Programmation Fonctionnelle
Fondaments de la programmation fonctionnelle :
le λ-calcul non typé

Luigi Santocanale
LIF, Aix-Marseille Université
Marseille, FRANCE

7 novembre 2016

Plan

Le λ-calcul

Confluence

Stratégies d'évaluations

Plan

Le λ-calcul

Confluence

Stratégies d'évaluations

Les λ-termes

L'ensemble des λ-termes (ou expressions) est défini par la grammaire :

$$T := x \mid T\ T \mid \lambda x.T$$

Exemple. Les suivants sont des λ-termes :

$$\lambda x.xx\,,\ (\lambda x.(xx))y\,,\ \ldots$$

Un λ-terme de la forme $t\ s$ est appelé *application* ; $\lambda x.t$ est appelé *abstraction*.

Les λ-termes

L'ensemble des λ-termes (ou expressions) est défini par la grammaire :

$$T := x \mid T\ T \mid \lambda x.T$$

Exemple. Les suivants sont des λ-termes :

$$\lambda x.xx\,,\ (\lambda x.(xx))y\,,\ \ldots$$

Un λ-terme de la forme $t\ s$ est appelé *application* ;

$\lambda x.t$ *abstraction* .

Les λ-termes

L'ensemble des λ-termes (ou expressions) est défini par la grammaire :

$$T := x \mid T\ T \mid \lambda x.T$$

Exemple. Les suivants sont des λ-termes :

$$\lambda x.xx\,,\ (\lambda x.(xx))y\,,\ \ldots$$

Un λ-terme de la forme $\begin{matrix} t\ s \\ \lambda x.t \end{matrix}$ est appelé $\begin{matrix} \textit{application} \\ \textit{abstraction} \end{matrix}$.

Les λ-termes

L'ensemble des λ-termes (ou expressions) est défini par la grammaire :

$$T := x \mid T\ T \mid \lambda x.T$$

Exemple. Les suivants sont des λ-termes :

$$\lambda x.xx\,,\ (\lambda x.(xx))y\,,\ \ldots$$

Un λ-terme de la forme $\begin{matrix} t\ s \\ \\ \lambda x.t \end{matrix}$ est appelé $\begin{matrix} \textit{application}\,; \\ \\ \textit{abstraction}\,. \end{matrix}$

Variables libres et variables liées

Variable libres (Free) :

$$\begin{aligned} \mathrm{FV}(x) &= \{x\}, \\ \mathrm{FV}(t\,s) &= \mathrm{FV}(t) \cup \mathrm{FV}(s), \\ \mathrm{FV}(\lambda x.t) &= \mathrm{FV}(t) \setminus \{x\}. \end{aligned}$$

Variables liées (Bound) :

$$\begin{aligned} \mathrm{BV}(x) &= \emptyset, \\ \mathrm{BV}(t\,s) &= \mathrm{BV}(t) \cup \mathrm{BV}(s), \\ \mathrm{BV}(\lambda x.t) &= \mathrm{BV}(t) \cup \{x\}. \end{aligned}$$

Substitution (I)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \ (***). \end{cases}$$

$(***)$: Cette règle s'applique sous la contrainte

$$\mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset,$$

pour éviter le « variable capture », c'est-à-dire qu'une variable libre dans r devient liée après substitution—comme par exemple dans :

$$(\lambda x.xy)[x/y] \overset{?}{\rightsquigarrow} \lambda x.xx.$$

Substitution (I)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \ (***). \end{cases}$$

$(***)$: Cette règle s'applique sous la contrainte

$$\mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset,$$

pour éviter le « variable capture », c'est-à-dire qu'une variable libre dans r devient liée après substitution—comme par exemple dans :

$$(\lambda x.xy)[x/y] \overset{?}{\rightsquigarrow} \lambda x.xx.$$

Substitution (I)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \ (***). \end{cases}$$

$(***)$: Cette règle s'applique sous la contrainte

$$\mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset,$$

pour éviter le « variable capture », c'est-à-dire qu'une variable libre dans r devient liée après substitution—comme par exemple dans :

$$(\lambda x.xy)[x/y] \overset{?}{\rightsquigarrow} \lambda x.xx.$$

Substitution (I)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \ (***). \end{cases}$$

$(***)$: Cette règle s'applique sous la contrainte

$$\mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset,$$

pour éviter le « variable capture », c'est-à-dire qu'une variable libre dans r devient liée après substitution—comme par exemple dans :

$$(\lambda x.xy)[x/y] \overset{?}{\rightsquigarrow} \lambda x.xx.$$

α-équivalence

La substitution est partiellement définie :
comment définir $(\lambda x.(xy))[x/y]$?

On peut répondre en déclarant équivalentes les abstractions par rapport à variables différentes (invariance d'une fonction par rapport aux paramètres formels). Par exemple :

$$\lambda x.xy \sim \lambda z.zy \qquad (\text{mais } \lambda x.xy \not\sim \lambda y.yy)\,.$$

Règle générale :

$$x \sim_\alpha x\,,$$
$$t \sim_\alpha t' \text{ et } s \sim_\alpha s' \quad \text{implique} \quad t\,s \sim_\alpha t'\,s'\,,$$
$$t \sim_\alpha t' \text{ et } z \notin \mathrm{FV}(t') \cup \mathrm{BV}(t') \quad \text{implique} \quad \lambda x.t \sim_\alpha \lambda z.t'[z/x]\,.$$

Si $t \sim_\alpha s$, alors on dit que t et s sont α-équivalentes,
ou bien que s est une α-variante de t.

α-équivalence

La substitution est partiellement définie :

comment définir $(\lambda x.(xy))[x/y]$?

On peut répondre en déclarant équivalentes les abstractions par rapport à variables différentes (invariance d'une fonction par rapport aux paramètres formels). Par exemple :

$$\lambda x.xy \sim \lambda z.zy \qquad (\text{mais } \lambda x.xy \not\sim \lambda y.yy)\,.$$

Règle générale :

$$x \sim_\alpha x\,,$$
$$t \sim_\alpha t' \text{ et } s \sim_\alpha s' \quad \text{implique} \quad t\,s \sim_\alpha t'\,s'\,,$$
$$t \sim_\alpha t' \text{ et } z \notin \mathrm{FV}(t') \cup \mathrm{BV}(t') \quad \text{implique} \quad \lambda x.t \sim_\alpha \lambda z.t'[z/x]\,.$$

Si $t \sim_\alpha s$, alors on dit que t et s sont α-équivalentes,

ou bien que s est une α-variante de t.

α-équivalence

La substitution est partiellement définie :

comment définir $(\lambda x.(xy))[x/y]$?

On peut répondre en déclarant équivalentes les abstractions par rapport à variables différentes (invariance d'une fonction par rapport aux paramètres formels). Par exemple :

$$\lambda x.xy \sim \lambda z.zy \qquad (\text{mais } \lambda x.xy \not\sim \lambda y.yy)\,.$$

Règle générale :

$$\begin{array}{rcl} & & x \sim_\alpha x\,, \\ t \sim_\alpha t' \text{ et } s \sim_\alpha s' & \text{implique} & t\,s \sim_\alpha t'\,s'\,, \\ t \sim_\alpha t' \text{ et } z \notin \mathsf{FV}(t') \cup \mathsf{BV}(t') & \text{implique} & \lambda x.t \sim_\alpha \lambda z.t'[z/x]\,. \end{array}$$

Si $t \sim_\alpha s$, alors on dit que t et s sont α-équivalentes,
ou bien que s est une α-variante de t.

α-équivalence

La substitution est partiellement définie :

comment définir $(\lambda x.(xy))[x/y]$?

On peut répondre en déclarant équivalentes les abstractions par rapport à variables différentes (invariance d'une fonction par rapport aux paramètres formels). Par exemple :

$$\lambda x.xy \sim \lambda z.zy \qquad (\text{mais } \lambda x.xy \not\sim \lambda y.yy)\,.$$

Règle générale :

$$\begin{array}{rcl} & & x \sim_\alpha x\,, \\ t \sim_\alpha t' \text{ et } s \sim_\alpha s' & \text{implique} & t\,s \sim_\alpha t'\,s'\,, \\ t \sim_\alpha t' \text{ et } z \notin \mathrm{FV}(t') \cup \mathrm{BV}(t') & \text{implique} & \lambda x.t \sim_\alpha \lambda z.t'[z/x]\,. \end{array}$$

Si $t \sim_\alpha s$, alors on dit que t et s sont α-équivalentes,

ou bien que s est une α-variante de t.

Substitution (II)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \text{ et } \mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset, \\ v[r/y], & \text{sinon,} \\ & \text{où } v \sim_\alpha \lambda x.t \text{ et} \\ & \mathrm{BV}(v) \cap \mathrm{FV}(r) = \emptyset. \end{cases}$$

Exemple : $\lambda x.xy \sim_\alpha \lambda z.zy$, donc :

$$(\lambda x.xy)[x/y] = (\lambda z.zy)[x/y] = \lambda z.(z[x/y]y[x/y]) = \lambda z.zx.$$

Substitution (II)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \text{ et } \mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset, \\ v[r/y], & \text{sinon,} \\ & \text{où } v \sim_\alpha \lambda x.t \text{ et} \\ & \mathrm{BV}(v) \cap \mathrm{FV}(r) = \emptyset. \end{cases}$$

Exemple : $\lambda x.xy \sim_\alpha \lambda z.zy$, donc :

$$(\lambda x.xy)[x/y] = (\lambda z.zy)[x/y] = \lambda z.(z[x/y]y[x/y]) = \lambda z.zx.$$

Substitution (II)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \text{ et } \mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset, \\ v[r/y], & \text{sinon,} \\ & \quad \text{où } v \sim_\alpha \lambda x.t \quad \text{et} \\ & \quad\quad \mathrm{BV}(v) \cap \mathrm{FV}(r) = \emptyset. \end{cases}$$

Exemple : $\lambda x.xy \sim_\alpha \lambda z.zy$, donc :

$$(\lambda x.xy)[x/y] = (\lambda z.zy)[x/y] = \lambda z.(z[x/y]y[x/y]) = \lambda z.zx.$$

Substitution (II)

$$x[r/y] = \begin{cases} x, & \text{si} \quad x \neq y, \\ r, & \text{si} \quad x = y; \end{cases}$$

$$(t\ s)[r/y] = t[r/y]\ s[r/y];$$

$$(\lambda x.t)[r/y] = \begin{cases} \lambda x.t, & \text{si } x = y, \\ \lambda x.(t[r/y]), & \text{si } x \neq y \text{ et } \mathrm{BV}(\lambda x.t) \cap \mathrm{FV}(r) = \emptyset, \\ v[r/y], & \text{sinon,} \\ & \quad \text{où } v \sim_\alpha \lambda x.t \quad \text{et} \\ & \quad\quad \mathrm{BV}(v) \cap \mathrm{FV}(r) = \emptyset. \end{cases}$$

Exemple : $\lambda x.xy \sim_\alpha \lambda z.zy$, donc :

$$(\lambda x.xy)[x/y] = (\lambda z.zy)[x/y] = \lambda z.(z[x/y]y[x/y]) = \lambda z.zx.$$

β-réduction

Loi fondamentale du λ-calcul :

$$(\lambda x.t)\ s \to_\beta t[x/s]$$

> *La fonction $\lambda x.t$ appliquée à son argument s,*
> *s'évalue à t, le corps de la fonction,*
> *avec le paramètre formel x remplacé par l'argument s.*

On appelle redex un λ-terme de la forme

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

i.e. une application dont l'opérande gauche est une abstraction.

Évaluation

Extension de la β-réduction en contexte :

$$
\begin{array}{rl}
(\lambda x.t)\, s \rightsquigarrow_\beta t[s/x], & (\beta\text{-réduction}) \\
t\, s \rightsquigarrow_\beta t'\, s, & \text{si } t \rightsquigarrow_\beta t', \\
t\, s \rightsquigarrow_\beta t\, s', & \text{si } s \rightsquigarrow_\beta s', \\
\lambda x.t \rightsquigarrow_\beta \lambda x.t', & \text{si } t \rightsquigarrow_\beta t'.
\end{array}
$$

Definition

Un λ-terme t est un valeur s'il n'existe pas un autre λ-terme s tel que $t \not\rightsquigarrow_\beta s$.

Exemple. Une variable x est un valeur.

Évaluation

Extension de la β-réduction en contexte :

$$
\begin{array}{rl}
(\lambda x.t)\, s \rightsquigarrow_\beta t[s/x], & (\beta\text{-réduction}) \\
t\, s \rightsquigarrow_\beta t'\, s, & \text{si } t \rightsquigarrow_\beta t', \\
t\, s \rightsquigarrow_\beta t\, s', & \text{si } s \rightsquigarrow_\beta s', \\
\lambda x.t \rightsquigarrow_\beta \lambda x.t', & \text{si } t \rightsquigarrow_\beta t'.
\end{array}
$$

Definition

Un λ-terme t est un valeur s'il n'existe pas un autre λ-terme s tel que $t \not\rightsquigarrow_\beta s$.

Exemple. Une variable x est un valeur.

Notation

On pose :

$$t \overset{n}{\rightsquigarrow}_\beta s,$$

s'il existe une suite

$$t = t_0 \rightsquigarrow_\beta t_1 \rightsquigarrow_\beta \dots t_{n-1} \rightsquigarrow_\beta t_n = s.$$

On pose :

$$t \overset{*}{\rightsquigarrow}_\beta s,$$

s'il existe $n \geq 0$ tel que $t \overset{n}{\rightsquigarrow}_\beta s$.

Notation

On pose :

$$t \overset{n}{\rightsquigarrow}_\beta s,$$

s'il existe une suite

$$t = t_0 \rightsquigarrow_\beta t_1 \rightsquigarrow_\beta \dots t_{n-1} \rightsquigarrow_\beta t_n = s.$$

On pose :

$$t \overset{*}{\rightsquigarrow}_\beta s,$$

s'il existe $n \geq 0$ tel que $t \overset{n}{\rightsquigarrow}_\beta s$.

Exemple : encodage de Church des nombres naturels

$$\hat{0} = \lambda f.\lambda x.x$$
$$\hat{1} = \lambda f.\lambda x.fx$$
$$\hat{2} = \lambda f.\lambda x.f(fx)$$
$$\hat{3} = \ldots$$
$$SUCC = \lambda n.\lambda f.\lambda x.f(nfx)$$
$$PLUS = \lambda m.\lambda n.\lambda f.\lambda x.mf(nfx)$$

Exemples

$$\mathit{SUCC}\ \hat{1} = \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x$$
$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x)$$
$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x)$$
$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},$$

$$\mathit{PLUS}\ \hat{1}\ \hat{1} = \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x$$
$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$
$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$
$$\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x)$$
$$\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x)$$
$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.$$

Exemples

$$SUCC\ \hat{1} = \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},$$

$$PLUS\ \hat{1}\ \hat{1} = \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x)$$

$$\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.$$

Exemples

$$SUCC\ \hat{1} = \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},$$

$$PLUS\ \hat{1}\ \hat{1} = \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x)$$

$$\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x)$$

$$\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.$$

Exemples

$$\begin{aligned} SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}. \end{aligned}$$

$$\begin{aligned} PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\ &\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}. \end{aligned}$$

Exemples

$$\begin{aligned} SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}. \end{aligned}$$

$$\begin{aligned} PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\ &\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\ &\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\ &\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}. \end{aligned}$$

Exemples

$$
\begin{aligned}
SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
SUCC\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
PLUS\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Exemples

$$
\begin{aligned}
\mathit{SUCC}\ \hat{1} &= \lambda n.\lambda f.\lambda x.f\,(n\,f\,x)\ \ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda f.\lambda x.f\,x)\,f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,((\lambda x.f\,x)\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2},
\end{aligned}
$$

$$
\begin{aligned}
\mathit{PLUS}\ \hat{1}\ \hat{1} &= \lambda m.\lambda n.\lambda f.\lambda x.m\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.f\,x)\,f\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda n.\lambda f.\lambda x.(\lambda x.f\,x)\,(n\,f\,x)\ \lambda f.\lambda x.f\,x \\
&\rightsquigarrow_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(\lambda f.\lambda x.f\,x\,f\,x) \\
&\overset{2}{\rightsquigarrow}_\beta \lambda f.\lambda x.(\lambda x.f\,x)\,(f\,x) \\
&\rightsquigarrow_\beta \lambda f.\lambda x.f\,(f\,x) = \hat{2}.
\end{aligned}
$$

Des théoremes

Théorème

Une fonction

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

est calculable par une machine de Turing si et seulement si il existe un λ-terme t tel que

$$t\ \hat{n} \overset{*}{\rightsquigarrow}_{\beta} \hat{m}, \qquad \text{pour tous entiers } n \text{ et } m \text{ tels que } f(n) = m.$$

Plan

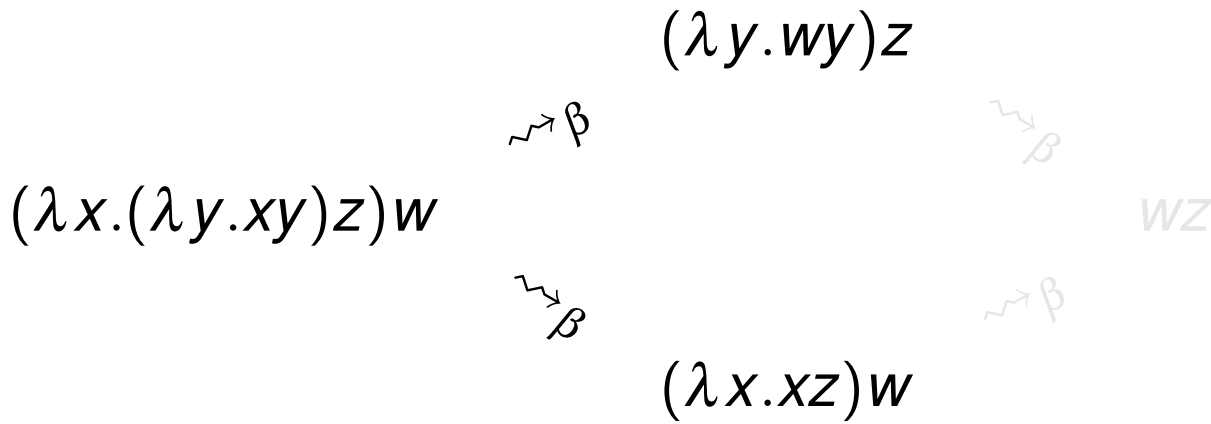
Le λ-calcul

Confluence

Stratégies d'évaluations

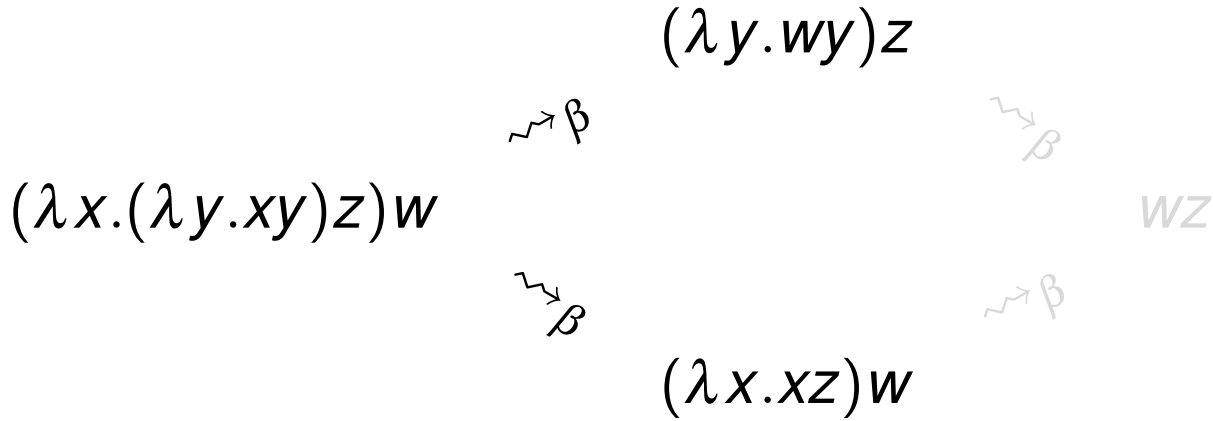
L'évaluation (β-réduction en contexte) n'est pas déterministe.

Par exemple, mais :



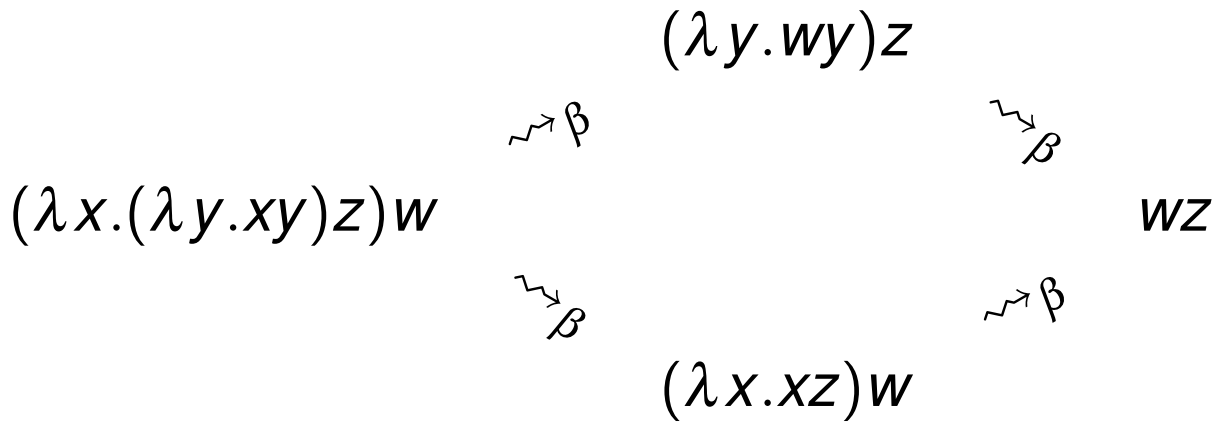
L'évaluation (β-réduction en contexte) n'est pas déterministe.

Par exemple, mais :



L'évaluation (β-réduction en contexte) n'est pas déterministe.

Par exemple, mais :



Confluence du λ-calcul

Theorem

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Il existe alors un λ-terme t_3 tel que $t_1 \rightsquigarrow_\beta t_3$ et $t_3 \rightsquigarrow_\beta t_3$.

Corollary

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Si t_1 et t_2 sont des valeurs, alors $t_1 = t_2$.

Confluence du λ-calcul

Theorem

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Il existe alors un λ-terme t_3 tel que $t_1 \rightsquigarrow_\beta t_3$ et $t_3 \rightsquigarrow_\beta t_3$.

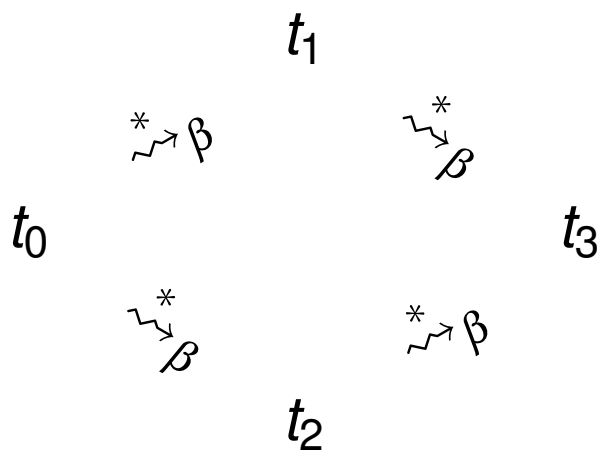
Corollary

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Si t_1 et t_2 sont des valeurs, alors $t_1 = t_2$.

Confluence du λ-calcul

Theorem

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Il existe alors un λ-terme t_3 tel que $t_1 \rightsquigarrow_\beta t_3$ et $t_3 \rightsquigarrow_\beta t_3$.



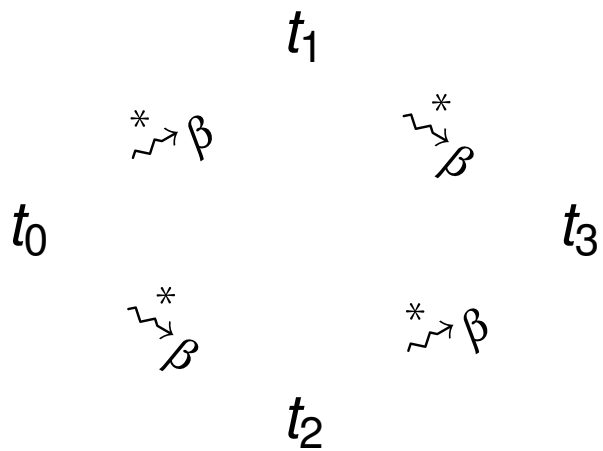
Corollary

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Si t_1 et t_2 sont des valeurs, alors $t_1 = t_2$.

Confluence du λ-calcul

Theorem

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Il existe alors un λ-terme t_3 tel que $t_1 \rightsquigarrow_\beta t_3$ et $t_3 \rightsquigarrow_\beta t_3$.



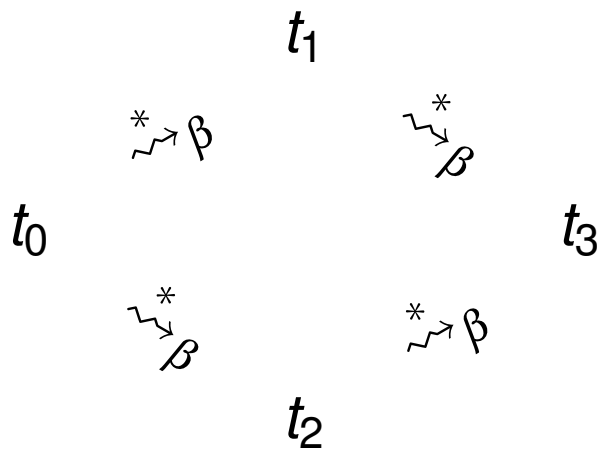
Corollary

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Si t_1 et t_2 sont des valeurs, alors $t_1 = t_2$.

Confluence du λ-calcul

Theorem

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Il existe alors un λ-terme t_3 tel que $t_1 \rightsquigarrow_\beta t_3$ et $t_3 \rightsquigarrow_\beta t_3$.



Corollary

Soient t_0, t_1, t_2 trois λ-termes tels que $t_0 \overset{*}{\rightsquigarrow}_\beta t_1$ et $t_0 \overset{*}{\rightsquigarrow}_\beta t_2$.
Si t_1 et t_2 sont des valeurs, alors $t_1 = t_2$.

Un λ-terme un peu bizarre

Soit

$$\omega := (\lambda x.xx)(\lambda x.xx)$$

Il n'existe aucun valeur t tel que $\omega \overset{*}{\rightsquigarrow}_\beta t$.

Plan

Le λ-calcul

Confluence

Stratégies d'évaluations

Comment rendre l'évaluation déterministe ?

Definition

Soit t un λ-terme. Un redex de t est un sous-terme t' de t de la forme

$$t' = (\lambda x.s)\ r\,.$$

Réduire un redex : substituer un tel t' par $s[r/x]$ dans t.
Par exemple, dans

$$t = (\lambda x.(\lambda y.xy)z)\ w$$

nous avons deux redexes :

- t : le λ-terme t lui-même ;
- $(\lambda y.xy)z$.

Comment rendre l'évaluation déterministe ?

Definition

Soit t un λ-terme. Un redex de t est un sous-terme t' de t de la forme

$$t' = (\lambda x.s)\ r\,.$$

Réduire un redex : substituer un tel t' par $s[r/x]$ dans t.

Par exemple, dans

$$t = (\lambda x.(\lambda y.xy)z)\ w$$

nous avons deux redexes :

1. le λ-terme t lui-même :

$$(\lambda x.(\lambda y.xy)z)\ w$$

2. le sous-terme $t' = (\lambda y.xy)\ z$:

$$(\lambda x.\ \ \ \ \ )\ w$$

Comment rendre l'évaluation déterministe ?

Definition

Soit t un λ-terme. Un redex de t est un sous-terme t' de t de la forme

$$t' = (\lambda x.s)\ r\,.$$

Réduire un redex : substituer un tel t' par $s[r/x]$ dans t.
Par exemple, dans

$$t = (\lambda x.(\lambda y.xy)z)\ w$$

nous avons deux redexes :

1. le λ-terme t lui-même :

$$(\lambda x.(\lambda y.xy)z)\ w$$

2. le sous-terme $t' = (\lambda y.xy)z$:

$$(\lambda y.xy)\ z.$$

Comment rendre l'évaluation déterministe ?

Definition

Soit t un λ-terme. Un redex de t est un sous-terme t' de t de la forme

$$t' = (\lambda x.s)\ r\,.$$

Réduire un redex : substituer un tel t' par $s[r/x]$ dans t.
Par exemple, dans

$$t = (\lambda x.(\lambda y.xy)z)\ w$$

nous avons deux redexes :

1. le λ-terme t lui-même :

$$(\lambda x.(\lambda y.xy)z)\ w$$

2. le sous-terme $t' = (\lambda y.xy)z$:

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

Comment rendre l'évaluation déterministe ?

Definition

Soit t un λ-terme. Un redex de t est un sous-terme t' de t de la forme

$$t' = (\lambda x.s)\ r\,.$$

Réduire un redex : substituer un tel t' par $s[r/x]$ dans t.
Par exemple, dans

$$t = (\lambda x.(\lambda y.xy)z)\ w$$

nous avons deux redexes :

1. le λ-terme t lui-même :

$$(\lambda x.(\lambda y.xy)z)\ w$$

2. le sous-terme $t' = (\lambda y.xy)z$:

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

Dans un même λ-terme, deux redexes peuvent être :

1. l'un plus extérieur (resp. plus intérieur) que l'autre.

$$(\lambda x.(\lambda y.xy)z)\, w \quad \text{est plus exterieur que} \quad (\lambda y.xy)\, z\,;$$

2. l'un plus à gauche que l'autre. Dans

$$(\lambda g.\lambda x.g\, x)\, h((\lambda y.f\, y)\, z)\,,$$

$$(\lambda g.\lambda x.g\, x)\, h \quad \text{est plus à gauche que} \quad (\lambda y.f\, y)\, z\,.$$

Les stratégies d'évaluation

Ordre applicatif (évaluation par l'intérieur) : le redex le plus à l'intérieur, le plus à gauche est réduit.

Ordre normal (évaluation par l'extérieur) : le redex le plus à l'extérieur, le plus à gauche est réduit.

Exemple :

$$(\lambda x.(\lambda y.xy)z)\ w \rightsquigarrow_\beta \ldots$$

Les stratégies d'évaluation

Ordre applicatif (évaluation par l'intérieur) : le redex le plus à l'intérieur, le plus à gauche est réduit.

Ordre normal (évaluation par l'extérieur) : le redex le plus à l'extérieur, le plus à gauche est réduit.

Exemple :

$$(\lambda x.(\lambda y.xy)z)\ w \rightsquigarrow_\beta \ldots$$

Call by name : comme pour l'ordre normal, mais pas de réduction sous une abstraction.

Call by need (évaluation paresseuse) : comme pour l'ordre normal, mais en utilisant le partage.

Call by value : seulement les redexes extérieurs sont réduits, une fois que l'argument a été réduit à un valeur.

C'est-à-dire : comme pour l'ordre applicatif, mais pas de réduction sous une abstraction.