Lambda Calcul

Ocan Sankur

January 20, 2008

1 Introduction

Le λ-calcul est un modèle de calcul introduit par Church et Kleene dans les années 1930, afin d'étudier les propriétés générales de certaines fonctions. Le λ-calcul peut être vu comme le plus simple des langages de programmation. Il est constitué d'une seule règle de transformation (celle de la substitution) et d'une unique façon de définir des fonctions.

Dans ce modèle, les fonctions sont définies par une suite de symboles qui constituent les λ-expressions et non pas par des ensembles de paires comme dans la théorie des ensembles. La définition de fonction du λ-calcul permet mieux de voir ce que calcule une fonction et comment elle le calcule.

Le λ-calcul a inspiré certains langages de programmation comme LISP, ses relatifs et d'autres langages fonctionnels.

1.1 Définitions et Règles de calcul

Une λ-expression est soit une variable, soit une fonction, soit l'application d'une λ-expression à une fonction. Formellement les λ-expressions sont définies inductivement comme suit.

Définition 1 (λ-expression)**.**

$$\begin{aligned} <\lambda - expression> &:= <variable> \mid <application> \mid <fonction> \\ <fonction> &:= \lambda <variable> . <\lambda - expression> \\ <application> &:= (<\lambda - expression>)<\lambda - expression> \end{aligned}$$

Ainsi dans une définition de fonction, la variable devant λ devient le paramètre qu'on utilise dans l'expression qui suit. Le syntaxe nous permet d'écrire les λ-expressions suivantes, données avec leur équivalent en Caml et en Lisp.

Exemple 1.

λ-expression	***Caml***	***Lisp***
$\lambda x.x$	*(fun x -> x)*	*(lambda (x) (x))*
$\lambda x.(f)x$	*(fun x -> f x)*	*(lambda(x) (f x)*
$\lambda f.\lambda g\lambda x(f)(g)x$	*(fun f g x -> (f (g x)))*	*(lambda (f g x) (f (g x))*

Définition 2. *L'ensemble $V(E)$ des variables d'une λ-expression E est l'ensemble de tout symbole de variable qui apparaît dans cette expression.*

1. $V(x) = \{x\}$ *pour une variable* x
2. $V(\lambda x.P) = V(P) \cup \{x\}$
3. $V((P)Q) = V(P) \cup V(Q)$

Définition 3. *Une occurence d'une variable x est dite liée si elle apparaît dans une sous-expression de la forme $\lambda x.P$. Sinon on dit qu'elle est libre.*

Définition 4. *L'ensemble $F(E)$ des variables libres de l'expression E est définie par induction comme suit:*

1. $F(x) = x$ *pour une variable* x
2. $F(\lambda x.P) = F(P) - \{x\}$
3. $F((P)Q) = F(P) \cup F(Q)$

Remarque 1.

1. *Une variable peut avoir des occurences libres et liées dans une même expression. Par exemple dans $(x)\lambda x.x$, la première occurence de x est libre, alors que les deux autres sont liées.*
2. *Une variable liée dans une λ-expression peut être libre pour une sous-expression. Par exemple, la variable x est liée dans $\lambda x.x$ mais libre dans sa sous-expression x.*

Conformément au sens commun, on considère que deux fonctions sont égales si elles ne diffèrent que des noms de leurs paramètres. Ainsi on va définir une relation d'équivalence qui regroupe les λ-expressions qui ne diffèrent que des noms de leurs variables liées. Pour cela, on aura besoin de renommer les variables d'une expression. Il s'agit de substituer toute occurence d'une variable par une autre.

Définition 5. *S'il s'agit de renommmer x par z, la nouvelle expression est obtenue comme suit:*

1. $\{z/x\}x := z$
2. $\{z/x\}y := y$ *si* $x \neq y$
3. $\{z/x\}\lambda x.E := \lambda z.\{z/x\}E$
4. $\{z/x\}\lambda y.E := \lambda y.\{z/x\}E$ *si* $x \neq y$
5. $\{z/x\}\lambda(E_1)E_2 := (\{z/x\}E_1)\{z/x\}E_2$

On peut maintenant définir la α-conversion.

Définition 6. *Pour une expression de la forme $\lambda x.P$, on définit la $\alpha-$conversion pour une variable $z \notin V(P)$ par:*

$$\lambda x.P \rightarrow_\alpha \lambda z.\{z/x\}P$$

On note que cette règle ne fait pas partie du langage du λ-calcul. C'est uniquement une façon d'obtenir une nouvelle expression à partir d'une autre. Cette transformation définit une relation d'équivalence.

Définition 7. *On note $E_1 \equiv_\alpha E_2$ si*

1. $E_1 = E_2 = x$ *pour une variable* x
2. $E_1 = (M\ N),\ E_2 = (M'\ N')$ *et* $M \equiv_\alpha M'$ *et* $N \equiv_\alpha N'$
3. $E_1 = (\lambda x.P),\ E_2 = (\lambda y.Q)$ *et* $P \equiv_\alpha \{y/x\}Q$

Définition 8. *Étant donné deux λ-expressions P, Q et une variable x, la substitution de Q aux occurences libres de la variable x dans P, noté $[Q/x]P$, est définie inductivement comme suit:*

1. $[Q/x]x \equiv Q$
2. $[Q/x]y \equiv y$ *si* $x \neq y$
3. $[Q/x]\lambda x.E \equiv \lambda x.E$
4. $[Q/x]\lambda y.E \equiv \lambda y.[Q/x]E$ *si* $x \neq y$ *et si on a* $x \notin F(E)$ *ou* $y \notin F(Q)$
5. $[Q/x]\lambda y.E \equiv \lambda z.[Q/x]z/yE$ *pour un* $z \notin V(E) \cup V(Q)$ *si* $x \neq y$, $x \in F(E)$ *et* $y \in F(Q)$
6. $[Q/x](E_1)E_2 \equiv ([Q/x]E_1)[Q/x]E_2$

On note que dans le cas 5, on remplace la variable liée y par z avant d'appliquer la substitution, afin d'éviter la capture de celle-ci qui a des occurences libres dans Q.

Exemple 2.
Considérons la substitution

$$[\lambda z.\lambda y.(y)z/x](\lambda u.(x)u)x$$

En appliquant le cas 6, on obtient,

$$([\lambda z.\lambda y.(y)z/x]\lambda u.(x)u)([\lambda z.\lambda y.(y)z/x](u)x)$$

et par les cas 4 et 6,

$$\begin{aligned}
&(\lambda u.[\lambda z.\lambda y.(y)z/x](x)u)([\lambda z.\lambda y.(y)z/x]u)([\lambda z.\lambda y.(y)z/x]x)\\
=&(\lambda u.([\lambda z.\lambda y.(y)z/x]x)[\lambda z.\lambda y.(y)z/x]u)(u)\lambda z.\lambda y.(y)z\\
=&(\lambda u.(\lambda z.\lambda y.(y)z)u)(u)\lambda z.\lambda y.(y)z
\end{aligned}$$

La règle de substitution est exprimée dans le λ-calcul par la forme $(\lambda x.P)Q$ qu'on appelle $\beta - redex$. On munit les λ-expressions d'une sémantique: on décide qu'on peut faire une substitution de x par Q dans P dès qu'on voit un $\beta - redex$ $(\lambda x.P)Q$. On appelle cela la règle de β.

Définition 9 (La règle de β)**.** *On note $M \to_\beta N$ si N est obtenue en réduisant un β-redex dans M:*

$$(\beta) \qquad (\lambda x.P)Q \to_\beta [Q/x]P$$

L'expression obtenue à l'application de cette règle s'appelle le contractum.

Même si on n'a pas encore donné un sens aux λ-expressions, on voit bien que l'application de la règle de β ne change pas la "sémantique" d'une fonction. On aimerait construire une classe d'équivalence où chaque classe est stable par l'application de la règle de β. Pourtant, contrairement à la règle de α, la règle de β n'est pas symétrique. On va donc définir une relation d'équivalence en considérant la reflexivité, et la clôture symétrique et transitive de la règle de β. On définit d'abord la β-réduction, qui est une relation d'ordre.

Définition 10. *La relation $\Rightarrow_\beta$ appelée β-réduction est défini comme suit:*

1. *$M \Rightarrow_\beta N$ si $M \equiv_\alpha N$*
2. *$M \Rightarrow_\beta N$ si $M \to_\beta N$*
3. *$(M)E \Rightarrow_\beta (N)E$ et $(E)M \Rightarrow_\beta (E)N$ si $M \Rightarrow_\beta N$*
4. *$\lambda x.M \Rightarrow_\beta \lambda x.N$ si $M \Rightarrow_\beta N$*
5. *$M \Rightarrow_\beta N$ s'il existe E tel que $M \Rightarrow_\beta E$ et $E \Rightarrow_\beta N$.*

Ainsi, le cas (1) exprime la reflexivité. Les cas (2), (3), (4) servent à étendre la définition par induction et le cas (5) exprime la transitivité. Enfin, on définit la β-équivalence qui introduit la symétrie.

Définition 11. *La relation β-équivalence est définie comme suit*

1. *$M \equiv_\beta N$ si $M \Rightarrow_\beta N$ ou $N \Rightarrow_\beta M$*
2. *$(M)P \equiv_\beta (N)Q$ si $M \equiv_\beta N$ et $P \equiv_\beta Q$*
3. *$\lambda x.M \equiv_\beta \lambda x.N$ si $M \equiv_\beta N$*
4. *$M \equiv_\beta N$ s'il existe E tel que $M \equiv_\beta E$ et $E \equiv_\beta N$.*

Définition 12. *On dit qu'une expression est sous la **forme normale** si elle ne contient aucun $\beta - redex$. Si pour une expression E, il existe N sous la forme normale, tel qu'on ait $E \equiv_\beta N$, on dit que N est la forme normale de E.*

Intuitivement la forme normale, est une expression β-équivalente qui ne peut pas être réduite davantage. Il existe des λ-expressions qui n'admettent pas de forme normale.
Considérons l'expression

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

Si on note $\omega := \lambda x.(x)x$, en appliquant la régle de β

$$\begin{aligned}(\lambda x.(x)x)\omega &= (w)w \\ &= (\lambda x.(x)x)\lambda x.(x)x\end{aligned}$$

Ainsi, on retrouve l'expression de départ: On ne peut pas "réduire" cette expression.

Une forme normale s'interprête comme un résultat de calcul. L'intêret de définir la relation d'équivalence précédente est de rassembler dans une classe toutes les λ-expressions ayant le même "résultat". Ce "résultat" n'est pas unique *a priori*. Mais on montre dans la section suivante qu'il l'est.

2 Théorème de Church-Rosser

Le théorème de Church-Rosser affirme que, sous réserve d'existence, la forme normale d'une λ-expression est unique à α-conversion près. Une autre conséquence est que la forme normale peut être obtenue en un nombre fini de réductions, à partir de n'importe quelle λ-expression d'une classe de β-équivalence. On commence par énoncer d'abord les deux lemmes suivants.

Lemme 1 (Lemme de la bande)**.** *Si on a $M \rightarrow_\beta M'$ et $M \Rightarrow_\beta N$, alors il existe N' tel que $M' \Rightarrow_\beta N'$ et $N \Rightarrow_\beta N'$.*

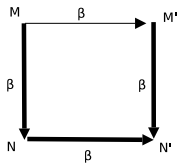

Figure 1: Le lemme de la bande

Le lemme du diamand suivant est une conséquence du lemme de la bande.

Lemme 2 (Lemme du diamand)**.** *Si $M \Rightarrow_\beta M'$ et $M \Rightarrow N$, alors il existe N' tel que $M' \Rightarrow_\beta N'$ et $N \Rightarrow_\beta N'$.*

Preuve. Il suffit d'itérer le lemme précedent autant de fois que du nombre dérivation de la réduction $M \Rightarrow_\beta M'$.
Soient $M = M_1, \ldots, M_n = M'$ les expressions obtenues étapes par étapes au

cours de la réduction de M en M'. On définit la suite N_k par récurrence de la façon suivante: Pour $k = 1$ on pose $N_1 = N$. Pour $k \geq 2$, N_{k-1} étant déjà défini, on applique le lemme de la bande à M_{k-1}, M_k et N_{k-1}. Comme on a $M_{k-1} \Rightarrow_\beta N_{k-1}$ et $M_{k-1} \rightarrow_\beta M_k$, on pose N_k l'expression que le lemme nous fournit.
L'éxpression N_n ainsi construite convient.

Figure 2: Le lemme du diamand

□

Muni du lemme du diamand, on peut démontrer le théorème de Church-Rosser.

Théorème 1. *Si M et N sont deux λ-expressions β-équivalentes, alors il existe une λ-expression Z telle que $M \Rightarrow_\beta Z$ et $N \Rightarrow_\beta Z$.*

Preuve. On fait une récurrence sur le nombre de β-réductions qui prouvent $M \equiv_\beta N$.
Si $M \Rightarrow_\beta N$, On peut prendre $Z = N$.
Sinon, il existe L différent de M et de N tel que $M \equiv_\beta L$ et $L \equiv_\beta N$. Par hypothèse de récurrence, il existe Z_1 et Z_2 tels que $M \Rightarrow_\beta Z_1 \Leftarrow_\beta L \Rightarrow_\beta Z_2 \Leftarrow_\beta N$. En appliquant le lemme du diamand à L on trouve une expression Z qui convient (voir la figure 3). □

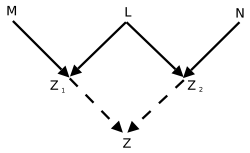

Figure 3: Théorème de Church-Rosser

Il reste à démontrer le lemme de la bande. Pour cela, on étend le langage du $\lambda - calcul$ en introduisant le symbole $\underline{\lambda}$ dont on va se servir pour marquer certains β-redex. Donc on ajoute à la définition 1 l'axiome suivante:

$$< fonction > := \underline{\lambda} < variable > . < \lambda - expression >$$

On appelle les formules de ce nouveau langage $\underline{\lambda}$-expressions.

Notation 1. *Pour deux expressions M et N, on continue de noter $M \Rightarrow_\beta N$ si M se transforme en N uniquement par des réductions des β-redex de la forme $(\lambda x.P)Q$.*
On note $M \Rightarrow_{\underline{\beta}} N$ si M se transforme en N par des réductions de $(\lambda x.P)Q$ ou $(\underline{\lambda} x.P)Q$.

Définition 13. *On définit deux fonctions $|\cdot|$ et ϕ de l'ensemble des $\underline{\lambda}$-expressions dans l'ensemble des λ-expressions de la façon suivante.*

- *Pour une $\underline{\lambda}$-expression M, $|M|$ est l'expression obtenue en remplaçant les $\underline{\lambda}$ par des λ.*
- *Pour une $\underline{\lambda}$-expression M, $\phi(M)$ est l'expression obtenue en β-réduisant les β-redex 'marqués' i.e. de la forme $(\underline{\lambda} x.P)Q$. On définit ϕ formellement par induction: Pour une variable x et les expressions M et N,*
  $\phi(x) = x$
  $\phi((M)N) = (\phi(M))\phi(N)$
  $\phi((\lambda x.M)N) = (\lambda x.\phi(M))\phi(N)$
  $\phi((\underline{\lambda} x.M)N) = [\phi(N)/x]\phi(M)$.

Notation 2. *Si $|M| = N$ ou $\phi(M) = N$, on note $M \Rightarrow_{||} N$ ou $M \Rightarrow_\phi N$ respectivement.*

On aura besoin du lemme suivant.

Lemme 3. *Si $x \neq y$ et $y \notin F(P)$, alors*

$$[P/x][Q/y]M = [[P/x]Q/y][P/x]M.$$

Preuve. On fait une induction sur la structure de M. □

Lemme 4. *Soient M et N deux $\underline{\lambda}$-expressions. Alors $\phi([N/x]M) = [\phi(N)/x]\phi(M)$.*

Preuve. On fait une induction sur la structure de M.
Si $M = x$ ou $M = y$ avec $y \neq x$ c'est trivial.
Si M s'écrit $M = (E_1)E_2$. On a

$$\begin{aligned}
\phi([N/x]M) &= \phi(([N/x](E_1))[N/x]E_2) \\
&= (\phi([N/x]E_1))[\phi(N)/x]\phi(E_2) \qquad \text{par hypothèse d'induction} \\
&= [\phi(N)/x](E_1)E_2 \\
&= [\phi(N)/x]M
\end{aligned}$$

Si $M = (\underline{\lambda} z.P)Q$, quitte à renommer z, on peut supposer $z \neq x$. On a

$$\begin{aligned}
\phi([N/x](\underline{\lambda} z.P)Q) &= \phi((\underline{\lambda} x.[N/x]P)[N/x]Q) \\
&= [\phi([N/x]Q)/z]\phi([N/x]P) \\
&= [[\phi(N)/x]\phi(Q)/z][\phi(N)/x]\phi(P) \qquad \text{par induction} \\
&= [\phi(N)/x][\phi(Q)/z]\phi(P) \text{en appliquant le lemme 3} \\
&= [\phi(N)/x]\phi((\underline{\lambda} z.P)Q)
\end{aligned}$$

Si $M = (\lambda z.P)Q$ alors, $\phi(M) = (\lambda z.\phi(P)\phi(Q))$ et on conclut par induction. □

La proposition suivante exprime le fait que si $M \Rightarrow_{\underline{\beta}} N$ alors on a une certaine liberté de l'ordre dans lequel on fait les réductions $\underline{\lambda}$et λ. En fait, en réduisant d'abord tous les $\underline{\lambda}$, en suite un certain nombre de λ, on peut obtenir l'expression $\phi(N)$.

Lemme 5. *Si $M \Rightarrow_{\underline{\beta}} N$, alors $\phi(M) \Rightarrow_\beta \phi(N)$.*

Preuve. On fait une récurrence sur le nombre de réduction.
Cas 1(a). La réduction est simplement $(\lambda x.P)Q \rightarrow_\beta [Q/x]P$

$$\begin{aligned}\phi((\lambda x.P)Q) &= \\ &= (\lambda x.\phi(P))\phi(Q) \\ &\rightarrow_\beta [\phi(Q)/x] \\ &= \phi([Q/x]P) \qquad \text{par le lemme précedant}\end{aligned}$$

Cas 1(b). La réduction est simplement $(\underline{\lambda} x.P)Q \rightarrow_{\underline{\beta}} [Q/x]P$

$$\begin{aligned}\phi((\underline{\lambda} x.P)Q) &= \\ &= [\phi(Q)/x]\phi(P) \\ &= \phi([Q/x]P) \qquad \text{par le lemme précédant}\end{aligned}$$

Cas 2(a). $M \Rightarrow_{\underline{\beta}} N$ s'écrit $(Z)P \Rightarrow_{\underline{\beta}} (Z)P'$ et se déduit directement de $P \Rightarrow_{\underline{\beta}} P'$, alors par induction, on a $\phi(P) \rightarrow_{\underline{\beta}} \phi(P')$ et donc $\phi(ZP) \rightarrow_\beta \phi(ZP')$.
Cas 2(b), 2(c). $M \Rightarrow_{\underline{\beta}} N$ s'écrit $(P)Z \Rightarrow_{\underline{\beta}} (P')Z$ ou $\lambda x.P \Rightarrow_{\underline{\beta}} \lambda x.P'$ et se déduit directement de $P \Rightarrow_{\underline{\beta}} P'$. Cas similaire au cas 2(a).
Cas 2(d). $M \Rightarrow_{\underline{\beta}} N$ s'écrit $(\underline{\lambda} x.P)Q \Rightarrow_{\underline{\beta}} (\underline{\lambda} x.P')Q'$ et se déduit directement de $P \rightarrow_{\underline{\beta}} P'$ et $Q \rightarrow_{\underline{\beta}} Q'$. Par induction on a $\phi(P) \Rightarrow_\beta \phi(P')$ et $\phi(Q) \Rightarrow_\beta \phi(Q')$. Alors,

$$\begin{aligned}((\underline{\lambda} x.P)Q) &= [\phi(Q)/x]\phi(P) \\ &\Rightarrow_\beta [\phi(Q')/x]\phi(P') \\ &= \phi((\underline{\lambda} x.P')Q')\end{aligned}$$

Cas 3. $M \Rightarrow_{\underline{\beta}} N$ se déduit directement de $M \Rightarrow_\beta L$ et $L \Rightarrow_\beta N$. Alors par induction, $\phi(M) \Rightarrow_\beta \phi(L) \Rightarrow_\beta \phi(N)$. □

Lemme 6. *Soit M une $\underline{\lambda}$-expression. Si $|M| = M'$ et $\phi(M) = M''$, alors $M' \Rightarrow_\beta M''$.*

Preuve. Induction sur M.
Cas 1. Si $M = x$ alors $|x| = \phi(x)$.
Cas 2. Si $M = (P)Q$ alors $|(P)Q| = (|P|)|Q| \Rightarrow_\beta \phi(P)\phi(Q)$, par induction.
Cas 3. Si $M = \lambda x.P$, alors $|\lambda x.P| = \lambda x.|P| \Rightarrow_\beta \lambda x.\phi(P)$, par induction

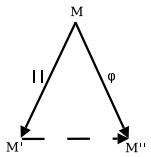

Figure 4:

Cas 4. Si $M = (\underline{\lambda}x.P)Q$, alors

$$\begin{aligned} |(\underline{\lambda}x.P)Q| &= (\lambda x.|P|)|Q| \\ &\Rightarrow_\beta (\lambda x.\phi(P))\phi(Q) \\ &\Rightarrow_\beta [\phi(Q)/x]\phi(P) \\ &= \phi([Q/xP) \end{aligned}$$

□

Lemme 7. *Soit M une $\underline{\lambda}$-expression. Si $|M| = N$ et $N \Rightarrow_\beta N'$, alors il existe une $\underline{\lambda}$-expression M' tel que $M \Rightarrow_{\underline{\beta}} M'$.*

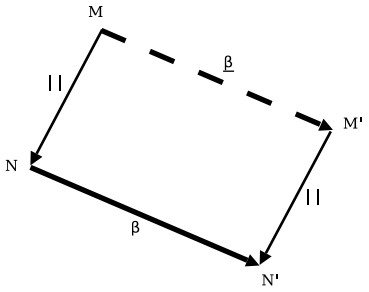

Figure 5: Lemme 7

Preuve. On fait une démonstration comme précédemment par induction, en utilisant le fait que $|[Q/x]P| = [|Q|/x]|P|$. Intuitivement, l'idée est qu'on peut appliquer à M, les même les réductions qui transforment N en N' par des $\underline{\beta}$-réductions. □

Preuve du lemme de la bande. Soit $R = (\lambda x.P)Q$ le β-redex qu'on réduit dans $M \rightarrow_\beta M'$. Notons M'' la $\underline{\lambda}$-expression obtenue en remplaçant le premier λ de M par $\underline{\lambda}$. La situation est représentée dans la figure suivante.

Par l'application du lemme 7 à M'', M et N, on a l'existence de l'expression N''; par le lemme 5 celle de N'; et par le lemme 6 on a la dernière connexion $N \Rightarrow_\beta N'$. L'expression N'' ainsi obtenue convient. □

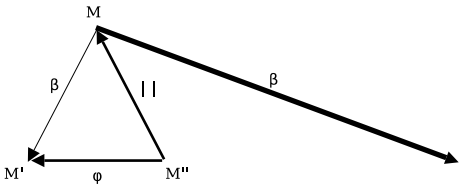

Figure 6: Situation initiale

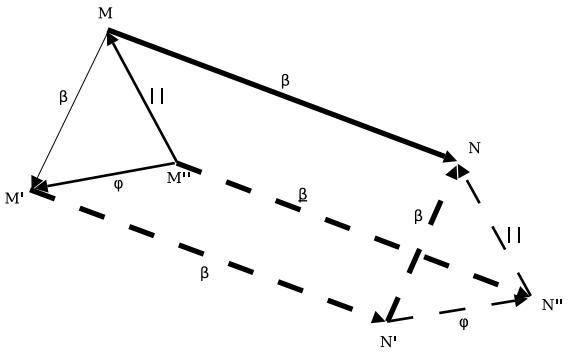

Figure 7: Le lemme de la bande

References

[1] Henk Barendregt *A global representation of the recursive functions in the λ-calculus* 1976. Theoretical Computer Science.

[2] Henk Barendregt *The Lambda Calculus. Its Syntax and Semantics* 1981. North Holland Publishing Company.

[3] György E. Révész *Lambda-Calculus, Combinators, and Functional Programming* 1988. Cambridge Tracts in Theoretical Computer Science 4.