx$ and $t = y$:

- $su =_\beta tu$ for every term u, indeed $su = (\lambda x.yx)u \to_\beta yu = tu$;
- but $s \neq_\beta t$, for if $s =_\beta t$ then by Church–Rosser $s = \lambda x.yx \to_\beta^* y = t$, impossible.

Def. $\beta\eta$-equivalence $=_{\beta\eta}$ is the least λ-theory with the axiom $\dfrac{}{\lambda x.tx =_{\beta\eta} t}\,\eta$ if $x \notin \mathrm{fv}(t)$.

Theorem (Extensionality)

If $su =_{\beta\eta} tu$ for every λ-term u, then $s =_{\beta\eta} t$.

Proof. By hypothesis, $sx =_{\beta\eta} tx$ with $x \notin \mathrm{fv}(s) \cup \mathrm{fv}(t)$. As $=_{\beta\eta}$ is a congruence, $\lambda x.sx =_{\beta\eta} \lambda x.tx$. By the η-axiom, $\lambda x.sx =_{\beta\eta} s$ and $\lambda x.tx =_{\beta\eta} t$. By transitivity, $s =_{\beta\eta} t$. □

η-reduction $\to_\eta$ is generated by the rule $\lambda x.tx \to_\eta t$ if $x \notin \mathrm{fv}(t)$, fired anywhere in a term.

Rmk. $=_{\beta\eta} = (\to_{\beta\eta} \cup {}_{\beta\eta}\!\leftarrow)^*$ where $\to_{\beta\eta} = \to_\beta \cup \to_\eta$ (exercise!).

$\beta\eta$-equivalence and extensionality

In mathematics, the principle of functional extensionality states that:

$$\text{given two functions } f, g\colon X \Rightarrow Y, \text{ if } f(x) = g(x) \text{ for all } x \in X \text{ then } f = g.$$

Question. As the λ-calculus is a theory of functions, does extensionality hold there?

If we take $=_\beta$ as the equality, the answer is negative. Take $s = \lambda x.yx$ and $t = y$:

- $su =_\beta tu$ for every term u, indeed $su = (\lambda x.yx)u \to_\beta yu = tu$;
- but $s \neq_\beta t$, for if $s =_\beta t$ then by Church–Rosser $s = \lambda x.yx \to_\beta^* y = t$, impossible.

Def. $\beta\eta$-equivalence $=_{\beta\eta}$ is the least λ-theory with the axiom $\dfrac{}{\lambda x.tx =_{\beta\eta} t}\,\eta$ if $x \notin \mathrm{fv}(t)$.

Theorem (Extensionality)

If $su =_{\beta\eta} tu$ for every λ-term u, then $s =_{\beta\eta} t$.

Proof. By hypothesis, $sx =_{\beta\eta} tx$ with $x \notin \mathrm{fv}(s) \cup \mathrm{fv}(t)$. As $=_{\beta\eta}$ is a congruence, $\lambda x.sx =_{\beta\eta} \lambda x.tx$. By the η-axiom, $\lambda x.sx =_{\beta\eta} s$ and $\lambda x.tx =_{\beta\eta} t$. By transitivity, $s =_{\beta\eta} t$. □

η-reduction $\to_\eta$ is generated by the rule $\lambda x.tx \to_\eta t$ if $x \notin \mathrm{fv}(t)$, fired anywhere in a term.

Rmk. $=_{\beta\eta} = (\to_{\beta\eta} \cup {}_{\beta\eta}\!\leftarrow)^*$ where $\to_{\beta\eta} = \to_\beta \cup \to_\eta$ (exercise!).

$\beta\eta$-equivalence and extensionality

In mathematics, the principle of functional extensionality states that:

$$\text{given two functions } f, g\colon X \Rightarrow Y, \text{ if } f(x) = g(x) \text{ for all } x \in X \text{ then } f = g.$$

Question. As the λ-calculus is a theory of functions, does extensionality hold there?

If we take $=_\beta$ as the equality, the answer is negative. Take $s = \lambda x.yx$ and $t = y$:

- $su =_\beta tu$ for every term u, indeed $su = (\lambda x.yx)u \to_\beta yu = tu$;
- but $s \neq_\beta t$, for if $s =_\beta t$ then by Church–Rosser $s = \lambda x.yx \to_\beta^* y = t$, impossible.

Def. $\beta\eta$-equivalence $=_{\beta\eta}$ is the least λ-theory with the axiom $\dfrac{}{\lambda x.tx =_{\beta\eta} t}\,\eta$ if $x \notin \mathrm{fv}(t)$.

Theorem (Extensionality)

If $su =_{\beta\eta} tu$ for every λ-term u, then $s =_{\beta\eta} t$.

Proof. By hypothesis, $sx =_{\beta\eta} tx$ with $x \notin \mathrm{fv}(s) \cup \mathrm{fv}(t)$. As $=_{\beta\eta}$ is a congruence, $\lambda x.sx =_{\beta\eta} \lambda x.tx$. By the η-axiom, $\lambda x.sx =_{\beta\eta} s$ and $\lambda x.tx =_{\beta\eta} t$. By transitivity, $s =_{\beta\eta} t$. □

η-reduction $\to_\eta$ is generated by the rule $\lambda x.tx \to_\eta t$ if $x \notin \mathrm{fv}(t)$, fired anywhere in a term.

Rmk. $=_{\beta\eta} = (\to_{\beta\eta} \cup {}_{\beta\eta}\!\leftarrow)^*$ where $\to_{\beta\eta} = \to_\beta \cup \to_\eta$ (exercise!).

Outline

1. The syntax and the operational semantics of the untyped λ-calculus
2. The proof of confluence of β-reduction
3. The notions of β-equivalence and $\beta\eta$-equivalence
4. Conclusion, exercises and bibliography

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?

What we have learned today?

1. Syntax and operational semantics of the untyped λ-calculus.
2. Different notions of β-reduction (full, leftmost, head).
3. Different notions of normalization (strong or not).
4. The proof of confluence of β-reduction via parallel β-reduction.
5. The notions of β-equivalence and $\beta\eta$-equivalence in the untyped λ-calculus.

Questions?



Exercises

1. Find terms t, t', s, s' such that $t =_\alpha t'$, $s =_\alpha s'$ and $t[s/x] \neq_\alpha t'[s'/x]$ (where $=_\alpha$ is α-equivalence and $t[s/x]$ is naïve substitution, see p. 22 on Lecture 2 slides).
2. Write the tree representation of following terms (as on p. 7), specifying $m, n \in \mathbb{N}$ and the subtrees corresponding to $h, t_1, \ldots, t_m$: x, I, $\lambda x.Ixx$, $\lambda x.I(xx)$, $\lambda x.xxx(xx)$, II.
3. The β-reduction graph of a term t is the directed graph with nodes $\{s \mid t \to_\beta^* s\}$ and with edges the single β-steps. Draw the β-reduction graph of the following terms:
   1. $(\lambda x.Ixx)(\lambda x.Ixx)$ where $I = \lambda z.z$.
   2. $(\lambda x.I(xx))(\lambda x.I(xx))$.
   3. $(II)(III)$.
   4. $\delta\delta$ where $\delta = \lambda x.xx$.
   5. $\delta_3\delta_3$ where $\delta_3 = \lambda x.xxx$.
   6. $\pi\pi\pi$ where $\pi = \lambda x.\lambda y.xyy$.
4. Find a term r such that $rt \to_\beta^* t(tr)$ for every t (*hint:* use fixpoint combinator Θ).
5. Find a term r such that $rst \to_\beta^* rtsr$ for every s, t (*hint:* use fixpoint combinator Θ).
6. Prove rigorously the remark and proposition on p. 13.
7. Let R be a binary relation on a set X, and let R^* be its reflexive-transitive closure, i.e. $x\, R^*\, y$ iff there are $n \in \mathbb{N}$ and $(z_i)_{0 \leq i \leq n}$ such that $x = z_0$, $y = z_n$ and $z_i\, R\, z_{i+1}$ for all $0 \leq i < n$. Prove that $R^{**} = R^*$.
8. Prove that $(\lambda x.xx)((\lambda z.z)y) \Rightarrow_\beta yy$ using the rules on p. 16 defining $\Rightarrow_\beta$.
9. Prove rigorously Points 1–3 in the lemma on p. 16.
10. Prove that $\to_\eta$ is strongly normalizing in the untyped λ-calculus.
11. Prove rigorously the proposition on p. 20 and the remark on p. 22.

Bibliography

- For more about the untyped λ-calculus:


Jean-Louis Krivine. *Lambda-Calculus. Types and Models*. Ellis Horwood. 1990. [Chapters 1-2] https://www.irif.fr/~krivine/articles/Lambda.pdf


Peter Selinger. *Lecture Notes on the Lambda Calculus*. vol. 0804, Department of Mathematics and Statistics, University of Ottawa. 2008 [Chapters 2-3] http://www.mathstat.dal.ca/~selinger/papers/lambdanotes.pdf


Henk P. Barendregt. *The Lambda-Calculus. Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics, vol. 103, North Holland, 1984. [Chapters 2-3, 6, 8]

- For an elegant proof of the confluence of β-reduction:


Masako Takahashi. *Parallel Reductions in λ-Calculus*. Information and Computation, vol. 118, issue 1, pages 120-127. 1995. https://doi.org/10.1006/inco.1995.1057