Prolégomène au λ-calcul

Rienzo

24 juillet 2025

Résumé

Ces prolégomènes exposent les fondements du λ-calcul de Church. Après avoir établi la syntaxe et les conventions de notation, nous définissons rigoureusement les notions de variable libre et liée, ainsi que la représentation de De Bruijn. Les mécanismes de réduction β, η et α sont analysés en détail, suivis d'une présentation des combinateurs canoniques. L'étude culmine avec une démonstration du théorème de Church-Rosser et de ses conséquences pour la confluence des réductions.

Table des matières

Introduction

Le λ-calcul, formalisé dans les années 1930 par Alonzo Church [1], vise à définir rigoureusement la notion de fonction calculable. Par son expressivité, ce formalisme s'est rapidement imposé comme un fondement majeur de la théorie de la computation, capable de modéliser l'ensemble des calculs réalisables par une machine de Turing − systèmes dits *Turing-complets*.

Il offre également une réponse essentielle à l'*Entscheidungsproblem* − problème de décision formulé dès Leibniz − qui s'articule autour des questions suivantes :

(1) *Existe-il un langage dans lequel tous les problèmes puissent être formulés ?*

(2) *Existe-il une méthode de décision qui permette de statuer sur la véracité de toute proposition issue de ce langage ?*

Le théorème de Church établit l'impossibilité d'une telle méthode dans le cadre du λ-calcul : aucun algorithme ne peut décider, pour toute expression, sa validité. Ce résultat clôt l'*Entscheidungsproblem*. Par ailleurs, la *thèse de Church* soutient que la notion intuitive de fonction calculable coïncide avec celle des fonctions exprimables en λ-calcul ou par une machine de Turing.

Aujourd'hui, le λ-calcul demeure un pilier fondamental en informatique théorique, à la base de la programmation fonctionnelle, de la logique combinatoire et de la théorie des types. Véritable édifice autonome, il incarne un système universel capable d'exprimer, calculer et démontrer toute construction algorithmique. À l'instar des équations de Maxwell en physique, le λ-calcul représente le principe premier et universel de l'informatique, le fondement à partir duquel tout s'articule.

1. Dès 1893, Frege observa que toute fonction pouvait se réduire à une fonction d'un seul argument. Il introduisit alors $\otimes$, un opérateur unaire tel que $(\otimes A)(B) = A + B$.

Présentation

Le λ-calcul, comme vu en introduction, est un formalisme mathématique qui témoigne d'une expressivité remarquable pour un nombre restreint de symboles. Il repose sur la notion mathématique de fonction *pure*, c'est à dire que les résultats que l'on obtiendra ne dépendront que des arguments que nous lui auront fournis. Dans cette première partie, destinée à une présentation rigoureuse du λ-calcul, nous nous attacherons à donner une définition exacte de ce formalisme, tant d'un point de vue syntaxique que sémantique, avant de nous intéresser à la notation de De Bruijn.

Syntaxe

Soit $V = \{x_1, x_2, \ldots, x_n\}$ un ensemble de variables. L'ensemble des λ-termes, noté Λ, est défini inductivement par les règles suivantes :

$$\begin{aligned} x \in V &\Rightarrow x \in \Lambda, \\ M, N \in \Lambda &\Rightarrow (M\,N) \in \Lambda, \\ M \in \Lambda,\ x \in V &\Rightarrow (\lambda x.\, M) \in \Lambda. \end{aligned}$$

Ainsi, tout terme $T \in \Lambda$ s'écrit selon l'une des formes :

$$T ::= x \quad | \quad \lambda x.\, T \quad | \quad (M\,N)$$

où x désigne une *variable*, libre ou liée, $\lambda x.\, T$ représente une fonction anonyme introduisant la variable x, que T peut employer dans son corps et finalement $(M\,N)$ exprime l'application du terme M au terme N. L'application correspond conceptuellement à une substitution : on remplace la variable liée par N dans le corps de l'abstraction de M, opération que l'on note $M[N/x]$.

Enfin, il convient d'établir les conventions de notation qui gouvernent l'écriture des λ-termes. Par souci de clarté et d'économie typographique, nous privilégions systématiquement l'omission des parenthèses lorsque celle-ci n'introduit aucune ambiguïté.

Notation 2.1 (Priorité). *La hiérarchie des opérateurs s'ordonne comme suit :*

(i) L'abstraction λ jouit de la précédence la plus élevée et s'associe à droite

(ii) L'application s'associe à gauche

Exemple 2.1. Les règles de priorité déterminent l'interprétation des expressions :

$$\begin{aligned} \lambda x.\, x\,y\,z &\equiv \lambda x.\, ((x\,y)\,z) \quad \text{(application associe à gauche)} \\ \lambda x.\lambda y.\, x\,y &\equiv \lambda x.\, (\lambda y.\, (x\,y)) \quad \text{(abstraction associe à droite)} \\ f\,g\,h &\equiv (f\,g)\,h \quad \text{(priorité de l'application)} \end{aligned}$$

Notez que l'abstraction λ s'étend aussi loin que possible vers la droite.

Exemple 2.2. Certaines parenthèses peuvent être omises sans ambiguïté :

$$\lambda x.\,(\lambda y.\,(\lambda z.\,(x\,(y\,z)))) \equiv \lambda xyz.\,x\,(y\,z)$$
$$\lambda f.\,(\lambda g.\,(\lambda x.\,((f\,x)\,(g\,x)))) \equiv \lambda fgx.\,f\,x\,(g\,x)$$

En revanche, les parenthèses sont indispensables dans ces cas :

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

Remarque 2.3. L'usage des parenthèses demeure indispensable lors de l'application d'un terme à une abstraction. Ainsi, l'expression $(\lambda x.\,x)\,y$ indique explicitement que le terme y s'applique à l'abstraction $\lambda x.\,x$, produisant le résultat y par β-réduction.

Variables libres et liées

L'ensemble des variables libres d'un terme T, noté $\mathrm{FV}(T)$, se définit de manière inductive comme suit :

$$\begin{aligned}\mathrm{FV}(x) &= \{x\}\\ \mathrm{FV}(\lambda x.\,T) &= \mathrm{FV}(T) \setminus \{x\}\\ \mathrm{FV}(M\,N) &= \mathrm{FV}(M) \cup \mathrm{FV}(N)\end{aligned}$$

Une variable est dite *liée* dans un terme T si elle a été introduite précédemment par une abstraction. Une variable est dite *libre* si elle n'est pas liée, c'est-à-dire qu'aucune abstraction antérieure ne l'a introduite.

Exemple 2.4. Considérons le terme $\lambda x.\,y$. Ici x est introduite par λx, donc elle est *liée*, tandis que y n'est introduite par aucune abstraction, on dit donc qu'elle est *libre*. Ainsi, on a $\mathrm{FV}(\lambda x.\,y) = \{y\}$.

Exemple 2.5. Voici un exemple poignant tel qu'il serait effectué à la main pour déterminer les variables libres d'un terme :

$$\begin{aligned}\mathrm{FV}(xy) &= \mathrm{FV}(x) \cup \mathrm{FV}(y) = \{x\} \cup \{y\} = \{x, y\}\\ \mathrm{FV}(\lambda x.\,xy) &= \mathrm{FV}(xy) \setminus \{x\} = \{x, y\} \setminus \{x\} = \{y\}\\ \mathrm{FV}(\lambda x.\lambda y.\,x) &= \mathrm{FV}(\lambda y.\,x) \setminus \{x\} = (\mathrm{FV}(x) \setminus \{y\}) \setminus \{x\} = \emptyset\end{aligned}$$

Un λ-terme qui ne contient aucune variable libre est dit *clos*.

Exemple 2.6. Voici quelques exemples de termes clos.

$$\lambda x.\,x$$
$$\lambda x.\lambda y.\,x$$
$$\lambda f.\lambda x.\,f\,x$$
$$\lambda f.\lambda g.\lambda x.\lambda y.\lambda z.\,f\,(g\,x\,y)\,((\lambda u.\lambda v.\,u\,(v\,v))\,(\lambda w.\,w\,x)\,(\lambda a.\,g\,a\,z))$$

Notation de De Bruijn

La notation de De Bruijn constitue une représentation alternative des λ-termes où les variables sont remplacées par des indices numériques.

Définition 2.1 (Syntaxe de De Bruijn). *L'ensemble des termes de De Bruijn se définit par la grammaire suivante :*

$$e ::= n \quad | \quad \lambda e \quad | \quad e\, e$$

où $n \in \mathbb{N}$ représente un indice de De Bruijn.

Notation 2.2 (Indices de De Bruijn). *Un indice n fait référence à la n-ième abstraction englobante, en comptant depuis la position courante vers l'extérieur (indexation à partir de 0).*

Exemple 2.7. Voici quelques traductions vers la notation de De Bruijn :

$$\begin{aligned}
\lambda x.\, x &\equiv \lambda\, 0 \\
\lambda x.\lambda y.\, x &\equiv \lambda\lambda\, 1 \\
\lambda x.\lambda y.\, x\, y &\equiv \lambda\lambda\, 1\, 0 \\
\lambda x.\lambda y.\, y\, (\lambda z.\, x\, z) &\equiv \lambda\lambda\, 0\, (\lambda\, 2\, 0)
\end{aligned}$$

Réduction

La réduction est le mécanisme par lequel les λ-termes se simplifient, conformément à des règles d'inférence, relevant de la congruence entre les termes. Nous en faisons l'étude dans cette section.

β_0-réduction

La β_0-réduction, dont la transformation se note par la flèche $\to_\beta$, est l'unité de réduction fondamentale du λ-calcul, défini comme suit, à α-réduction près :

Définition 3.1. *Soit M et N appartenant tout deux à Λ, dont N est un λ-terme clos. On dit que M subit une β_0-réduction en N, ou que M se réduit une fois en N, si l'on peut obtenir le terme M' en remplaçant la variable liée x dans M par le terme N, c'est à dire si :*

$$M \to_\beta M' \quad \textit{où} \quad M' = M[N/x]$$

Le schème de réduction β_0 s'opère selon les règles suivantes :

$$\begin{aligned}
x &\to_\beta x \\
\lambda x.\, M &\to_\beta \lambda x.M \\
(\lambda x.\, M)\, N &\to_\beta M[N/x]
\end{aligned}$$

β-réduction

La β-réduction, quand à elle, désigne l'ensemble des β_0-réductions nécessaires pour atteindre l'état dit β-normal d'un λ-terme, c'est à dire l'état irréductible d'un terme M, noté M^β, tel que :

Définition 3.2. *Soit M et N appartenant tous deux à Λ. On dit que M se β-réduit en N, noté $M \twoheadrightarrow_\beta N$, s'il existe une suite finie de β_0-réductions qui mène de M à N, c'est-à-dire s'il existe des termes $M_1, M_2, \ldots, M_k$ tels que :*

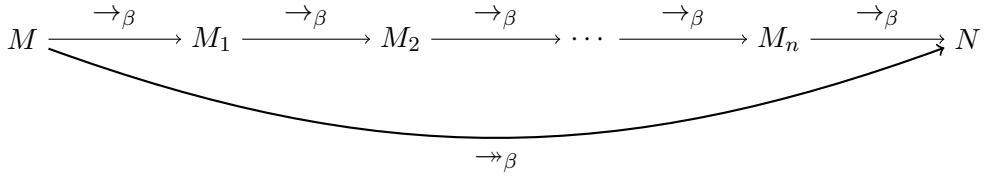
$$M \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots \to_\beta M_k \to_\beta N$$

Cette β-réduction se note par la flèche $\twoheadrightarrow_\beta$ et se définit comme suit :

Définition 3.3. *Soit M et N appartenant tout deux à Λ. On dit que M subit une β-réduction en N, ou que M se réduit complètement en N, si l'on peut obtenir le terme β-normal que l'on note M^β à l'issue d'une suite finie de β_0-réductions, c'est à dire si :*

$$\exists M^\beta,\ M \twoheadrightarrow_\beta M^\beta$$

Un schéma récapitulatif qui met en exergue les différentes réductions $\to_\beta$ et $\twoheadrightarrow_\beta$ ainsi que l'illustration de la β-réductibilité du λ-terme M en N :

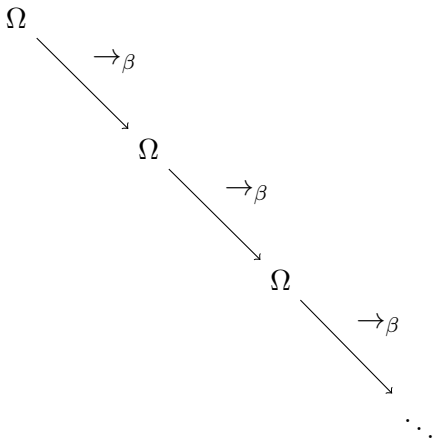


Il existe, toutefois, des termes qui ne soient point β-réductibles.

Exemple 3.1. Considérons le combinateur $\Omega = \omega\omega$ que nous avons étudié précédemment. Lors de sa β-réduction, nous obtenons :

$$\begin{aligned} \Omega &= (\lambda x.xx)(\lambda x.xx) \\ &\to_\beta (\lambda x.xx)(\lambda x.xx) \\ &= \Omega \end{aligned}$$

Ce combinateur se réduit indéfiniment en lui-même, créant une chaîne infinie de réductions sans jamais atteindre de forme normale :



Par conséquent, Ω n'est pas β-réductible et n'a pas de forme β-normale. Il constitue un exemple paradigmatique de terme divergent dans le λ-calcul.

η-réduction

Finalement, la η-réduction, notée $\rightarrow_\eta$, est une forme de réduction visant à alléger les λ-termes en éliminant les abstractions superflues. Elle est définie comme suit :

Définition 3.4. *Soit M un λ-terme et x une variable. On dit que le terme $\lambda x.Mx$ subit une η-réduction en M, noté $(\lambda x.Mx) \rightarrow_\eta M$, si et seulement si $x \notin FV(M)$.*

Plus formellement :

$$(\lambda x.Mx) \rightarrow_\eta M \quad \textit{si et seulement si } x \notin FV(M)$$

Cette règle exprime l'idée qu'une fonction qui se contente d'appliquer son argument à une autre fonction M (indépendante de l'argument en question) est extensionnellement équivalente à M elle-même.

Exemple 3.2.

$$\begin{aligned} (\lambda x.fx) &\rightarrow_\eta f \quad \text{si } x \notin \mathrm{FV}(f) \\ (\lambda y.(\lambda z.z)y) &\rightarrow_\eta (\lambda z.z) \\ (\lambda x.gxy) &\not\rightarrow_\eta gy \quad \text{car } x \in \mathrm{FV}(gxy) \end{aligned}$$

La η-réduction capture ainsi le principe d'extensionnalité : deux fonctions qui produisent le même résultat pour tout argument sont considérées comme identiques.

α-réduction

Nul besoin de s'attarder sur l'α-réduction, qui consiste en un simple renommage de variable pour éviter que variables libres et liées ne se confondent. Concrètement, l'α-réduction, qui intervient en complément de la β-réduction qui a été détaillée précédemment, se définit comme suit :

Définition 3.5. *Soit T appartenant à Λ et $x, y \in V$ deux variables. On dit que T subit une α-réduction en y si l'on peut obtenir le terme T' en renommant toutes les occurrences de x dans T par y, c'est à dire si :*

$$T \to_\alpha T' \quad où \quad T' = T[y/x]$$

Combinateurs

Les combinateurs sont des λ-termes ayant de propriétés particulières, que l'on peut pour la première fois qualifier d'« utiles ». Dans cette section il sera question d'introduction nombre des plus importants d'entre eux.

Le système SKI constitue l'un des fondements de la logique combinatoire. Il se compose de trois combinateurs primitifs qui, par leur simplicité apparente, révèlent une expressivité remarquable :

Définition 4.1 (SKI).

$$\begin{aligned} \mathsf{I} &\equiv \lambda x.x \\ \mathsf{K} &\equiv \lambda x.\lambda y.x \\ \mathsf{S} &\equiv \lambda f.\lambda g.\lambda x.(fx)(gx) \end{aligned}$$

Le combinateur I retourne exactement son argument. Le combinateur K projette le premier argument en ignorant le second. Le combinateur S applique une valeur à deux fonctions et combine les résultats.

Exemple 4.1.

$$\begin{aligned} \mathsf{I}f &\to_\beta f \\ \mathsf{K}fg &\to_\beta f \\ \mathsf{S}fgx &\to_\beta fx(gx) \end{aligned}$$

Le système $\{\mathsf{S}, \mathsf{K}\}$ présente une propriété fondamentale : il est extensionnellement complet, c'est-à-dire capable de représenter tout λ-terme. Pour en établir la complétude, commençons par montrer que le combinateur identitaire I peut s'exprimer à l'aide de S et K :

$$\begin{aligned} \mathsf{SKK}x &\to_\beta \mathsf{K}x(\mathsf{K}x) \\ &\to_\beta x \end{aligned}$$

Donc $\mathsf{SKK} \equiv \mathsf{I}$, ce qui montre que le système $\{\mathsf{S}, \mathsf{K}\}$ est aussi expressif que $\{\mathsf{S}, \mathsf{K}, \mathsf{I}\}$.

La complétude du système $\{\mathsf{S}, \mathsf{K}\}$ s'établit par la construction d'un algorithme de traduction $\mathcal{T}$ qui convertit tout λ-terme en expression combinatoire :

$$\begin{aligned}\mathcal{T}[x] &= x \\ \mathcal{T}[MN] &= \mathcal{T}[M]\mathcal{T}[N] \\ \mathcal{T}[\lambda x.M] &= \mathcal{A}_x[\mathcal{T}[M]]\end{aligned}$$

où l'algorithme d'abstraction $\mathcal{A}_x$ est défini par :

$$\begin{aligned}\mathcal{A}_x[x] &= \mathsf{I} \\ \mathcal{A}_x[E] &= \mathsf{K}\,E \quad \text{si } x \notin \mathrm{FV}(E) \\ \mathcal{A}_x[EF] &= \mathsf{S}\,(\mathcal{A}_x[E])\,(\mathcal{A}_x[F])\end{aligned}$$

Par exemple, la traduction du terme $\lambda x.\lambda y.x$ donne :

$$\begin{aligned}\mathcal{T}[\lambda x.\lambda y.x] &= \mathcal{A}_x[\mathcal{T}[\lambda y.x]] \\ &= \mathcal{A}_x[\mathcal{A}_y[x]] \\ &= \mathcal{A}_x[\mathsf{K}\,x] \quad (\text{car } y \notin \mathrm{FV}(x)) \\ &= \mathsf{S}\,(\mathcal{A}_x[\mathsf{K}])\,(\mathcal{A}_x[x]) \\ &= \mathsf{S}\,(\mathsf{K}\,\mathsf{K})\,\mathsf{I} \\ &= \mathsf{K} \quad (\text{car } \mathsf{S}\,(\mathsf{K}\,\mathsf{K})\,\mathsf{I} \equiv \mathsf{K})\end{aligned}$$

Ainsi, tout λ-terme peut être exprimé en termes de S et K, ce qui établit la complétude du système combinatoire SK.

Cette universalité du système SK inspire naturellement la recherche d'autres bases combinatoires complètes. Le système BCKW constitue une alternative élégante, introduisant des combinateurs aux propriétés plus spécialisées :

Définition 4.2 (Système BCKW).

$$\begin{aligned}\mathsf{B} &\equiv \lambda f.\lambda g.\lambda x.f(gx) \quad \textit{(composition)} \\ \mathsf{C} &\equiv \lambda f.\lambda x.\lambda y.fyx \quad \textit{(permutation)} \\ \mathsf{K} &\equiv \lambda x.\lambda y.x \quad \textit{(constante)} \\ \mathsf{W} &\equiv \lambda f.\lambda x.fxx \quad \textit{(duplication)}\end{aligned}$$

Le combinateur B encode la composition fonctionnelle classique $(f \circ g)(x) = f(g(x))$. Le combinateur C permute les arguments d'une fonction binaire. Le combinateur W duplique un argument, appliquant une fonction à deux copies du même terme. Ce dernier s'exprime également entièrement en termes du système $\{\mathsf{S}, \mathsf{K}\}$, révélant une hiérarchie dans l'expressivité combinatoire :

$$\begin{aligned}\mathsf{B} &\equiv \mathsf{S(KS)K} \\ \mathsf{C} &\equiv \mathsf{S(S(K(S(KS)K))S)(KK)} \\ \mathsf{W} &\equiv \mathsf{SS(SK)}\end{aligned}$$

Cette bijection entre les combinateurs BCKW et SKI réaffirme une fois de plus le rôle central qu'occupe ce dernier dans la logique combinatoire.

Ensuite, comme autre type de combinateurs, on retrouve ceux dit de « divergence », qui sont des termes qui ne peuvent pas être réduits en une forme β-normale. Ils sont souvent utilisés pour illustrer les limites de l'évaluation dans le λ-calcul.

Définition 4.3 (Combinateurs de divergence).

$$\omega \equiv \lambda x.xx \quad \textit{(auto-application)}$$
$$\Omega \equiv \omega\omega \quad \textit{(divergent)}$$

Le combinateur ω applique un terme à lui-même. Son application à lui-même donne naissance à Ω, une expression non normalisable, se réduisant indéfiniment en elle-même :

Exemple 4.2.

$$\begin{aligned}\Omega &= (\lambda x.xx)(\lambda x.xx)\\ &\to_\beta (\lambda x.xx)(\lambda x.xx)\\ &= \Omega\end{aligned}$$

Par conséquent, Ω n'a pas de forme β-normale et constitue un exemple paradigmatique de terme divergent. Une propriété intéressante de Ω est qu'il permet de tester la stratégie d'évaluation d'un interpréteur : $\mathsf{K}\mathsf{I}\Omega$ se réduit vers I dans un interpréteur paresseux, mais diverge dans un interpréteur strict.

Finalement, les combinateurs de point fixe permettent d'implémenter la récursivité dans le λ-calcul. Ils satisfont la propriété fondamentale qu'un combinateur de point fixe $\mathcal{F}$ vérifie $\mathcal{F}f \equiv f(\mathcal{F}f)$ pour toute fonction f.

Définition 4.4 (Combinateur Y de Curry).

$$Y \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

Ce combinateur rend possible la récursivité par la propriété suivante : $Yf \equiv f(Yf)$. Plus rigoureusement, pour tout λ-terme T :

$$\begin{aligned}YT &\to_\beta (\lambda x.T(xx))(\lambda x.T(xx))\\ &\to_\beta T((\lambda x.T(xx))(\lambda x.T(xx)))\\ &\to_\beta T(YT)\end{aligned}$$

Définition 4.5 (Combinateur Θ de Turing). *Le combinateur de Turing se définit à partir d'un combinateur auxiliaire A :*

$$A \equiv \lambda xy.y(xxy)$$
$$\Theta \equiv AA$$

Proposition 4.3. *Pour tout $F \in \Lambda$, on a : $\Theta F \twoheadrightarrow_\beta F(\Theta F)$*

Démonstration.

$$\begin{aligned}\Theta F &\equiv AAF \\ &\to_\beta (\lambda y.y(AAy))F \\ &\to_\beta F(AAF) \\ &\equiv F(\Theta F)\end{aligned}$$

□

Le combinateur Θ évite une divergence prématurée en différant le déclenchement de la récursion jusqu'au moment nécessaire, ce qui permet l'implémentation effective de fonctions récursives dans des contextes à évaluation stricte.

À ce stade, il est pertinent d'introduire la distinction entre équivalence *intensionnelle* et équivalence *extensionnelle*. Deux termes sont *intensionnellement* équivalents s'ils partagent la même structure syntaxique ; ils sont *extensionnellement* équivalents s'ils produisent le même résultat pour toute entrée donnée. À cet égard, les combinateurs Y et Θ sont extensionnellement équivalents, bien qu'intensionnellement distincts : leurs définitions syntaxiques diffèrent, mais leur comportement observable coïncide.

Théorème de Church-Rosser

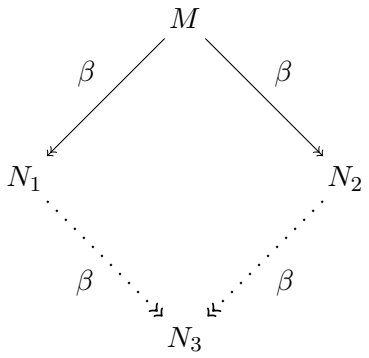
Bien qu'il ne sera pas faite de démonstration complète et rigoureuse du théorème de Church-Rosser (ou théorème de confluence) dans ce document, il est toutefois important de comprendre les implications de ce théorème ainsi que les conséquences qui découlent de son existence.

Pour ce faire, attachons nous d'abord à définir la notion de confluence avant d'introduire la formulation du théorème de Church-Rosser.

Définition 5.1 (Confluence). *La relation $\twoheadrightarrow_\beta$ est* confluente *si pour tous termes M, N_1, N_2 :*

$$M \twoheadrightarrow_\beta N_1 \text{ et } M \twoheadrightarrow_\beta N_2 \implies \exists N_3.\ N_1 \twoheadrightarrow_\beta N_3 \text{ et } N_2 \twoheadrightarrow_\beta N_3$$

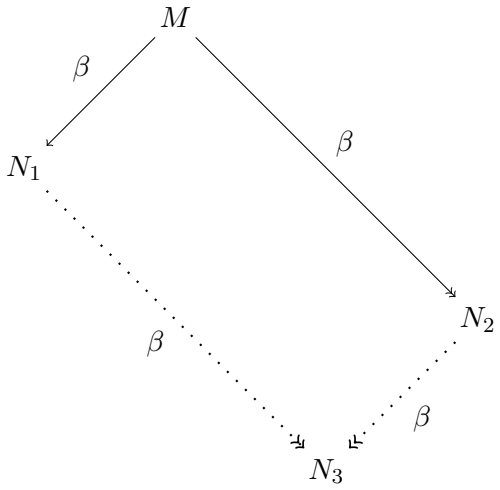
Théorème 5.1 (Church-Rosser). *Si $M \twoheadrightarrow_\beta N_1$ et $M \twoheadrightarrow_\beta N_2$, alors il existe N_3 tel que $N_1 \twoheadrightarrow_\beta N_3$ et $N_2 \twoheadrightarrow_\beta N_3$.*



Le théorème de Church-Rosser, ou théorème de confluence, affirme que la β-réduction est confluente : si un λ-terme peut être réduit de deux manières différentes, il existe toujours un terme commun atteignable par chacune de ces deux dernières.

Afin de prouver ce théorème, il suffit de prouver le *lemme de bandeau* qui se définit ainsi :

Lemme 5.2 (Lemme de bandeau). *Si $M \to_\beta N_1$ et $M \twoheadrightarrow_\beta N_2$, alors il existe N_3 tel que $N_1 \twoheadrightarrow_\beta N_3$ et $N_2 \to_\beta N_3$.*



Ce lemme repose sur trois autres lemmes auxiliaires, eux-mêmes fondés sur un nouvel ensemble de λ-termes que l'on désignera ultérieurement sous le nom de *termes marqués*. La démonstration du lemme de bandeau requiert en effet un mécanisme permettant de retracer l'historique des réductions appliquées à un rédex $\mathcal{R}$, condition nécessaire pour établir que les deux trajectoires de réduction issues de ce dernier convergent vers un terme commun − autrement dit, pour démontrer la confluence. À cette fin, nous introduisons l'ensemble $\underline{\Lambda}$ des $\underline{\lambda}$-termes, contenant Λ comme sous-ensemble strict : $\Lambda \subseteq \underline{\Lambda}$.

Définition 5.2 (Termes marqués). *Soit $\underline{\Lambda}$ l'ensemble des $\underline{\lambda}$-termes, ils se définissent par récurrence sur la structure des λ-termes classiques :*

$$\frac{x \in V}{x \in \underline{\Lambda}} \quad \frac{M \in \underline{\Lambda},\ N \in \underline{\Lambda}}{(MN) \in \underline{\Lambda}} \quad \frac{M \in \underline{\Lambda},\ x \in V}{(\lambda x.M) \in \underline{\Lambda}} \quad \frac{M \in \underline{\Lambda},\ N \in \underline{\Lambda},\ x \in V}{((\underline{\lambda} x.M)N) \in \underline{\Lambda}}$$

Définition 5.3 ($\underline{\beta}$-réduction). *La $\underline{\beta}$-réduction, notée $\to_{\underline{\beta}}$, est définie comme une extension de la β-réduction classique par les règles suivantes :*

$$(\underline{\lambda} x.M)N \to_{\underline{\beta}} M[N/x]$$
$$(\lambda x.M)N \to_{\underline{\beta}} M[N/x]$$

Comme pour le λ-calcul classique, nous notons $\twoheadrightarrow_{\underline{\beta}}$ la fermeture réflexive et transitive de $\to_{\underline{\beta}}$ (réductions multiples) et $=_{\underline{\beta}}$ sa fermeture symétrique, réflexive et transitive (équivalence entre termes marqués).

Définition 5.4 (Fonction d'application des termes marqués). *La fonction qui applique seulement les réductions des $\underline{\lambda}$-termes, est la fonction $\varphi : \underline{\Lambda} \to \Lambda$ définie inductivement de la manière suivante :*

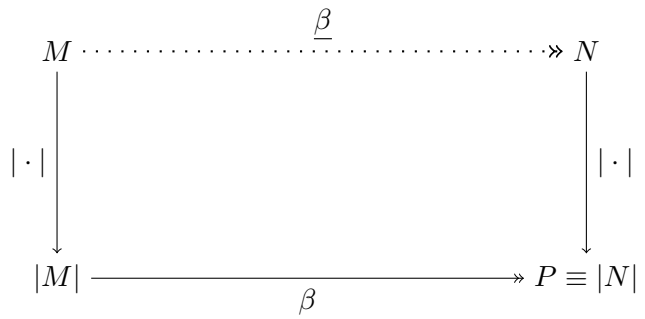
$$\begin{aligned} \varphi(x) &\equiv x & x &\in V \\ \varphi(MN) &\equiv \varphi(M)\varphi(N) & M, N &\in \underline{\Lambda} \\ \varphi(\lambda x.M) &\equiv \lambda x.\varphi(M) & M &\in \underline{\Lambda}, x \in V \\ \varphi((\underline{\lambda} x.M)N) &\equiv \varphi(M)[\varphi(N)/x] & M, N &\in \underline{\Lambda}, x \in V \end{aligned}$$

Notation 5.1 (Effacement des marques). *Soit $M \in \underline{\Lambda}$ un terme marqué. On notera $|M|$ le terme obtenu en effaçant purement et simplement l'ensemble des marques dans M.*

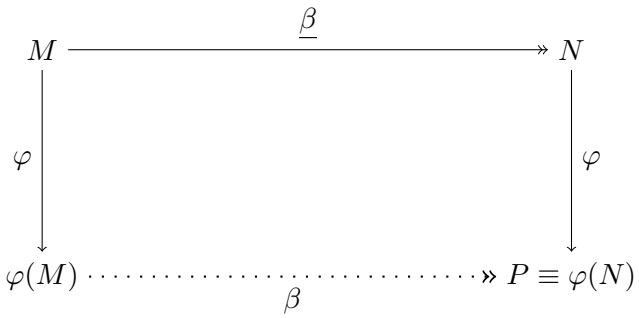
Notation 5.2. *Si $|M| \equiv N$ ou $\varphi(M) \equiv N$, on notera alors les transformations suivantes dans les schémas de la façon suivante :*

$$M \underset{|\cdot|}{\longrightarrow} N \text{ ou } M \underset{\varphi}{\longrightarrow} N.$$

Lemme 5.3. *Pour tout terme marqué $M \in \underline{\Lambda}$ et tout terme $P \in \Lambda$, si $|M| \twoheadrightarrow_{\beta} P$, alors il existe un terme marqué $N \in \underline{\Lambda}$ tel que $M \twoheadrightarrow_{\underline{\beta}} N$ et $|N| \equiv P$.*

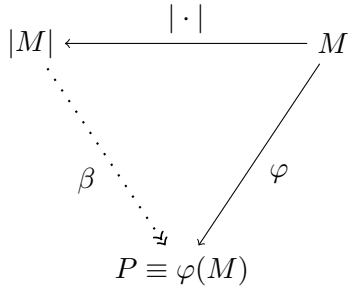


Lemme 5.4. *Soit $M, N \in \underline{\Lambda}$, si $M \twoheadrightarrow_{\underline{\beta}} N$, alors il existe un terme $P \in \Lambda$ tel que $\varphi(M) \twoheadrightarrow_\beta P$ et $|N| \equiv P$.*



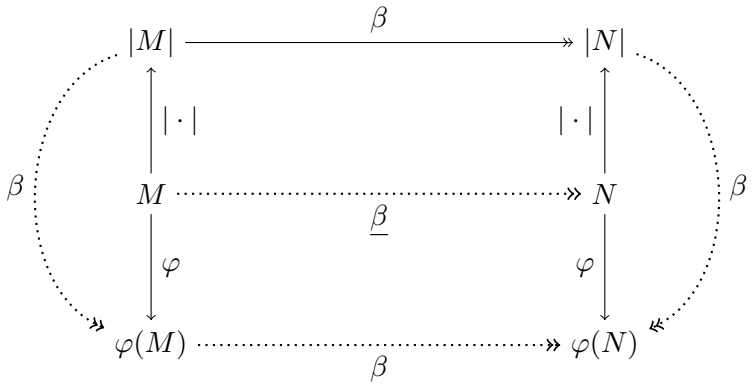
Le schéma montre que, pour un rédex $\mathcal{R}$ marqué dans un terme M, si l'on β-réduit ce dernier de façon arbitraire en un terme N contenant alors des résidus de $\mathcal{R}$, alors la réduction ciblée $\varphi(N)$ mène à un terme P tel qu'il existe une séquence de β-réductions depuis $\varphi(M)$ qui atteint P ; autrement dit il n'existe pas qu'un seul chemin de réduction pour parvenir à un λ-terme donné.

Lemme 5.5. *Soit $M \in \underline{\Lambda}$, si $M \to_\varphi P$, alors il existe une β-réduction finie telle que $|M| \twoheadrightarrow_\beta P$.*

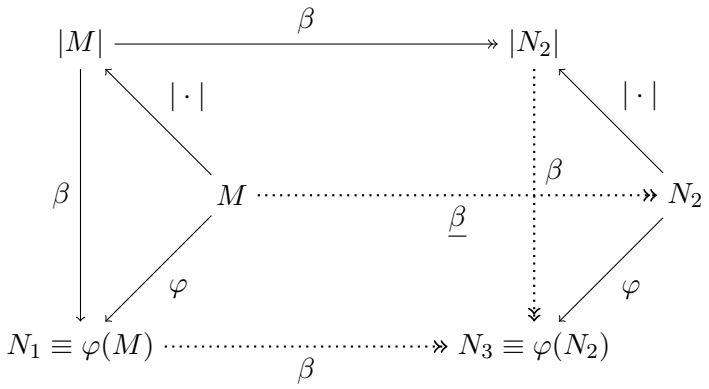


Enfin, nous pouvons établir le lemme du bandeau :

Démonstration. *Etant donné les lemmes 6.3, 6.4 et 6.5, le lemme de bandeau se démontre en composant ces derniers de la manière suivante.*



La confluence peut davantage être mise en avant si on s'en réfère à cette légère remodélation du schéma précédent :



Il en découle donc la démonstration du théorème de Church-Rosser. □

Conclusion

Le λ-calcul, à mi-chemin entre logique formelle et théorie computationnelle, incarne la quintessence de l'abstraction fonctionnelle. C'est avec étonnement que l'on constate combien l'absence apparente de mécanismes que l'on pourrait qualifier d'« utiles » n'altère en rien l'expressivité de ce formalisme. C'est précisément sur ce dépouillement apparent − qui n'est qu'apparent − que se fonde et s'érige l'universalité remarquable de ce système.

Ces prolégomènes ne constituent, il faut l'avouer, qu'un condensé de quelques investigations personnelles sur le sujet et ne sauraient prétendre à l'exhaustivité. Bien au contraire, nombre d'aspects y sont sommairement survolés, et la rigueur n'est pas toujours au rendez-vous. C'est pourquoi je vous encourage à poursuivre vos propres recherches en vous référant aux sources citées dans la bibliographie. Vous y trouverez les textes fondamentaux et contributions majeures ayant façonné ce formalisme, exposés avec la rigueur et la structure que requiert un véritable travail scientifique − qualités que ce document, simple synthèse de recherches personnelles menées dans des contraintes de temps étroites, n'a ni l'ambition ni la prétention de revendiquer.

Références

[1] A. Church, *The Calculi of Lambda Conversion*, Annals of Mathematical Studies, vol. 6, Princeton University Press, 1941.

[2] H. Barendregt et E. Barendsen, *Introduction to Lambda Calculus*, Version révisée, 1998-2000.

[3] Raúl Rojas, *A Tutorial Introduction to the λ-Calculus*, Freie Universität Berlin, WS-97/98, 1998.

[4] Cornell University, *CS 4110 − Programming Languages and Logics, Lecture 15 : De Bruijn, Combinators, Encodings*, 2018.

[5] Adrien Surée, *Congruence du λ-calcul non-typé*, 2010.