929

# Lambda calcul pur comme modèle de calcul. Exemples.

(*) M et N sont alors des sous termes de (MN)  (**) M est alors un sous terme de (λx.M)

But : Avoir un modèle de calcul fonctionnel pour calculer des fonctions

## I Termes de λ-calcul : syntaxe

**Def 1** : On se donne un ensemble de variables VAR dénombrable $\{x, y, z, \dots\}$. On définit les termes de λ-calcul par induction
(i) Les variables sont des termes
(*) (ii) Si M et N sont des termes, alors (MN) est un terme
(**) (iii) Si M est un terme, alors $(\lambda x.M)$ est un terme où $x \in VAR$

**Ex 2** : $(\lambda x.(xy))$ ; $((\lambda y.y)(\lambda x.(xy)))$ sont des termes
$(x.(\lambda x))$ n'est pas un terme.

**Rmq 3** : On notera MNP pour ((MN)P) où M, N, P termes
et $\lambda xy.M$ pour $(\lambda x.(\lambda y.M))$ où M terme et $x, y \in VAR$

**Def 4** : La longueur d'un terme est définie par induction
$lg(x) = 1$ pour $x \in VAR$
$lg(MN) = lg(M) + lg(N)$ pour M, N termes
$lg(\lambda x.M) = 1 + lg(M)$ pour $x \in VAR$ et M un terme.

**Def 5** : Dans un sous terme $\lambda x.M$ on dit que M et ses sous termes sont dans la portée de $\lambda x$

**Def 6** (lié) Une occurrence x est liée dans un terme P si x est un sous terme de P et x est dans la portée d'un $\lambda x$ qui est dans P.

**Def 7** (libre) Une occurrence x est libre dans un terme P si elle n'est pas liée et qu'elle est dans le terme P

**Ex 8** : $\lambda x.y$ on a y est libre
$(x(\lambda x.x))$ la première occurrence de x est libre
la deuxième est liée
$((\lambda y.y)(\lambda x.(xy)))$ le 1er y lié, le 2ème est libre, x est lié

**Def 9** : Si un terme n'a pas de variable libre, il est dit clos.

**Ex 10** : $I = \lambda x.x$  $K = \lambda xy.x$  $S = \lambda xyz.xz(yz)$ sont 3 termes clos.

**Def 11** (Substitution) Soit M, N des termes, $x \in VAR$
La substitution de x par N dans M est le remplacement de chaque occurrence libre de x dans M par N en renommant si besoin les variables libres de N pour qu'elles restent libres dans $[N/x]M$ où l'on a noté la substitution $[N/x]M$. → par des variables fraîches

**Def 12** : On définit l'α-conversion comme la plus petite relation d'équivalence $\equiv_\alpha$ satisfaisant :
(i) $\lambda x.M \equiv_\alpha \lambda y.([y/x]M)$ si y n'est pas libre dans M
(ii) $\lambda x.M \equiv_\alpha \lambda x.N$ si $M \equiv_\alpha N$
(iii) $MN \equiv_\alpha PQ$ si $M \equiv_\alpha P$ et $N \equiv_\alpha Q$

**Ex 13** : $\lambda xy.x(xy) \equiv_\alpha \lambda uv.u(uv)$

## II β-réduction

**Def 14** : La β-réduction est définie par induction sur les termes
- $(\lambda x.M)N \longrightarrow_\beta [N/x]M$
- Si $M \to_\beta M'$, alors $\lambda x.M \to_\beta \lambda x.M'$
- Si $M \to_\beta M'$, alors $MN \to_\beta M'N$
- Si $N \to_\beta N'$, alors $MN \to_\beta MN'$

**Ex 15** : $(\lambda xy.yx)uv \to_\beta (\lambda y.yu)v \to_\beta vu$
$(\lambda x.xxy)(\lambda y.yz) \to_\beta (\lambda y.yz)(\lambda y.yz)y$
$\to_\beta (\lambda y.yz)zy \to_\beta zzy$
(cf Annexe ①)

**Def 16** (forme normale) Un terme M est sous forme normale si il ne peut plus être β-réduit.

**Ex 17** : $\lambda xy.x$ et $\lambda zy.y$ sont en forme normale
$(\lambda x.xx)(\lambda x.xx)$ n'a pas de forme normale

**Thm 18** (Church-Rosser) Si $P \to_\beta^* M$ et $P \to_\beta^* N$ alors il existe T tel que $M \to_\beta^* T$ et $N \to_\beta^* T$ (cf Annexe ② a)

**Corollaire 19** : Si P admet une forme normale, alors cette forme normale est unique (à $\equiv_\alpha$ près).

**Def 20** (β-équivalence) M et N sont β-équivalents ($M =_\beta N$) ssi il existe $M_1 \dots M_n$ avec $M_1 \equiv_\alpha M$ et $M_n \equiv_\alpha N$ et $\forall i \in 1, \dots n-1$ $M_i \to_\beta^* M_{i+1}$ ou $M_{i+1} \to_\beta^* M_i$
(c'est la clôture symétrique transitive de $\to_\beta$)

**Thm 21** (Church Rosser) Si $M =_\beta N$, alors il existe T tel que $M \to_\beta^* T$ et $N \to_\beta^* T$ (cf Annexe ② b)

**Ex 22** : $(\lambda xyz.xzy)(\lambda xy.x) =_\beta (\lambda xy.x)(\lambda x.x)$

$\downarrow_\beta$ $\qquad$ $\downarrow_\beta$

$\lambda yz.[(\lambda xy.x)zy]$ $\qquad$ $\lambda yx.x$

$\downarrow_\beta$

$\lambda yz.[\lambda y.z] \to_\beta \lambda yz.z$ et $\lambda yz.z \equiv_\alpha \lambda yx.x$

**Def 23** : On dit que M est un combinateur de point fixe si M est clos et pour tout terme clos F :
$MF =_\beta F(MF)$

**Ex 24** : $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ est un combinateur de point fixe
$A = \lambda xy.y(xxy)$ $\Theta = AA$ est un combinateur de point fixe (Turing)
(cf Annexe ③)

## III Représentation de données

### A - Booléens :

$T := \lambda xy.x$ (Vrai) $\qquad$ $F := \lambda xy.y$ (Faux)

si B alors M sinon N $:= BMN$ (Conditionnelle)

### B - Paires et listes :

$\langle M,N \rangle := \lambda x.xMN$ (paire) (liste)

$[M_1, \dots, M_n] := \langle M_1, \langle M_2, \langle \dots \langle M_{n-1}, M_n \rangle \dots \rangle \rangle \rangle$

$hd := \lambda x.xT$ $\qquad$ $tl := \lambda x.xF$

$\langle M_1, \dots, M_n \rangle := \lambda x.xM_1 \dots M_n$ (tuple)

$\Pi_i^n := \lambda x.xU_i^n$ où $U_i^n = \lambda x_1 \dots x_n.x_i$ (projection)

### C - Entiers de Church

(entiers) $\to [\![n]\!]_C := \lambda sz.s^n z$ où $s^n z = \underbrace{s(s \dots (sz) \dots)}_{n \text{ fois}}$

(successeur) $\to [\![succ]\!]_C := \lambda nsz.ns(sz)$

(is-zero) $\to [\![is\text{-}zero]\!]_C := \lambda n.n(\lambda x.F)T$

(addition) $\to [\![+]\!]_C := \lambda n_1 n_2.n_1 [\![succ]\!]_C n_2$

### D - Entiers de Barendregt

(zero) $\to [\![0]\!]_B := \lambda x.x$

(entiers) $\to [\![n+1]\!]_B := \langle F, [\![n]\!]_B \rangle$

(successeur) $\to [\![succ]\!]_B := \lambda x.\langle F, x \rangle$

(predécesseur) $\to [\![pred]\!]_B := \lambda x.xF$

(is-zero) $\to [\![is\text{-}zero]\!]_B := \lambda x.xT$

# IV. Modèle de calcul

**Def 25 :** Une fonction $f : \mathbb{N}^p \to \mathbb{N}$ est dite $\lambda$-définissable s'il existe $F$ un terme clos tel que
$\forall n_1, \dots, n_p \in \mathbb{N} \quad [\![ f(n_1, \dots, n_p) ]\!]_\beta =_\beta F\, [\![n_1]\!]_\beta \dots [\![n_p]\!]_\beta$
On dit que $F$ représente $f$.

**Thèse de Church 27 :** Les fonctions calculables par une procédure effective sont exactement les fonctions $\lambda$-définissables.

## 1) Lien avec les fonctions $\mu$-récursives

**Def 28 :** Les fonctions primitives récursives sont définies par induction
- zero, succ, projections sont primitives récursives
- la composition de f.p.r est f.p.r
- si $g$ et $h$ sont des f.p.r alors le schéma de récursion $f(\vec{x}, k) = \begin{cases} h(\vec{x}) \text{ si } k = 0 \\ g(\vec{x}, k, f(\vec{x}, k-1)) \text{ sinon} \end{cases}$ est une fonction primitive récursive.

**Def 29 :** On appelle minimisation non bornée la fonction $\mu_i [\![F]\!]$ qui à $\vec{x}$ associe le plus petit indice $i$ tel que $F(\vec{x}, i) \neq 0$

**Def 30 :** Les fonctions $\mu$-récursives sont le plus petit ensemble contenant les fonctions primitives récursives et clos par minimisation non bornée.

**Thm 31 :** Les fonctions $\mu$-récursives sont $\lambda$-définissables

DEV 1

**Thm 32 :** Les fonctions $\lambda$-définissables sont $\mu$-récursives.

**Def 33 (Séparabilité)** $A, B \in \mathcal{P}(\mathbb{N})$ sont récursivement séparables s'il existe une fonction récursive totale $\varphi$ telle que $n \in A \Rightarrow \varphi(n) = 0$
$n \in B \Rightarrow \varphi(n) = 1$

**Thm 34 (Scott Curry)** Aucune paire d'ensembles non vides de termes clos par $\beta$-équivalence ne sont récursivement séparables.

DEV 2

**Corollaire 35 :** Le problème
{ entrée : un terme $T$ du $\lambda$-calcul
{ sortie : oui si $T$ a une forme normale, non sinon
est indécidable.

## 2) Lien avec les machines de Turing

**Thm 36 :** Les fonctions $\mu$-récursives sont exactement les fonctions calculables par une machine de Turing.

**Corollaire 37 :** Le $\lambda$-calcul et les machines de Turing sont des modèles de calcul équivalents


**Références :** Barendregt, The Lambda Calculus its syntax and semantics

Hindley & Seldin, Lambda Calculus and combinators

Lassaigne & de Rougemont, Logique et fondements de l'informatique

## Annexes :

### ① β-réduction

$(\lambda xyz.\, xz(yz))((\lambda xy.\, yx)u)((\lambda xy.\, yx)v)\,w$

$\to_\beta (\lambda xyz.\, xz(yz))(\lambda y.\, yu)(\lambda y.\, yv)\,w$

$\to_\beta (\lambda yz.\, (\lambda y.\, yu)z(yz))(\lambda y.\, yv)\,w$

$\to_\beta (\lambda yz.\, zu(yz))(\lambda y.\, yv)\,w$

$\to_\beta (\lambda z.\, zu((\lambda y.\, yv)z))\,w$

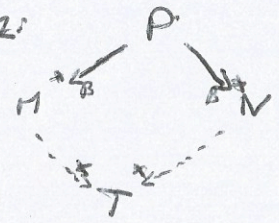
$\to_\beta (\lambda z.\, zu(zv))\,w$

$\to_\beta wu(wv)$

### ② Propriété de Church-Rosser

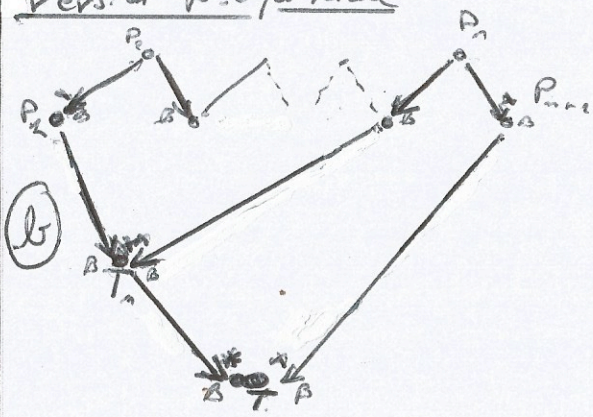
Version β-réduction :

(a)




Version β-équivalence :

(b)




### ③ Combinateurs de point fixe.

$YF \to_\beta (\lambda x.\, F(xx))(\lambda x.\, F(xx))$

$\to_\beta^+ F((\lambda x.\, F(xx))(\lambda x.\, F(xx)))$

$_\beta\!\leftarrow F(YF)$

$\Theta F \to_\beta AAF$

$\to_\beta F(AAF) = F(\Theta F)$.