FLOLAC 2020

Programming Language Theory

Untyped λ-Calculus

陳亮廷 Chen, Liang-Ting

2020 邏輯、語言與計算暑期研習營

Formosan Summer School on Logic, Language, and Computation

Institute of Information Science, Academia Sinica

Assessment guidelines

Deadline 17:00, 20 Aug

Assessment **Assignment** (10% + 10% + 10%)

Exam (100%)

Email `liang.ting.chen.tw(at)gmail(dot)com`

Please follow the instructions for assignments below.

1. Write down your name and your student number.
2. Use A4 paper only in physical form or in *non-scanned* PDF.
3. Be brief but comprehensive.
4. Submit assignments in person or by email with

   subject `[FLOLAC] PL HW%x%`

   attachment `PL-HW%x% - %STDNO% - %NAME%.pdf`

   body (optional)

Untyped λ-Calculus: Statics

λ-calculus: Terms

Let $V := \{x, y, z, \ldots\}$ be a countably infinite set of *variables*.

Definition 1 (Syntax of λ-calculus)

The formation $M : \mathtt{Term}_\lambda$ is defined by

1. Variable:

$$\frac{x \in V}{x : \mathtt{Term}_\lambda}\,(\text{var})$$

2. Function application of M to the argument N:

$$\frac{M : \mathtt{Term}_\lambda \qquad N : \mathtt{Term}_\lambda}{M\,N : \mathtt{Term}_\lambda}\,(\text{app})$$

3. Function abstraction with an argument x and a function body M:

$$\frac{M : \mathtt{Term}_\lambda \qquad x \in V}{\lambda x.\, M : \mathtt{Term}_\lambda}\,(\text{abs})$$

Examples and non-examples

1. $(x\ y)\ z$
2. $x\ (y\ z)$
3. $\lambda x.\, y$
4. $\lambda x.\, x$
5. $\lambda s.\, (\lambda z.\, (s\ z))$
6. $\lambda a.\, (\lambda b.\, (a\ (\lambda c.\, a\ b)))$
7. $(\lambda x.\, x)\ (\lambda y.\, y)$

The following are NOT examples

1. $\lambda(\lambda x.\, x).\, y$
2. $\lambda x.$
3. $\lambda.\, x$
4. $\ldots$

Closed terms

A term M is closed if each variable which occurs in M are captured by an abstraction.

Put differently, M is closed if every variable is in scope.

Closed terms:

1. $\lambda x.\, x$
2. $\lambda s.\, (\lambda z.\, (s\ z))$
3. $\lambda a.\, (\lambda b.\, (a\ (\lambda c.\, a\ b)))$
4. $(\lambda x.\, x)\ (\lambda y.\, y)$

Open terms:

1. $(x\ y)\ z$
2. $x\ (y\ z)$
3. $\lambda x.\, y$

Conventions

Consecutive abstractions

$$\lambda x_1\, x_2\, \ldots x_n.\, M := \lambda x_1.\, (\lambda x_2.\, (\ldots (\lambda x_n.\, M) \ldots))$$

Consecutive applications

$$M_1\, M_2\, M_3\, \ldots\, M_n := (\ldots ((M_1\, M_2)\, M_3) \ldots)\, M_n$$

Function body extends as far right as possible

$$\lambda x.\, M\, N := \lambda x.\, (M\, N)$$

instead of $(\lambda x.\, M)\, N$.

For example, $\lambda x_1.\, (\lambda x_2.\, x_1) \equiv \lambda x_1\, x_2.\, x_1$ and $x\ y\ z$ means $(x\ y)\ z$.
Warning. We apply these rules in our *meta*-language.

Meta-language and object-language

- *Meta-language* is the language we use to describe the object of study. E.g. English, or naive set theory.
- *Object-language* is the object of study. E.g., arithmetic expressions and λ-terms.

Naming a function is not supported in λ-calculus, so the following

$$\texttt{id} := \lambda x.\, x$$

happens in the meta-language.

1. `id` is a symbol different from '$\lambda x.\, x$' in the meta-language.
2. `id` and $\lambda x.\, x$ are *syntactically equivalent* denoted by

   $$\texttt{id} \equiv \lambda x.\, x$$

   in the object language.

Example 2 (Identity function)

$$\mathtt{id} := \lambda x.\, x$$

Example 3 (Projections)

$$\mathtt{fst} := \lambda x.\, \lambda y.\, x \quad \text{and} \quad \mathtt{snd} := \lambda x.\, \lambda y.\, y$$

Example 4 (Church encoding of Natural numbers)

$$\begin{aligned}
\mathtt{c}_0 &:= \lambda f x.\, x \\
\mathtt{c}_1 &:= \lambda f x.\, f\, x \\
\mathtt{c}_2 &:= \lambda f x.\, f\,(f\, x) \\
\mathtt{c}_3 &:= \lambda f x.\, f\,(f\,(f(\, x))) \\
&\vdots
\end{aligned}$$

α-equivalence, informally

Definition 5

Two terms M an N are α-equivalent

$$M =_\alpha N$$

if variables *bound* by abstractions can be renamed to derive the same term.

Example 6

1. $\lambda x.\, x$ and $\lambda y.\, y$ are distinct λ-terms but $\lambda x.\, x =_\alpha \lambda y.\, y$.
2. $\lambda x.\, \lambda y.\, y =_\alpha \lambda z.\, \lambda y.\, y$.
3. $\lambda x.\, \lambda y.\, x \neq_\alpha \lambda x.\, \lambda y.\, y$.

α-equivalent terms are *programs of the same structure*.

Evaluation, informally

The evaluation of λ-calculus happens in this form

$$\underbrace{(\lambda x.\, M)\, N}_{\beta\text{-redex}} \longrightarrow \underbrace{M\,[N/x]}_{\text{substitution of } N \text{ for } x \text{ in } M}$$

For example, $(\lambda x.\, x + 1)\ 3 \to 3 + 1$.

How to evaluate the following terms?

1. $(\lambda x.x)\, z$
2. $(\lambda x\, y.\, x)\, y$
3. $(\lambda y\, y.\, y)\, x$

Equality, equality, equality!

Moreover, we will have different notions of equality.

- $1 + 1 \neq_\alpha 2$
- $1 + 1 \longrightarrow_{\beta *} 2$ but $2 \not\longrightarrow_{\beta *} 1 + 1$
- $1 + 1 =_\beta 2$

Each of above statements says the following.

- $1 + 1$ is a different expression from 2.
- $1 + 1$ reduces to 2, but 2 does not reduce to $1 + 1$.
- $1 + 1$ and 2 have the same computational meaning.

Free and bound variables

Definition 7

The set $\mathbf{FV}$ of free variables of a term M is defined by

$$\begin{aligned} \mathbf{FV}(x) &= \{x\} \\ \mathbf{FV}(\lambda x.\, M) &= \mathbf{FV}(M) - \{x\} \\ \mathbf{FV}(M\ N) &= \mathbf{FV}(M) \cup \mathbf{FV}(N) \end{aligned}$$

$\mathbf{FV}(x\ y\ z) = \mathbf{FV}(x\ y) \cup \mathbf{FV}(z) = \mathbf{FV}(x) \cup \mathbf{FV}(y) \cup \{z\} = \{x, y, z\}$

Definition 8

1. A variable y in M is free if $y \in \mathbf{FV}(M)$.
2. A λ-term M is *closed* if $\mathbf{FV}(M) = \emptyset$.

Exercise: free variables

Calculate the set of free variables of following terms:

1. $x\ (y\ z)$
2. $\lambda x.\, y$
3. $\lambda x.\, x$
4. $\lambda s\, z.\, (s\ z))$
5. $(\lambda x.\, x)\ \lambda y.\, y$

Exercise: bound variables

A variable x in a term M is called *bound* if it is not free.

Define

- $\mathbf{Var}(M)$ the set of variables of a term M by structural recursion on Λ.
- $\mathbf{BV}(M)$ the set of bound variables.

Untyped λ-Calculus: Substitution

Substitution

A substitution is a process of replacing *free* variables by another terms (on the meta-level).

The name of a variable does not matter but the location does. So, ...

1. bound variables should remain bound after substitution.
2. other free variables should remain free after substitution.

Concretely, we want to avoid ...

1. $(\lambda y.\, y)[x/y] \equiv (\lambda y.\, x)$
2. $(\lambda y.\, x)[y/x] \equiv (\lambda y.\, y)$

Naive substitution I

Definition 9

For $x \in V$ and $L : \mathtt{Term}_\lambda$, the substitution of L for x is defined by

$$
\begin{aligned}
x[L/x] &= L \\
y[L/x] &= y && \text{if } x \neq y \\
(M\ N)[L/x] &= M[L/x]\ N[L/x] \\
(\lambda y.\, M)[L/x] &= \lambda y.\, M[L/x]
\end{aligned}
$$

A bound variable may become free.

$$(\lambda x.\, x)[y/x] = \lambda x.\, y$$

Naive substitution II

Definition 10

For $x \in V$ and $L : \mathtt{Term}_\lambda$, the substitution of L for x is defined by

$$
\begin{aligned}
x[L/x] &= L && \\
y[L/x] &= y && \text{if } x \neq y \\
(M\,N)[L/x] &= M[L/x]\ N[L/x] && \\
(\lambda y.\, M)[L/x] &= \lambda y.\, M[L/x] && \text{if } x \neq y \\
(\lambda y.\, M)[L/x] &= \lambda y.\, M && \text{if } x = y
\end{aligned}
$$

A variable may be captured by an abstraction.

$$(\lambda x.y)[x/y] = \lambda x.\, x$$

Capture-avoiding substitution

Capture-avoiding substitution of L for the free occurrences of x is a *partial* function $(\cdot)[L/x] : \mathtt{Term}_\lambda \to \mathtt{Term}_\lambda$ defined by

$$
\begin{aligned}
x[L/x] &= L \\
y[L/x] &= y && \text{if } x \neq y \\
(M\,N)[L/x] &= (M[L/x]\ N[L/x]) \\
(\lambda x.\, M)[L/x] &= \lambda x.\, M \\
(\lambda y.\, M)[L/x] &= \lambda y.\, M[L/x] && \text{if } x \neq y \text{ and } y \notin \mathbf{FV}(L)
\end{aligned}
$$

Definition 11 (Freshness)

A variable y is fresh for L if $y \notin \mathbf{FV}(L)$.

Renaming of bound variables

If a variable y is *fresh* for M, the bound variable x of $\lambda x.\, M$ can be renamed to y without changing the meaning.

Definition 12 (α-conversion)

α-conversion is a relation $\to_\alpha$ defined by

$$(\lambda x.\, M) \longrightarrow_\alpha \lambda y.\, M[y/x] \quad \text{if } y \text{ is fresh for } M.$$

Yet, $M\ (\lambda x.\, x) \longrightarrow_\alpha M\ (\lambda y.\, y)$ does not hold.

Definition 13 (α-equivalence)

$$\frac{M_1 \to_\alpha M_2}{M_1 =_\alpha M_2} \qquad \frac{M_1 =_\alpha M_2}{(M_1\, N) =_\alpha (M_2\, N)}$$

$$\frac{M_1 =_\alpha M_2}{M_2 =_\alpha M_1} \qquad \frac{N_1 =_\alpha N_2}{(M\, N_1) =_\alpha (M\, N_2)}$$

$$\frac{M_1 =_\alpha M_2 \qquad M_2 =_\alpha M_3}{M_1 =_\alpha M_3} \qquad \frac{M_1 =_\alpha M_2}{(\lambda x.\, M_1) =_\alpha (\lambda x.\, M_2)}$$

$(\lambda y.\, y)\ (\lambda x.\, x) =_\alpha (\lambda x.\, x)\ (\lambda y.\, y)$. Why? Apply our rules!

Proof.

$$\dfrac{\dfrac{\dfrac{}{\lambda x.\, x \to_\alpha \lambda y.\, x[y/x]}}{\lambda x.\, x =_\alpha \lambda y.\, y}}{(\lambda y.\, y)\ (\lambda x.\, x) =_\alpha (\lambda y.\, y)\ (\lambda y.\, y)} \qquad \dfrac{\dfrac{\dfrac{}{\lambda y.\, y \to_\alpha \lambda x.\, y[x/y]}}{\lambda y.\, y =_\alpha \lambda x.\, x}}{(\lambda y.\, y)\ (\lambda y.\, y) =_\alpha (\lambda x.\, x)\ (\lambda y.\, y)}$$

$$(\lambda y.\, y)\ (\lambda x.\, x) =_\alpha (\lambda x.\, x)\ (\lambda y.\, y)$$

□

Exercise

Which of the following pairs are α-equivalent? Why?

1. x and y
2. $\lambda x\, y.\, y$ and $\lambda z\, y.\, y$
3. $\lambda x\, y.\, x$ and $\lambda y\, x.\, y$
4. $\lambda x\, y.\, x$ and $\lambda x\, y.\, y$

$(\lambda y.\, y\ y)\ (\lambda z.\, \lambda x.\, z\ x)$

Convention

Terms are equal up to α-equivalence of bound variables.

Feel free to rename any bound variable whenever convenient.

Untyped λ-Calculus: Dynamics

β-conversion

Definition 14 (β-conversion)

β-conversion is defined by

$$\underbrace{(\lambda x.\, M)\ N}_{\beta\text{-redex}} \longrightarrow_\beta M[N/x]$$

Good:

$$((\lambda x.\, \lambda y.\, x)\ M) \longrightarrow_\beta (\lambda y.\, x)[M/x] = \lambda y.\, x[M/x] = \lambda y.\, M$$

Bad:

$$((\lambda x\, y.\, x)\ M)\ N \longrightarrow_\beta ?$$

Full β-reduction

Definition 15

The full β-reduction is defined by

$$\frac{M_1 \longrightarrow_\beta M_2}{M_1 \longrightarrow_{\beta 1} M_2} \qquad \frac{M_1 \longrightarrow_{\beta 1} M_2}{M_1\ N \longrightarrow_{\beta 1} M_2\ N}$$

$$\frac{M_1 \longrightarrow_{\beta 1} M_2}{\lambda x.\, M_1 \longrightarrow_{\beta 1} \lambda x.\, M_2} \qquad \frac{N_1 \longrightarrow_{\beta 1} N_2}{M\ N_1 \longrightarrow_{\beta 1} M\ N_2}$$

$M \longrightarrow_{\beta *} M'$ denotes there are finitely many M_i such that

$$M_0 \longrightarrow_{\beta 1} M_1 \longrightarrow_{\beta 1} \ldots M_i \ldots \longrightarrow_{\beta 1} M'$$

with $M = M_0$ and $M_n = M'$ (where n can be 0).

α-conversion during β-conversion

Renaming of bound variables may need to happen during reduction:

$$\begin{aligned}
(\lambda y.\, y\ y)\ (\lambda z\, x.\, z\ x) &\longrightarrow_{\beta 1} && (\lambda z\, x.\, z\ x)\ (\lambda z\, x.\, z\ x) \\
&\longrightarrow_{\beta 1} && \lambda x.\, (\lambda z\, x.\, z\ x)\ x \\
&=_{\alpha} && \lambda x.\, (\lambda z\, y.\, z\ y)\ x \\
&\longrightarrow_{\beta 1} && \lambda x.\, (\lambda y.\, x\ y)
\end{aligned}$$

Even worse, we actually need infinitely many variables:

$$(\lambda y.\, y\ s\ y)\ (\lambda t\, z\, x.\, z\ (t\ x)\ z)$$

Exercise

Evaluate the above term.

Computational meaning

Definition 16

M and N have the same *computational meaning* if $M =_\beta N$ where $=_\beta$ is defined inductively by

$$\frac{M \longrightarrow_{\beta 1} N}{M =_\beta N} \qquad \frac{M =_\beta N}{N =_\beta M}$$

$$\frac{}{M =_\beta M} \qquad \frac{L =_\beta M \qquad M =_\beta N}{L =_\beta N}$$

- $\mathsf{c}_2 =_\beta (\lambda x\, y.y)\ \mathsf{c}_1\ \mathsf{c}_2$
- $\lambda z.\, (\lambda x\, y.x)\ z =_\beta \lambda z\, y.\, z$

Programming in λ-Calculus

Church encoding of boolean values

Boolean and conditional can be encoded as combinators.

Boolean

$$\begin{aligned} \texttt{True} &:= \lambda x\, y.\, x \\ \texttt{False} &:= \lambda x\, y.\, y \end{aligned}$$

Conditional

$$\texttt{if} := \lambda b\, x\, y.\, b\, x\, y$$

$$\texttt{if True}\ M\ N \longrightarrow_{\beta *} M$$

$$\texttt{if False}\ M\ N \longrightarrow_{\beta *} N$$

for any two λ-terms M and N.

Church Encoding of natural numbers i

Natural numbers can be encoded as λ-terms, so can arithmetic operations.

Church numerals

$$
\begin{array}{lcl}
\mathbf{c}_0 & := & \lambda f x.\, x \\
\mathbf{c}_1 & := & \lambda f x.\, f\, x \\
\mathbf{c}_2 & := & \lambda f x.\, f\, (f\, x) \\
\mathbf{c}_{n+1} & := & \lambda f x.\, f^{n+1}\, (x)
\end{array}
$$

where $f^1(x) := f\, x$ and $f^{n+1}(x) := f\, (f^n(x))$.

Church Encoding of natural numbers ii

Successor

$$\begin{array}{lll} \mathsf{succ} & := & \lambda n.\, \lambda f x.\, f\,(n\, f\, x) \\ \mathsf{succ}\ \mathsf{c}_n & \longrightarrow_{\beta *} & \mathsf{c}_{n+1} \end{array}$$

for any natural number $n \in \mathbb{N}$.

Addition

$$\begin{array}{lll} \mathsf{add} & := & \lambda n\, m.\, \lambda f x.\, n\, f\, (m\, f\, x) \\ \mathsf{add}\ \mathsf{c}_n\ \mathsf{c}_m & \longrightarrow_{\beta *} & \mathsf{c}_{n+m} \end{array}$$

Church Encoding of natural numbers iii

Conditional

$$\begin{aligned} \mathtt{ifz} &:= \lambda n\, x\, y.\, n\, (\lambda z.\, y)\, x \\ \mathtt{ifz}\ \mathtt{c}_0\ M\ N &\longrightarrow_{\beta *} M \\ \mathtt{ifz}\ \mathtt{c}_{n+1}\ M\ N &\longrightarrow_{\beta *} N \end{aligned}$$

Predecessor

$$\begin{aligned} \mathtt{pred} &:= \lambda n.\, \lambda f\, x.\, ? \\ \mathtt{pred}\ \mathtt{c}_0 &\longrightarrow_{\beta *} \mathtt{c}_0 \\ \mathtt{pred}\ \mathtt{c}_{n+1} &\longrightarrow_{\beta *} \mathtt{c}_n \end{aligned}$$

Exercise

1. Define the *flip* operation, i.e. a λ-term $\mathtt{flip}$ such that

$$\mathtt{flip}\ M\ N\ P =_\beta M\ P\ N$$

2. Define Boolean operations $\mathtt{not}$, $\mathtt{and}$, and $\mathtt{or}$.
3. Evaluate $\mathtt{succ}\ \mathtt{c}_0$ and $\mathtt{add}\ \mathtt{c}_1\ \mathtt{c}_2$.
4. Define the multiplication $\mathtt{mult}$ over Church numerals.
5. (Hard) Define $\mathtt{pred}$ so that $\mathtt{pred}\ \mathtt{c}_0 =_\beta \mathtt{c}_0$ and $\mathtt{pred}\ \mathtt{c}_{n+1} =_\beta \mathtt{c}_n$.

General Recursion, informally

The summation $\sum_{i=0}^{n} i$ for $n \in \mathbb{N}$ *can be* defined as

$$\mathtt{sum}(n) = \begin{cases} 0 & \text{if } n = 0 \\ n + \mathtt{sum}(n-1) & \text{otherwise.} \end{cases}$$

Can we avoid the self-reference? Consider the function $G\colon (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$ defined by

$$(Gf)(n) := \begin{cases} 0 & \text{if } n = 0 \\ n + f(n-1) & \text{otherwise.} \end{cases} \tag{1}$$

If $G(\mathtt{sum}') = \mathtt{sum}'$, then $\mathtt{sum}' = \mathtt{sum}$.

Curry's paradoxical combinator

Proposition 17

Define

$$\mathsf{Y} := \lambda f.\,(\lambda x.\,f\,(x\,x))\,(\lambda x.\,f\,(x\,x)).$$

Then,

$$\begin{aligned}\mathsf{Y}F &\longrightarrow_{\beta 1} (\lambda x.\,F\,(x\,x))\,(\lambda x.\,F\,(x\,x)) \\ &\longrightarrow_{\beta 1} F\,((\lambda x.\,F\,(x\,x))\,(\lambda x.\,F\,(x\,x)))\end{aligned}$$

for every λ-term F.

Summation, formally

Using the combinators we have known so far, the equation (1) can be defined as λ-terms:

$$G := \lambda f\, n.\, \mathtt{ifz}\ n\ \mathbf{c}_0\ (\mathtt{add}\ n\ (f\ (\mathtt{pred}\ n)))$$

$$\mathtt{sum} := \mathbf{Y} G$$

For example

$$\begin{aligned}
\mathtt{sum}\ \mathbf{c}_1 &\equiv (\mathbf{Y} G)\ \mathbf{c}_1 \\
&\longrightarrow_{\beta 1} G'\ \mathbf{c}_1 \\
&\longrightarrow_{\beta 1} G\ G'\ \mathbf{c}_1 \\
&\longrightarrow_{\beta 1} (\lambda n.\, \mathtt{ifz}\ n\ \mathbf{c}_0\ (\mathtt{add}\ n\ (G'\ (\mathtt{pred}\ n))))\ \mathbf{c}_1 \\
&\longrightarrow_{\beta 1} \mathtt{ifz}\ \mathbf{c}_1\ \mathbf{c}_0\ (\mathtt{add}\ \mathbf{c}_1\ (G'\ (\mathtt{pred}\ \mathbf{c}_1))) \\
&\longrightarrow_{\beta 1} \cdots
\end{aligned}$$

where $G' := ((\lambda x.\, G\ (x\ x))\ (\lambda x.\, G\ (x\ x)))$.

Turing's fixed-point combinator

Here is a fixed-point operator such that $\Theta F \longrightarrow_{\beta *} F(\Theta F)$.

Proposition 18

Define

$$\Theta := (\lambda x f.\, f\,(x\,x\,f))\ (\lambda x f.\, f\,(x\,x\,f))$$

Then,

$$\Theta F \longrightarrow_{\beta *} F(\Theta F)$$

Try Turing's fixed-point combinator with G to define $\sum_{i=0}^{n} i$.

$$G := \lambda f n.\, \mathtt{ifz}\ n\ \mathbf{c}_0\ (\mathtt{add}\ n\ (f\,(\mathtt{pred}\ n)))$$

$$\mathtt{sum} := \Theta\, G$$

Exercise

1. Evaluate $\mathtt{sum}\ \mathbf{c}_1$ to its normal form in detail.
2. Define the factorial $n!$ on Church numerals with Turing's fixed-point combinator.

Properties of λ-Calculus

Example 19

Suppose $M : \mathtt{Term}_\lambda$ and $y \notin \mathbf{FV}(M)$. Then, consider

$$(\lambda y.\, M)\,((\lambda x.\, x\, x)(\lambda x.\, x\, x))$$

Observations:

- Some evaluation may diverge while some may converge.
- Full β-reduction lacks for determinacy.

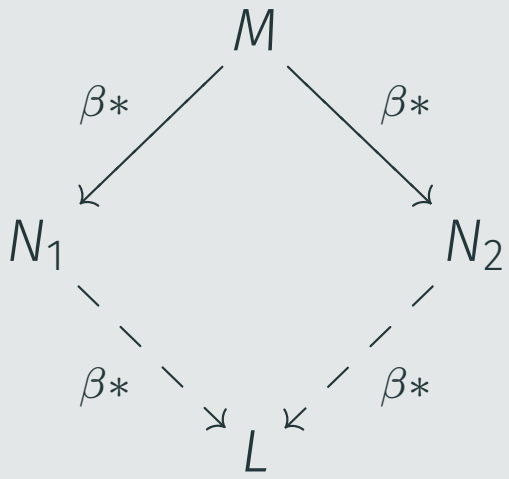
Question:

- Does every path give the same evaluation?

Confluence

Theorem 20 (Church-Rosser)

Given N_1 and N_2 with $M \longrightarrow_{\beta *} N_1$ and $M \longrightarrow_{\beta *} N_2$, there is L such that



Normal form

M is in *normal form* if $M \not\longrightarrow_{\beta 1}$.

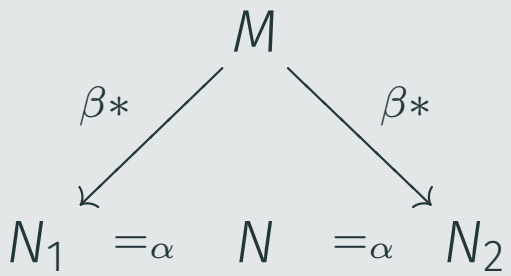
Lemma 21

Suppose that M is in normal form. Then

$M \longrightarrow_{\beta *} N \implies M =_\alpha N.$

Corollary 22 (Uniqueness of normal forms)

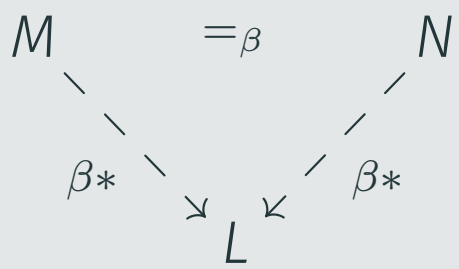
Suppose that N_1 and N_2 are in normal form. Then,



Computationally equal terms have a confluent term

Corollary 23

If $M =_\beta N$, then there exists L satisfying



Proof sketch.

By induction on the derivation of $M =_\beta N$.

For example, if $M \longrightarrow_{\beta 1} N$, then choose L as N. □

Evaluation strategies i

An evaluation strategy is a procedure of selecting β-redexes to reduce. It is a subset $\longrightarrow_{\mathrm{ev}}$ of the full β-reduction $\longrightarrow_{\beta 1}$.

Innermost β-redex does not contain any β-redex.

Outermost β-redex is not contained in any other β-redex.

Evaluation strategies ii

the leftmost-outermost (*normal order*) strategy reduces the leftmost outermost β-redex in a term first. For example,

$$
\begin{aligned}
& \underline{(\lambda x.\,(\lambda y.\,y)\,x)} \quad \underline{(\lambda x.\,(\lambda y.\,y\,y)\,x)} \\
\longrightarrow_{\beta 1} & \underline{(\lambda y.\,y)} \quad \underline{(\lambda x.\,(\lambda y.\,y\,y)\,x)} \\
\longrightarrow_{\beta 1} & \lambda x.\,\underline{(\lambda y.\,y\,y)} \quad \underline{x} \\
\longrightarrow_{\beta 1} & (\lambda x.\,x\,x) \\
\not\longrightarrow_{\beta 1} &
\end{aligned}
$$

Evaluation strategies iii

the leftmost-innermost strategy reduces the leftmost innermost β-redex in a term first. For example,

$$
\begin{aligned}
& (\lambda x.\, \underline{(\lambda y.\, y)}\ \ \underline{x})\, (\lambda x.\, (\lambda y.\, y\, y)\, x) \\
\longrightarrow_{\beta 1} & (\lambda x.\, x)\ (\lambda x.\, \underline{(\lambda y.\, y\, y)}\ \ \underline{x}) \\
\longrightarrow_{\beta 1} & \underline{(\lambda x.\, x)}\ \ \underline{(\lambda x.\, x\, x)} \\
\longrightarrow_{\beta 1} & (\lambda x.\, x\, x) \\
\not\longrightarrow_{\beta 1} &
\end{aligned}
$$

the rightmost-innermost/outermost strategy are defined similarly where terms are reduced from right to left instead.

CBV versus CBN

Call-by-value strategy rightmost-outermost but not under any abstraction

Call-by-name strategy leftmost-outermost but not under any abstraction

Proposition 24 (Determinacy)

Each of evaluation strategies is deterministic, i.e. if $M \longrightarrow_{\beta 1} N_1$ *and* $M \longrightarrow_{\beta 1} N_2$ *then* $N_1 = N_2$.

Exercise

Define following terms

$$\Omega := (\lambda x.\, x\, x)\ (\lambda x.\, x\, x)$$
$$\mathsf{K}_1 := \lambda x\, y.\, x$$

Evaluate

$$\mathsf{K}_1\ z\ \Omega$$

using the call-by-value and the call-by-name strategy respectively.

Normalisation

Definition 25

1. M is in *normal form* if $M \not\rightarrow_{\beta 1} N$ for any N.
2. M is *weakly normalising* if $M \longrightarrow_{\beta *} N$ for some N in normal form.

1. Ω is not weakly normalising.
2. K_1 is normal and thus weakly normalising.
3. $\mathsf{K}_1\ z\ \Omega$ is weakly normalising.

Theorem 26

The normal order strategy reduces every weakly normalising term to a normal form.

Homework

1. (25%) Show that $\longrightarrow_{\beta *}$ is transitive, i.e. $L \longrightarrow_{\beta *} N$ whenever $L \longrightarrow_{\beta *} M$ and $M \longrightarrow_{\beta *} N$. **Hint**. By induction on $L \longrightarrow_{\beta *} M$.
2. (25%) Show Lemma 21
3. (25%) Show Corollary 22
4. (25%) Show Corollary 23.