Confluence du λ-calcul non-typé

Adrien Surée

Décembre 2010

Le λ-calcul est un langage de programmation théorique conçu dans les années 30 par Alonzo Church. Il est utilisé principalement comme cadre d'étude des propriétés des langages de programmation fonctionnels. C'est aussi un outil de la théorie de la calculabilité en général et il trouve aussi des applications dans la démonstration formelle.

Notations On désignera le n-uplet x_0, x_1, ..., x_n par $\bar{x}$. On notera $\bar{x} := \bar{y}$ pour $x_0 := y_0$, $x_1 := y_1$, ..., $x_n := y_n$.

Table des matières

1 Présentation

1.1 Syntaxe

On se donne ici un ensemble infini de symboles V. On appellera ses éléments des *variables*.

Définition 1. On définit les λ-termes par induction :
- toute variable est un λ-terme ;
- si t et t' sont des λ-terme, alors tt' est un λ-terme ;
- si x est une variable et t un λ-terme, alors $\lambda x.t$ est un λ-terme.

Intuitivement, on peut voir tt' comme l'application de t à t' et $\lambda x.t$ comme la fonction qui à x associe t.

Notation 2. On note $\lambda x_1 \dots x_n.t$ le terme $\lambda x_1.\lambda x_2.\dots.\lambda x_n.t$. Fonctionnellement parlant, on considère ce terme comme ayant n arguments. On dit que les λ-termes sont curryfiés.

1.2 Un modèle de calcul

Nous avons défini la syntaxe du λ-calcul et donné une idée informelle de sa sémantique. On peut donc se demander qu'est-ce que ce modèle permet de calculer. À premier abord, on est difficilement convaincu de l'expressivité du modèle. Par exemple, les constantes n'existent pas en λ-calcul. Il est en fait possible d'encoder toutes les structures de données qui nous intéressent en λ-calcul. On donnera ici l'exemple des entiers dont la représentation en λ-calcul porte le nom d'entiers de Church. Ils correspondent au n-ième itéré de leur argument.

Exemple 3. Pour tout entier n, on définit par récurrence le n-ième entier de Church noté $\bar{n}$ comme ceci :
- $\overline{0} = \lambda x.x$;
- $\overline{n+1} = \lambda fx.\overline{n}fx$.

On peut aussi montrer (mais ce n'est pas notre propos ici) que le λ-calcul est équivalent en termes de calculabilité aux machines de Turing. Cela rentre dans le cadre de la thèse de Church qui affirme que tous les définitions raisonnables de la calculabilité sont équivalentes.

1.3 Variables libres et liées

Définition 4. On appelle $S(t)$ l'ensemble des sous-termes d'un terme t. On le définit par induction sur la structure de t :
- si $t = v \in V$, $S(t) = v$;
- si $t = \lambda x.t'$, $S(t) = \{t\} \cup S(t')$;
- enfin si $t = uv$, alors $S(t) = \{t\} \cup S(u) \cup S(v)$.

Définition 5. On dit qu'une occurrence d'une variable v est liée dans t si il existe un sous-terme de t qui contient cette occurrence de v de la forme $\lambda v.t$. Si une occurrence d'une variable v n'est pas liée dans t alors on dit qu'elle est libre dans t.

v est liée dans t si elle admet au moins une occurrence liée dans t. De même elle est libre si au moins une occurrence l'est.

Attention, $\lambda x.y$ ne contient pas d'occurrence de x !

Exemple 6. x est libre et liée dans $(\lambda x.x)x$.

Définition 7. On note $Vlib(t)$ l'ensemble des variables libres de t.

Définition 8. On appelle L l'ensemble des λ-termes et L^0 l'ensemble des λ-termes t tels que $Vlib(t) = \emptyset$.

2 Substitutions et α-équivalence

On développe dans cette partie deux mécanismes très importants : l'α-équivalence et la substitution. Le premier va nous servir à reconnaître les termes qui sont égaux à un "changement de variables" près. Le deuxième est un mécanisme général qui trouvera son utilité dans la définition et l'étude des réductions, les étapes de calcul sur les termes.

Le problème majeur que l'on rencontre dans la définition de la substitution est celui de la "capture de variables". C'est pour cela qu'on donne d'abord un mécanisme simple de subtitution puis un plus sophistiqué.

2.1 Substitution simple

Informellement, la substitution simple consiste à remplacer dans un terme les occurrences libres d'une variable par un terme fixé.

Exemple 9. 1. $(\lambda x.x) < x := t >= \lambda x.x$

2. $(\lambda x.xy) < y := (\lambda x.x) >= (\lambda x.x(\lambda x.x))$

Définition 10. On définit la substitution par induction sur t :
- si $u = x_i \in \bar{x}$ alors $u < \bar{x} := \bar{v} >= v_i$;
- si $u = y \notin \bar{x}$ alors $u < \bar{x} := \bar{v} >= y$;
- si $u = \lambda x_i.u'$ avec $x_i \in \bar{x}$ alors $u < \bar{x} := \bar{v} >= \lambda x_i.u' < \bar{x} - \{x_i\} := \bar{v} - \{v_i\} >$;
- si $u = \lambda y.u'$ avec $y \notin \bar{x}$ alors ; $u < \bar{x} := \bar{v} >= \lambda y.u' < \bar{x} := \bar{v} >$;
- si $u = vw$ alors $u < \bar{x} := \bar{v} >= (v < \bar{x} := \bar{v} >)(w < \bar{x} := \bar{v} >)$.

Les deux lemmes suivants se montrent par de simples inductions sur la structure de u.

Lemme 11. (substitutions simples successives) *Pour tout $\bar{x}$, $\bar{y}$, $\bar{v}$ et $\bar{w}$, si $\bar{y} \cap Vlib(\bar{v}) = \emptyset$ et que $\bar{x} \cap \bar{y} = \emptyset$ alors : $u < \bar{x} := \bar{v} >< \bar{y} := \bar{w} >= u < \bar{x} := \bar{v}, \bar{y} := \bar{w} >$.*

Lemme 12. (substitution après changement de variables libres) *Si les $\bar{y}$ n'ont pas d'occurrences dans u, alors : $u < \bar{x} := \bar{y} >< \bar{y} := \bar{w} >= u < \bar{x} := \bar{w} >$*

2.2 α-équivalence

On va définir ici une relation d'équivalence, l'α-équivalence, visant à "rassembler" les termes qui peuvent s'obtenir en renommant des variables liées sans changer le sens de la formule.

Définition 13. On définit la relation d'α-équivalence par induction sur t :
- si $t \in V$ alors $t \Leftrightarrow_\alpha t'$ ssi $t' = t$;
- si $t = uv$ alors $t \Leftrightarrow_\alpha t'$ ssi $t' = u'v'$ avec $u \Rightarrow_\alpha u'$
- si $t = \lambda x.u$ alors $t \Leftrightarrow_\alpha t'$ ssi $t' = \lambda x'.u'$ avec $v < x := y > \Leftrightarrow_\alpha v' < x' := y >$ pour presque tout y.

Exemple 14. 1. $\lambda x.x =_\alpha \lambda y.y$

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

3. $\lambda x.y \neq_\alpha \lambda x.x$

Lemme 15. *Si x' n'apparait pas dans u, alors $\lambda x.u =_\alpha \lambda x'.u < x := x' >$.*

Démonstration. D'après le lemme de changement de variables libres, $u < x := y >= u < x := x' >< x' := y >$ pour tout $y \in V$ donc $\lambda x.u =_\alpha \lambda x'.u < x := x' >$. □

Proposition 16. *Si $u =_\alpha u'$ et aucune variable libre de $\bar{v}$ n'est liée dans u ou u' alors $u < \bar{x} := \bar{v} >=_\alpha u' < \bar{x} := \bar{v} >$.*

Démonstration. On remarque que u et u' ont les mêmes variables libres. D'autre part on peut supposer que tous les x_i sont libres dans u et u'. On raisonne par induction sur u, le seul cas difficile est celui de l'abstraction.

Alors, $u = \lambda z.w$ et $u' = \lambda z'.w'$ avec $w < z := y >=_\alpha w' < z' := y >$ pour tout y hors d'un ensemble fini F. On sait que z et z' n'appartiennent pas à $\bar{x}$ car on a supposé que les x_i étaient libres dans u et u'. On a donc $u < \bar{x} := \bar{v} >= \lambda x.w < \bar{x} := \bar{v} >$ et de même pour u'.

On prend $y \notin \bar{x}$, alors : $w < \bar{x} := \bar{v} >< z := y >$
$= w < \bar{x} := \bar{v}, z := y >$ par le lemme de subtitutions simples successives ($z \notin Vlib(\bar{v})$ et $y \notin \bar{x}$ par hypothèse)
$= w < z := y, \bar{x} := \bar{v} >$
$= w < z := y >< \bar{x} := \bar{v} >$

De même pour w'.

Si de plus $y \notin F$, alors $w < z := y >=_\alpha w' < z' := y >$. On conclut en applicant l'hypothèse d'induction à $w < z := y >$ et $w' < z' := y >$. □

Proposition 17. *$=_\alpha$ est une relation d'équivalence.*

Proposition 18. *La relation d'α-équivalence est contextuelle :*
- *elle est réflexive ;*
- *si $t =_\alpha t'$, alors $\lambda x.t =_\alpha \lambda x.t'$ pour tout x de V.*
- *si $u =_\alpha u'$ et $v =_\alpha v'$, alors $uv =_\alpha u'v'$.*

Démonstration. Le plus important de voir est qu'elle est conservée par l'abstraction. Supposons $u =_\alpha u'$, on veut montrer que $\lambda x.u =_\alpha \lambda x'.u'$, pour tout $x \in V$. Il suffit de montrer que $u < x := y >= u' < x := y >$ pour presque tout y. On sait qu'il suffit de prendre $y \notin Vlib(u) \cup Vlib(u')$, c'est donc bien vrai pour presque tout y. □

2.3 Substitution correcte

On appelle substitution correcte le mécanisme qui permet de remplacer dans un terme des variables libres par des termes sans que les variables libres de ces termes soient capturées. Cependant, il faut d'abord s'assurer que c'est possible.

Lemme 19. *Soit $u \in L$ et $F \subset V$ tel que $V - F$ soit infini. Alors il existe $u' \Leftrightarrow_\alpha u$ tel qu'aucune variable de F n'est liée dans u'.*

Démonstration. Il suffit de faire une induction sur u. On va détailler le cas où u est une abstraction, $u = \lambda x.w$. D'après les résultats précédents, soit $x' \notin Vlib(w)$ alors $\lambda x.w =_\alpha \lambda x'.w < x := x' >$. D'autre part, par hypothèse d'induction on a $w' =_\alpha w < x := x' >$ et les variables de F ne sont pas liées dans w'. On en déduit $\lambda x.w =_\alpha \lambda x'.w'$ et ce dernier terme convient si on prend $x' \notin F$. □

Définition 20. $u[\bar{x} := \bar{v}] = u' < \bar{x} := \bar{v} >$ où u' est un terme quelconque α-équivalent à u et dont aucune variable liée n'est libre dans $\bar{v}$.

Cette définition est correcte grâce au lemme précédent. De plus elle définit bien $u[\bar{x} := \bar{v}]$ de manière unique puisque soit u' et u'' vérifiant la propriété, alors ils sont α-équivalents et donc $u' < \bar{x} := \bar{v} >=_\alpha u'' < \bar{x} := \bar{v} >$. Le résultat est donc défini à α-équivalence près.

Exemple 21. $(\lambda x.x(\lambda y.y))[y := x] =_\alpha \lambda z.z(\lambda y.x)$

Lemme 22. (substitutions correctes successives) *Soient $\bar{x}$ et $\bar{y}$ tels que $\bar{x} \cap \bar{y} = \emptyset$:*

1. $u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{x} := \bar{v}', \bar{y} := \bar{w}]$ *avec* $\bar{v}' = \bar{v}[\bar{y} := \bar{w}]$;
2. *si de plus les x_i ne sont pas libres dans $\bar{w}$ alors :* $u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{y} := \bar{w}][\bar{x} := \bar{v}']$

Démonstration. 1. Par induction sur u.

2. $u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{x} := \bar{v}', \bar{y} := \bar{w}]$
   $= u[\bar{y} := \bar{w}, \bar{x} := \bar{v}']$
   $= u[\bar{y} := \bar{w}][\bar{x} := \bar{v}']$

□

3 β-réduction

À partir de maintenant on travaillera uniquement à α-réduction près, on travaille sur $\Lambda = L/ =_\alpha$.

3.1 β_0-réduction

La β_0-réduction représente une étape de calcul, elle consiste à appliquer une abstraction à un endroit du terme.

Définition 23. L'ensemble $A(t)$ des β_0-réduits de t est défini par induction :
- si $t \in V$ alors $A(t) = \emptyset$;
- si $t = \lambda x.u$ alors $t' \in A(t)$ ssi $t' = \lambda x.u'$ avec $u' \in A(u)$;
- si $t = uv$ alors $t' \in A(t)$ ssi
  - $t' = uv'$ avec $v' \in A(v)$;
  - ou $t' = u'v$ auec $u' \in A(u)$;
  - ou $u = \lambda x.w$ et $u' = w[x := v]$.

Exemple 24. 1. $(\lambda x.x)y \to_{\beta_0} y$

2. $((\lambda x.x)x)((\lambda x.x)x) \to_{\beta_0} x(\lambda x.x)$
3. $((\lambda x.x)x)((\lambda x.x)x) \to_{\beta_0} (\lambda x.x)x$

On appelle *redex* les "expressions réductibles" par β_0 c'est-à-dire les termes de la forme $(\lambda x.u)v$.

3.2 β-réduction

Là où la β_0-réduction représentait une (et même exactement une) étape de calcul, la β-réduction représente un calcul de longueur quelconque.

Définition 25. $\rightarrow_\beta$ est la clôture réflexive et transitive de $\rightarrow_{\beta_0}$.

Exemple 26. 1. $((\lambda x.x)x)((\lambda x.x)x) \rightarrow_\beta xx$

2. pour tout $k \geqslant 2$, $(\lambda x.xxx)(\lambda x.xxx) \rightarrow_\beta \underbrace{(\lambda x.xxx)(\lambda x.xxx)\ldots(\lambda x.xxx)}_{k \text{ fois } (\lambda x.xxx)}$

Lemme 27. *β est contextuelle.*

3.3 Normalisation

Avec la β-réduction on a défini ce qu'était un calcul. La notion de forme normale correspond au résultat de ce calcul.

Définition 28. Un terme t est dit normal ssi $t \rightarrow_\beta t'$ implique $t = t'$ c'est-à-dire si il ne contient pas de redex.

Exemple 29. $\lambda x.x$ est normal, $(\lambda x.x)y$ ne l'est pas.

Définition 30. On dit qu'un terme u est une forme normale de t ssi u est normal et $t \rightarrow_\beta u$.

Exemple 31. $(\lambda x.xx)(\lambda x.x)$ a pour forme normale $\lambda x.x$ car $(\lambda x.xx)(\lambda x.x) \rightarrow_{\beta_0} (\lambda x.x)(\lambda x.x) \rightarrow_{\beta_0} \lambda x.x$.

Définition 32. Un terme est dit normalisable ssi il admet une forme normale.

4 Théorème de Church-Rosser

Ce résultat, introduit par Alonzo Church et J. B. Rosser en 1936 dans [2] est un théorème fondamental du λ-calcul. La preuve que nous allons donner est inspirée des travaux de Tait et Martin-Lof, pour plus de détails voir [3].

Définition 33. Une relation $\rightarrow$ sur les termes est dite *confluente* si pour tous termes t, u, v, tels que $t \rightarrow u$ et $t \rightarrow v$ il existe t' tel que $u \rightarrow t'$ et $v \rightarrow t'$

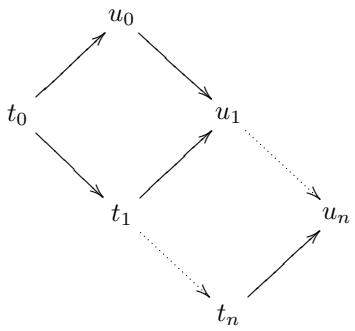
Cela implique notamment que si un terme a une forme normale alors elle est unique.

Théorème 34. (Church-Rosser) *La relation $\rightarrow_\beta$ est confluente.*

Lemme 35. *Si une relation est confluente, alors sa clôture transitive l'est aussi.*

Démonstration. Soit $\rightarrow$ une relation.

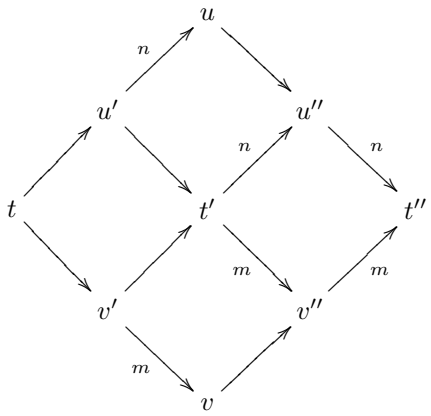
On montre d'abord, par récurrence, que si $t_0 \rightarrow u_0$ et $t_0 \rightarrow \ldots \rightarrow t_n$ alors il existe u_1, ..., u_n tels que $u_0 \rightarrow \cdots \rightarrow u_n$ et $t_n \rightarrow u_n$. Pour n nul cela découle de la confluence de $\rightarrow$. Sinon, on a $t_0 \rightarrow u_0$ et $t_0 \rightarrow t_1$ donc, par confluence, on a u_1 tel que $u_0 \rightarrow u_1$ et $t_1 \rightarrow u_1$. Il suffit alors d'appliquer l'hypothèse de récurrence à t_1, t_2, ..., t_n et u_1.



On notera maintenant $u \to^n v$ ssi $u \to w \to^{n-1} v$ pour un certain w avec la convention que $u \to^0 v$ ssi $u = v$

Dans le cas général, on fait une récurrence sur $k = \min(n, m)$. Si $k = 1$ alors on est ramené au cas précédent. Sinon, on a $t \to u'$, $t \to v'$, $u' \to^n u$ et $v' \to^m v$. Par confluence, il existe t' tel que $u' \to t'$ et $v' \to t'$. De plus d'après le résultat précédent on a u'' et v'' tels que $u \to u''$, $t' \to^n u''$, $v \to v''$ et $t' \to^m v''$. Alors, par hypothèse de récurrence, on a t'' tel que $u'' \to^n t''$ et $v'' \to^m t''$.

Finalement, on a $u \to u'' \to^n t''$ d'où $u \to^{n+1} t''$ et de même $v \to^{m+1} t''$.



□

Il suffit donc de trouver une relation confluente dont la clôture transitive est $\to_\beta$. On ne peut pas prendre la clôture refléxive de $\to_{\beta_0}$ car elle n'est pas confluente. Par exemple, soit $\delta = \lambda x.xx$ et $I = \lambda x.x$, le terme $\delta(II)$ se réduit en $(II)(II)$ et en δI. Le premier doit se réduire en $I(II)$ ou $(II)I$ et le second se réduit impérativement en II.

On va définir la relation $\Rightarrow_0$ qui correspond à la réduction simultanée de tous les redex du terme (mais pas des redex qui apparaîssent suite à une réduction, elle n'est donc pas transitive). Ce sont Tait et Martin-Lof qui ont introduit les premiers cette notion de réduction simultanée.

Définition 36. L'ensemble $B(t)$ des réduits de t pour $\Rightarrow_0$ est défini par induction structurelle comme suit :

– si $t \in V$ alors $B(t) = \{t\}$;

– si $t = \lambda x.g$ alors $t' \in B(t)$ ssi $t' = \lambda x.u$ avec $u \in B(t)$;

- si $t = uv$ alors $t' \in B(t)$ ssi
  - $t' = u'v'$ avec $u' \in B(u)$ et $v' \in B(v)$;
  - ou $u = \lambda x.w$ et $t' = w'[x := v']$ avec $w' \in B(w)$ et $v' \in B(v)$.

Lemme 37. *1. $\rightarrow_{\beta_0}$ implique $t \Rightarrow_0 t'$.*

2. $t \Rightarrow_0 t'$ implique $t \rightarrow_\beta t'$.

3. $\rightarrow_\beta$ est la clôture transitive de $\Rightarrow_0$.

Démonstration. 1. Il suffit de montrer que $A(t) \subset B(t)$, ce que l'on fait facilement par induction sur la structure de t.

2. On raisonne également par induction sur la structure de t et on utilise le fait que $\rightarrow_\beta$ est contextuelle.

3. (a) $\rightarrow_\beta = \rightarrow_{\beta_0}^*$ par définition.
   (b) $\rightarrow_{\beta_0}^* \subset \Rightarrow_0^*$ d'après 1.
   (c) $\Rightarrow_0^* \subset \rightarrow_\beta$ car $\rightarrow_\beta$ est réflexive, transitive et contient $\Rightarrow_0$.
   On a donc $\Rightarrow_0^* = \rightarrow_\beta$.

□

Lemme 38. *Si $t \Rightarrow_0 t'$ et $\bar{v} \Rightarrow_0 \bar{v}'$ alors $t[\bar{x} := \bar{v}] \Rightarrow_0 t'[\bar{x} := \bar{v}']$.*

Démonstration. On raisonne par induction sur la structure de t :
- si t est une variable alors $t = t'$ et c'est immédiat ;
- si $t = \lambda y.u$ on travaille à α-équivalence près donc peut supposer que y n'est libre ni dans $\bar{v}$ ni dans $\bar{x}$. On a alors $t' = \lambda y.u'$ avec $u \Rightarrow_0 u'$. Donc $t[\bar{x} := \bar{v}] = \lambda y.u[\bar{x} := \bar{v}] \Rightarrow_0 \lambda y.u'[\bar{x} := \bar{v'}] = t'[\bar{x} := \bar{v'}]$ par hypothèse d'induction ;
- si $t = uw$ alors
  - $t' = u'w'$ avec $u \Rightarrow_0 u'$ et $w \Rightarrow_0 w'$ et alors c'est immédiat ;
  - ou $t = (\lambda y.u)w$ et $t' = u'[y := w']$ avec $u \Rightarrow_0 u'$ et $w \Rightarrow_0 w'$. On suppose de nouveau que y n'est libre ni dans $\bar{v}$ ni dans $\bar{x}$ et on a $t[\bar{x} := \bar{v}] = (\lambda y.u[\bar{x} := \bar{v}])w[\bar{x} := \bar{v}] \Rightarrow_0 u'[\bar{x} := \bar{v'}][y := w'[\bar{x} := \bar{v'}]] = u'[y := w'][\bar{x} := \bar{v'}] = t'[\bar{x} := \bar{v'}]$, par hypothèse d'induction et d'après le lemme des substitutions successives.

□

Proposition 39. *Pour tout terme t, il existe un terme t^* tel que pour tout terme $u \in B(t)$, $u \Rightarrow_0 t^*$.*

Chaque réduit d'un terme t par $\Rightarrow_0$ correspond à une stratégie de réduction. Le terme que nous allons créer correspond à une réduction complète. Il est alors naturel de penser que toute réduction "incomplète" peut se complèter en une réduction complète.

Démonstration. On construit t^* par induction :
- $(x)^* = x$ si $x \in V$.
- $(\lambda x.t) = \lambda x.t^*$.
- $(vw)^* = v^*w^*$ si v n'est pas une abstraction.
- $(vw)^* = r^*[x := w^*]$ si $v = \lambda x.r$.

On vérifie ensuite que cette construction vérifie bien la propriété annnoncée, encore une fois par induction structurelle. On ne vérifiera que le cas du redex.

On a $t = (\lambda x.r)w$. Alors, $t^* = r^*[x := w^*]$. Si $t \Rightarrow_0 u$, alors $u = (\lambda x.r')w'$ ou $u = r'[x := w']$, $r \Rightarrow_0 r'$ et $w \Rightarrow_0 w'$. On a alors $r' \Rightarrow_0 r^*$ et $w' \Rightarrow_0 w^*$ par hypothèse d' induction. Dans le premier cas on a $u \Rightarrow_0 t^*$ par définition de $\Rightarrow_0$. Dans le second cas, d'après le lemme précédent, $u = r'[x := w'] \Rightarrow_0 r^*[x :=^*] = t^*$. □

Références

[1] H. P. Barendregt. *The lambda calculus : its syntax and semantics / H.P. Barendregt*. North-Holland Pub. Co. ; sole distributors for the U.S.A. and Canada Elsevier North-Holland, Amsterdam ; New York : New York :, 1981.

[2] Alonzo Church and J. B. Rosser. Some Properties of Conversion. *Transactions of the American Mathematical Society*, 39(3) :472–482, 1936.

[3] Jean-Jacques Lévy. An algebraic interpretation of the lambda beta - calculus and a labeled lambda - calculus. In *Proceedings of the Symposium on Lambda-Calculus and Computer Science Theory*, pages 147–165, London, UK, 1975. Springer-Verlag.