LAMBDA-CALCUL

Le λ-calcul est un formalisme introduit par Church pour traduire l'aspect mécanique de l'évaluation dans les langages applicatifs (ou fonctionnels) notamment le traitement de la récursivité.
Au minimum, on prendra un ensemble dénombrable V de "variables" et les expressions ou termes seront les mots construits à partir des variables, les "applications" u.v où u et v sont des termes, et les "abstractions" λx u où x est une variable et u un terme. Si on rajoute un ensemble de "constantes" C, alors L est le plus petit ensemble tel que $L = V \cup C \cup LL \cup \lambda VL$.

Notations et Curryfication
Le point peut être omis ainsi que les parenthèses à condition de convenir d'une priorité à gauche, ainsi uvw signifiera (u.v).w. On note parfois aussi λ x y u pour λx (λ y u)) s'il est clair que x y sont des variables. Le λ-calcul réalise une modèlisation des fonctions de plusieurs variables dans la mesure où une fonction x, $y \rightarrow u$ est "curryfiée" en $x \rightarrow (y \rightarrow u)$ c'est à dire une fonction de l'unique variable x dont le résultat est une fonction de l'unique variable y dont le résultat est u.

Variables libres
Les variables libres d'une expression sont définies par :
si $x \in V$ alors libres(x) = {x}, libres(λx u) = libres(u) - {x}, libres(uv) = libres(u) ∪ libres(v)
Une variable ayant une occurrence non libre dans une expression est dite liée dans cette expression.
Une expression est dite close si elle n'a pas de varible libre.
Longueur, si $x \in V$ alors long(x) = 1, long(uv) = long(u) + long(v), long(λx u) = 1 + long(u)
Règle de substitution
Le terme u[x<-t] nommé "u dans lequel la variable x est remplacée par le terme t" est défini par :
Si u est la variable x alors t
Si u est une variable différente de x alors u
Si u est l'application vw alors c'est (v[x<-t])(w[x<-t])
Si u est l'abstraction λy v et $x \notin$ libres(u) alors λy v
Si u = λy v et $y \notin$ libres(t) alors λy (v[x<-t])
Si u = λy v et $y \in$ libres(t) alors λz ((v[y<-z]) [x<-t]) où z est une variable non libre dans u ni dans t.
Lemme de substitution u[x<-v] [y<-w] = u[y<-w] [x<-v[y<-w]] évident par induction
Relations de conversion
Renommage ou alpha-conversion
C'est le dernier cas de substitution, on définit dans L la relation $\rightarrow_\alpha$ par :
Si $y \notin$ libre(u) alors λx u $\rightarrow_\alpha$ λy u[x<-y] (changement de variable liée)
On définit une α–équivalence des termes λx u $\equiv_\alpha$ λx'u' dans la mesure où u' est équivalent à u dans lequel x' est substitué à toutes les occurences de x (ils ne diffèrent que par des changements de nom des variables liées).
Beta-conversion, c'est la relation (λx u)t $\rightarrow_\beta$ u[x<-t] (application d'une abstraction sur un terme t)
Exemple :
λx (xy)[x<-y] $\rightarrow_\alpha$ λz (xz)[x<-y] $\rightarrow_\beta$ λz(yz) et (λz (zx)) (λx x)[x<-y] = (λz (zx)) (λy y) $\rightarrow_\beta$ (λy y)x $\rightarrow_\beta$ x
Eta-conversion, c'est la relation (λx u)x $\rightarrow_\eta$ x à la condition que x non libre dans dans u.
Exemple : (λx (λyλz zy)x)u (λx x) $\rightarrow_\eta$ (λyλz zy)u (λx x) $\rightarrow_\beta$ λz(zu) (λx x) $\rightarrow_\beta$ (λy y)x $\rightarrow_\beta$ (λx x) u $\rightarrow_\beta$ u
Enchaînement, on note $\rightarrow^*$ ou plus simplement $\rightarrow$ la clôture transitive de $(\rightarrow_\alpha) \cup (\rightarrow_\beta)$ et soit ≡ la clôture symétrique et transitive, Λ est l'ensemble quotient.
Expression normale ou irréductible : n'ayant pas d'image par $\rightarrow_\beta$.
Le λ-calcul n'est pas noethérien (toute chaîne de termes liés par β-réduction n'est pas nécessairement finie), démonstration en posant Δ = λx (xx) et Ω = ΔΔ alors Ω $\rightarrow_\beta$ (λx (xx))(λx (xx)) $\rightarrow_\beta$ (λx (xx))(λx (xx))(λx (xx)) $\rightarrow_\beta$ ... longueur croissante et pas de terminaison.

Théorème de Church-Rosser, la β-réduction est confluente (si une expression conduit à deux formes normales (irréductibles), elles sont α-équivalentes.

Second théorème de Church-Rosser, si u se réduit en une forme normale v, il existe une suite de réductions suivant la stratégie standard.
On définit la stratégie normale (ou standard) : réduction des termes les plus externes et les plus à gauche, ainsi :

$\lambda x\ \lambda y\ x\ ((\lambda x\ x)y) \to_\beta \lambda y\ ((\lambda x\ x)\ y)$ à gauche $\to_\beta \lambda y\ y$ à l'intérieur
et $\lambda x\ \lambda y\ x\ ((\lambda x\ x)y) \to_\beta \lambda x\ ((\lambda y\ x)\ y)$ à droite $\to_\beta \lambda x\ x$ à l'intérieur qui sont α-équivalentes

La preuve se fait par récurrence sur (prof(u), |u|).
Le passage des paramètres par valeur (ou appel par nom), c'est l'évaluation applicative qui exige que tous les paramètres soient calculés avant d'être communiqués à la fonction.

Combinateurs : expressions closes, exemples classiques :

Faux $F = \lambda x\ \lambda y\ y$ (aussi notée nil) on montre $\forall u\ \forall v$ termes clos $Fuv \to v$, $Tuv \to u$

Vrai (annulateur) $T = K = \lambda x\ \lambda y\ x$

Identité $I = \lambda x\ x$

Combinateur de Turing $\Theta = AA$ tel que $A = \lambda x\ \lambda y\ (y(xxy))$

Compositeur (composition de deux fonctions) $B = \lambda f\ \lambda g\ \lambda x\ f(gx)$

Permutateur ou Condition $C = IF = \lambda b\ \lambda x\ \lambda y\ bxy$ (bxy signifiant bien (bx)y d'où $B \neq C$)

Distributeur $S = \lambda x\ \lambda y\ \lambda z\ (xz(yz))$ engendre tous les autres à l'aide de T

Combinateur de Curry $Y = \lambda h\ ((\lambda x\ h(xx))\ (\lambda x\ h(xx)))$

Propriétés $STT \to I$
En effet $STT = \lambda x\ \lambda y\ \lambda z\ (xz(yz))\ (\lambda x\ \lambda y\ x)\ (\lambda x\ \lambda y\ x) \to_\beta \lambda y\ \lambda z\ ((\lambda x\ \lambda y\ x)z(yz))\ (\lambda x\ \lambda y\ x)$
$\to_\beta \lambda z\ ((\lambda x\ \lambda y\ x)\ z\ ((\lambda x\ \lambda y\ x)\ z)) \to_\beta \lambda z\ ((\lambda y\ z)\ (\lambda y\ z)) \to_\beta \lambda z\ z \to_\alpha I$

$S(TS)T \to B$
En effet $S(TS)T = \lambda x\ \lambda y\ \lambda z\ (xz(yz))\ (TS)\ T \to_\beta \lambda z\ ((TS)\ z\ (Tz)) = \lambda z\ (((\lambda x\ \lambda y\ x)\ S)\ z\ ((\lambda x\ \lambda y\ x)\ z))$
$\to \lambda z\ (((\lambda x\ \lambda y\ x)\ S)\ z\ ((\lambda x\ \lambda y\ x)\ z))$

$YI \to \Omega$
Car $YI = \lambda h\ ((\lambda x\ h(xx))\ (\lambda x\ h(xx)))\ I \to_\beta \lambda x\ I(xx)\ (\lambda x\ I(xx)) \to_\beta (\lambda x\ (xx))\ (\lambda x\ (xx)) = \Omega$
$\Theta u = AAu = \lambda x\ \lambda y\ (y(xxy))\ Au \to u(AAu) = u(\Theta u)$ ce qui fait que Θu est un point fixe de u.

Combinateurs de point fixe X : expression X telle que $\forall f$ on a $Xf = f(Xf)$

Θ comme Y sont des combinateurs de point fixe.
$P = YT$ est un terme "poubelle" avalant tous ses arguments $Px \to P$ car $Px = YTx \to T(YT)x \to Yx$
Théorème du point fixe : tout terme f possède un point fixe v

Posons $v = \lambda x\ f(xx)$ et $v = uu$ alors $v = (\lambda x\ f(xx))\ u \to_\beta f(uu) = fv$

Théorème de Bhöm X est un combinateur de point fixe $\Leftrightarrow X \equiv SI\ X$

En effet posons $SI = \lambda x\ \lambda y\ \lambda z\ (xz(yz))\ I \to_\beta \lambda y\ \lambda z\ (Iz(yz)) \to_\beta \lambda y\ \lambda z\ (z(yz)) \to_\alpha \lambda x\ \lambda y\ y(xy)$ et supposons $X = SIX$ alors pour tout f, $Xf = (\lambda x\ \lambda y\ y(xy))\ Xf \to f(Xf)$ c'est à dire que Xf est un point fixe de f. Réciproquement, si $\forall f$, $Xf \to f(Xf)$ alors $SIX \to \lambda y \lambda f\ f(yf))\ X \to \lambda f\ f(Xf) \equiv \lambda f\ Xf \equiv X$

Second théorème de Bhöm $Y_0 = Y$ et les $Y_{n+1} = Y_n(SI)$ sont des combinateurs de point fixe.
Y_0 en est un et par récurrence comme $Y_n = SI\ Y_n$, $Y_{n+1} = Y_n(SI) \to SI(Y_n(SI)) \to SI(Y_{n+1})$ donc Y_{n+1} est un combinateur de point fixe d'après le premier théorème.

Théorème du double point fixe $\forall F\ \forall G\ \exists A\ \exists B$ tels que $A \equiv FAB$ et $B \equiv GAB$
Prenons $A = \Theta(\lambda a\ (FaB))$ et $B = \Theta(\lambda b\ (GAb))$, ayant $\forall H\ \Theta H \to H(\Theta H)$ on a $A = \Theta(\lambda a\ (FaB))$
$\to (\lambda a\ (FaB))(\Theta(\lambda a\ (FaB))) \to (\lambda a\ (FaB))A' \to FAB$ et $B = \Theta(\lambda b\ (GAb)) \to (\lambda b\ (GAb))B \to GAB$

Théorème, l'existence d'une forme normale est indécidable.

On peut facilement numéroter les termes c'est à dire construire une fonction # de Λ dans N, et en déduire num(n) = #(n) qui sera récursive ainsi que app(#x, #y) = #(xy).

Grâce au deuxième théorème du point fixe, pour tout f, si $w = \lambda x\ f(\underline{app}\ x\ (\underline{num}\ x))$ alors $X = w(\underline{\#w})$, vérifie $f(\underline{\#(X)}) \to X$.

Soit A l'ensemble des termes ayant une forme normale, il est stable par réduction et s'il est récursif, il existe une fonction indicatrice k telle que k(#x) = si x ∈ A alors 1 sinon 0.

Soit H = λx k x T Ω I on vérifie H(#x) → si x ∈ A alors I sinon Ω, ce qui contredit l'existence du point fixe X tel que H(#X) → X.

REPRÉSENTABILITÉ EN Λ-CALCUL

On pose d'autres combinateurs :

La condition IF = λb λx λy (b x) y, on a en effet :

IF T xy →αβ λx λy ((λu λv u) x) y →β x et par ailleurs IF F xy →αβ λx λy ((λu λv v) x) y →β y

La négation ¬ = λx (IF x F T) on a ¬T → (λx (IF xFT))T → IF TFT → F et l'inverse.

Les connecteurs ET = λx λy x y F, OU = λx λy x T y, on a ET TT → λx λy x y F TT → TTF → λx λy x TF → T de même ET TF → F, ET FT → F, ET FF → F

Notation de séquence de Church [u, v] = λx xuv et si n ∈ N, u, m ∈ Λ $u^0m = m$, $u^{n+1}m = u(u^nm)$

first = λx xT et second = λx xF, alors first [u, v] → λx xT (λx xuv) → (λx xuv) T → Tuv → u et :

second[u, v] → λx xF (λx xuv) → (λx xuv) F → Fuv → v

Généralisation $\langle u_1 \dots u_n \rangle$ = λx $xu_1 \dots u_n$ alors la i-ième projection est définie par $p_{i,n}$ = λx (xλx$_1$λx$_2$ ... λx$_n$ x$_i$) et

$p_{i,n}\langle u_1 \dots u_n \rangle$ = λx (xλx$_1$λx$_2$ ... λx$_n$ x$_i$) (λx $xu_1 \dots u_n$) → ((λx $xu_1 \dots u_n$) (λx$_1$λx$_2$ ... λx$_n$ x$_i$))
→ (λx$_1$λx$_2$ ... λx$_n$ x$_i$) $u_1 \dots u_n$ → u_i

cons = λa λb λs (sab)

first (cons a b) = (λx xT) (λs (sab)) → (λs (sab))T → Tab → a

Les entiers de Church : 0 = F, 1 = λf λx (f x), 2 = λf λx (f (f x)) et n = λf λx (f^n x).

Autre définition avec 0 = I, n+1 = [F, n], suc = λx [F, x], zero = first = λx xT alors zero 0 = (λx xT) I → IT → T et zero 1 = (λx xT) 1 → [F, I] T → TFI → (λx λy x) FI → F enfin pred = second = λx (xF) = second, par exemple pred 3 = second [F, [F, [F, I]]] → [F, [F, I]] = 2.

Théorème Les fonctions λ-représentables sont exactement les fonctions récursives.

On dit alors que la fonction f de N^n dans N est représentable en λ-calcul par le terme clos F si et seulement si pour tous entiers $k_1, \dots, k_n$ si $f(k_1, \dots, k_n)$ n'est pas défini alors l'expression $F\underline{k_1}, \dots, \underline{k_n}$ n'est pas réductible sinon si $f(k_1, \dots, k_n) = k$ alors l'expression $F\ \underline{k_1}, \dots, \underline{k_n}$ se réduit en $\underline{k}$.

On peut en effet construire toutes les fonctions récursives telles que :

SUC = λn λf λx ((n f) (f x))

En particulier SUC $\underline{n}$ → λf λx (((λf λx (f^n x)) f) (f x)) → λf λx ((λx (f^n x)) (f x))
→ λf λx (f^n (f x)) = λf λx (f^{n+1} x) = $\underline{n+1}$

Somme + = λn λm λf λx ((n f) ((m f) x)) et produit * = λn λm λf λx ((n (m f)) x)

Exemple + $\underline{1}$ $\underline{2}$ = λn λm λf λx ((n f) ((m f) x)) (λf λx (f x)) (λf λx (f (f x)))
→ λf λx (((λf λx (f x)) f) ((λf λx (f (f x)) f) x)) → λf λx ((λx (f x)) (f (f x)) → λf λx (f (f (f x))) = $\underline{3}$

zero = λn λx λy n(Ty)x

pred = λn (((n λa λb b (SUC(aT)) (aT))) (λc c00) F)

expo = λn λm λf λx mnfx et fac = Y(λf λn zero n 1 (* n (f (pred n))))

Projections, il suffit de définir pr_i = λx$_1$ λx$_2$ ... λx$_p$ x$_i$

Composition, si f1, ... fp sont à q arguments et h à p arguments, la représentation de $h(f_1\ f_2 \dots f_p)$ sera $\underline{h(f_1 \dots f_p)}$ = λx$_1$ λx$_2$ ... λx$_q$ $\underline{h}$ ($\underline{f_1}$ x$_1$ x$_2$ ... x$_q$)($\underline{f_2}$ x$_1$ x$_2$ ... x$_q$) ... ($\underline{f_p}$ x$_1$ x$_2$ ... x$_q$)

Récurrence, si ϕ est définie par ϕ(0, x) = k(x) et ϕ(n+1, x) = h((n, x), n, x) où x est lui-même un vecteur, on pose = Y(λf λn λx n T($\underline{k}$ x) ($\underline{h}$ (f (n F)x) (nF) x))

Minimisation (schéma μ), si ϕ(n) = min {k / h(k) = 0}, soient P = λ y $\underline{h}$ yxT et Q = Y(λf λz (Pz) z (f (suc z))) et $\underline{\phi}$ =λx Q$\underline{0}$

Ces résultats prouvent que les fonctions λ-définissables sont closes par composition, récurrence et minimisation.

Exemples de λ-calcul appliqués

Si L = V ∪ C ∪ LL ∪ λVL avec un ensemble de constantes C
a) Si C = 0 | suc | pred | zero
En posant 1 = suc 0, 2 = suc 1, ... n+1 = suc n et grâce à des règles telles que :

pred 0 ⊢ 0, pred (suc n) ⊢ n, zero 0 ⊢ λx λy x, zero (suc n) ⊢ λx λy y
Si Add = λx λy x(zero x y (suc (Add (pred x) y))), on aura Add n m = n + m par récurrence.

b) Si C = e | a1 | a2 | a3 | .... | vide | car | cdr | cons | ap et les règles :

vide e ⊢ λx e, vide x ⊢ λy y, cons x e ⊢ x, car a1 ... an ⊢ a1, cdr a1 ... an ⊢ a2 ... an,
ap e x ⊢ x, ap (a1 ... an) b ⊢ a1 ... an b

LAMBDA-CALCUL TYPÉ

Soit $T_0 = \{\alpha, \beta, \dots\}$ un ensemble de "types simples" de base et → un opérateur binaire dit "exponentiaion", l'ensemble des types T est le plus petit ensemble contenant T_0 et stable par →.

Le λ-calcul typé est le triplet (Λ, T, t) où t est une application de Λ sur T telle que ∀u ∀v ∈ Λ t(λu v) = t(u) → t(v).et t(f) = α→β, t(x) = α entraînent t(fx) = β.

Λ est alors partitionné par les types et on note Λ^α les termes de type α, on note également x : α cette appartenance x ∈ Λ^α à un type.

Exemple de calcul de type, pour l'expression λf λg λx (f x) (g x) on pose x de type inconnu α, alors f est nécessairement de type α→β , et g : α→γ mais (f x) : β s'applique sur (g x) : γ donc β = γ→δ puis (f x) (g x) : δ, et → enfin le type de l'expression est (α→(γ→δ))→ (α→γ) → α→δ.

La β-réduction typée est (λx : α u)v →β u[x ← v] si v est du même type α.

Théorème de Church-Rosser Le λ-calcul typé est noethérien, la β-réduction typée est confluente d'où l'unicité de la forme normale et la stratégie normale l'atteint.

On peut définir la taille d'un type par 1 pour les types de base et par |α→β| = $|\beta|^{|\alpha|}$. La récurrence se fait sur (taille(t(u)), prof(u), |u|) le seul cas délicat est celui de uv avec u → λx u' et v → v' où u', v' sont normalisables.

LOGIQUE COMBINATOIRE

C'est le formalisme équivalent au λ-calcul où on part d'un ensemble dénombrable de variable v, un ensemble C de constantes et deux symboles T, S (T désignant le vrai était noté K à l'origine). Les termes de CL sont les mots sur C ∪ V ∪ {T, S}, ils sont dits clos s'ils n'ont pas d'occurrence de variables. Les "combinateurs" sont les mots formés sur T, S uniquement.
Syntaxe La théorie CL est définie par les axiomes de l'égalité et Txy = x et Spqr = pr(qr)

Théorème Si I = STT alors ∀x CL ⊢ Ix = x, en effet Ix = STTx = Tx(Tx) = x
Théorème L'ω-réduction définie par Txy →ω x et Sxyz →ω xz(yz) est confluente, la ω-forme normale est unique.

La définition des λ-abstractions peut se faire par λxx = I, λx u = Tx si x ∉ libres(u), λx uv = S(λx u) (λx v), on montre par induction que (λx u)v →ω u[x ← v].

Exemple λx λy yx = λx (λy yx) = λx (S(λy y) (λy x)) = λx (SI(Tx)) = S(λx SI) (λx Tx)
= S (T(SI)) (S (λx T) (λx x)) = S(T(SI)) (S(TT) I)

Sémantique Une algèbre combinatoire (D, ., t, s) ⇔ D ensemble de cardinal ≥ 2 où t, s ∈ D, . est une loi binaire interne telle que ∀a, b, c ∈ D tab = a et sabc = ac(bc).

Une interprétation de CL dans D est une "valuation" ρ : V → D étendue par ρ(T) = t, ρ(S) = s, ρ(xy) = ρ(x).ρ(y) alors ρ(u[x ← v]) = (ρ + (ρ(x) = ρ(v)) (u) où + est la superposition de fonctions.

Théorème de complétude (D modèle de formule Φ) D ⊨ Φ ⇔ ∀ρ (D, ρ) ⊨Φ ⇔ CL ⊢ Φ

POINT DE VUE ALGEBRIQUE, SYSTEMES DE REDUCTION

Soit G = (E, R) un graphe où la relation R est notée $\rightarrow$ et $\rightarrow^*$ désigne la clôture réflexive et transitive de $\rightarrow$, alors que $\equiv$ désigne la clôture équivalence. On pose les définitions suivantes :

a irréductible (normal) $\Leftrightarrow \forall b \ \neg(a \rightarrow b)$

R localement confluente en a $\Leftrightarrow \forall b, c \ \exists d \ a \rightarrow b \wedge a \rightarrow c \Rightarrow b \rightarrow^* d \wedge c \rightarrow^* d$

R confluente en a $\Leftrightarrow \forall b, c \ \exists d \ a \rightarrow^* b \wedge a \rightarrow^* c \Rightarrow b \rightarrow^* d \wedge c \rightarrow^* d$

R fortement confluente en a $\Leftrightarrow \forall b, c \ \exists d \ a \rightarrow b \wedge a \rightarrow c \Rightarrow b \rightarrow d \wedge c \rightarrow d$

R confluente $\Leftrightarrow \forall a$, R confluente en a $\Leftrightarrow$ R* fortement confluente partout

R noethérien $\Leftrightarrow \forall a$, toute suite $a \rightarrow a_1 \rightarrow a_2 \rightarrow ... \rightarrow a_n$ est finie $\Leftrightarrow \forall a$, a possède une forme normale

Théorèmes 1) Si R est confluente $\forall a, b \ \exists c \ a \equiv b \Rightarrow a \rightarrow^* c \wedge b \rightarrow^* c$

2) R est confluente $\Rightarrow \forall a$, si a possède une forme normale, celle-ci est unique.

3) R est noethérien et localement confluente $\Rightarrow$ R confluente.

4) Théorème de Hindsey-Rosen Si r, S fortement confluentes commutent ($\forall a, b, c \ \exists d$, a R b $\wedge$ a S c $\Rightarrow$ b S d $\wedge$ c R d) alors R $\cup$ S est confluente.

> 1) Evident par récurrence sur la longueur de la chaîne de a vers b, 2) si a_1, a_2 sont deux formes normales de a, la confluence impose un d tel que $a_1 \equiv d \equiv a_2$ 3) évident à cause de l'existence de formes normales 4) évident

Treillis complet (D, $\leq$) ensemble muni d'un ordre où toute partie possède une borne supérieure. On a alors deux éléments F = min($\varnothing$) et T = sup(D)

A partie directe de D $\Leftrightarrow$ A partie non vide de D et $\forall a, b \in A \ \exists m \in A$, m majorant de a et de b.

f application continue entre deux treillis complets D et D' $\Leftrightarrow \forall A$ directe de D f(supA) = sup f(A)

> Si f est continue, elle est donc monotone, par ailleurs D*D' est un treillis complet et l'ensemble C(D, D') des fonctions continues en est un également avec $\forall x \ (\sup\{f_i\})(x) = \sup \{f_i(x) / \ x \in D\}$

Théorème du point fixe Toute fonction continue entre deux treillis complets, admet un plus petit point fixe.

> A = $\{f^n(F) / \ n \in N\}$ direct et $f^n(F) \sqsubseteq f^{n+1}(F)$ d'où en posant fix(f) = sup(A) on a f(fix(f)) = sup $f(f^n(F))$ = fix(F)

Références

Alliot J.M. Schiex T. Intelligence artificielle et informatique théorique, Cepaduès 1994
Glaser H. Hankin C. Till D. Principes de programmation fonctionnelle, Masson 1987
Barendregt H.P. The lambda calculus, Its syntax and semantics, North Holland 1981
Krivine J.L. Lambda-calcul, Types et modèles, Masson 1990