Le λ-calcul simplement typé
(Présentation à la Church)

Alexandre Miquel

1 La syntaxe

Les catégories syntaxiques des types et des termes sont définies par :

Types	τ, σ	$::=$	α	(Type de base)
		$\mid$	$\tau \to \sigma$	(Type flèche)
Termes	M, N	$::=$	x	(Variable)
		$\mid$	$\lambda x : \tau \,.\, M$	(Abstraction)
		$\mid$	MN	(Application)

Remarques :

1. La présentation du λ-calcul simplement typé est paramétrée par un ensemble de types de base, notés α, β, γ, etc. qu'il n'est pas nécessaire de préciser davantage. La présentation originelle (due à Church) ne fait intervenir qu'un seul type de base, noté o (« omicron »).
2. La variable x abstraite dans la construction $\lambda x : \tau \,.\, M$ porte une annotation de type « $: \tau$ » explicite ; on parle d'abstraction *à la Church*.
   Il existe une présentation alternative du λ-calcul simplement typé dans laquelle les abstractions sont dépourvues d'annotation ; on parle alors d'abstraction *à la Curry* (cf Caml & cours de sémantique).

L'ensemble des variables libres d'un terme M est noté $FV(M)$. Étant donnés deux termes M, N et une variable x, on note $M\{x := N\}$ le terme obtenu en remplaçant dans M toutes les occurrences libres de la variable x par le terme N (sans oublier d'effectuer tous les renommages nécessaires !)

2 La réduction

- La relation de *β-réduction en une étape*, notée $M \succ M'$, est définie inductivement à l'aide des 4 règles suivantes :

  Cas de base (règle β) :

$$\frac{}{(\lambda x : \tau \,.\, M)N \;\succ\; M\{x := N\}}$$

  Propagation sous le contexte :

$$\frac{M \;\succ\; M'}{\lambda x : \tau \,.\, M \;\succ\; \lambda x : \tau \,.\, M'} \qquad \frac{M \;\succ\; M'}{MN \;\succ\; M'N} \qquad \frac{N \;\succ\; N'}{MN \;\succ\; MN'}$$

- La relation de *β-réduction*, notée $M \succ^* M'$, est définie comme la clôture réflexive et transitive de la relation $M \succ M'$:

$$\frac{}{M \succ^* M} \qquad \frac{M \succ^* M' \quad M' \succ M''}{M \succ^* M''}$$

- Enfin, la relation de *β-conversion*, notée $M \simeq M'$, est définie comme la clôture réflexive, symétrique et transitive de la relation $M \succ M'$:

$$\frac{}{M \simeq M} \qquad \frac{M \simeq M' \quad M' \succ M''}{M \simeq M''} \qquad \frac{M \simeq M' \quad M'' \succ M'}{M \simeq M''}$$

La β-réduction vérifie les deux propriétés suivantes

Confluence $\quad (M \succ^* M_1 \ \wedge \ M \succ^* M_2) \ \Rightarrow \ \exists M' \ (M_1 \succ^* M' \ \wedge \ M_2 \succ^* M')$

Church-Rosser $\quad M_1 \simeq M_2 \ \Leftrightarrow \ \exists M' \ (M_1 \succ^* M' \ \wedge \ M_2 \succ^* M')$

Normalisation Les notions de *forme normale*, de *faible normalisation* et de *forte normalisation* se définissent comme dans le λ-calcul pur.[1] Plus précisément, on dit que :

- Un terme M est en *forme normale* s'il n'existe aucun terme M' vers lequel M se réduit en une étape. De manière équivalente, M est en forme normale si pour tout M' tel que $M \succ^* M'$ on a $M' \equiv M$.
- Un terme M est *faiblement normalisable* s'il existe un terme N en forme normale tel que $M \succ^* N$. Le terme N, lorsqu'il existe, est unique — c'est une conséquence de la confluence de la réduction — et on l'appelle la *forme normale* de M.

Le fait qu'un terme M admette une forme normale N ne signifie pas nécessairement que toute stratégie de réduction appliquée à M aboutisse à cette forme normale.[2] C'est pourquoi on considère également une forme forte de la notion de normalisation en disant que :

- Un terme M est *fortement normalisable* s'il n'existe aucune séquence de réduction infinie $M \succ M_1 \succ M_2 \succ \cdots$ issue du terme M.

3 Le système de types

Les contextes de typage sont définis comme des suites finies de déclarations de la forme $x : \tau$, où x est une variable et τ un type :

Contextes $\qquad \Gamma \quad ::= \quad x_1 : \tau_1, \ \ldots, \ x_n : \tau_n$

(avec la contrainte que $x_i \not\equiv x_j$ si $i \neq j$). L'ensemble des variables déclarées dans un contexte Γ est noté $DV(\Gamma)$, et la concaténation de deux contextes Γ

[1] L'annotation de type « $: \tau$ » ne joue en effet aucun rôle dans le processus de β-réduction ; elle n'intervient qu'au moment du typage.

[2] Par exemple, le terme $M \equiv \mathbf{KI}(\mathbf{\Delta\Delta})$ (en posant $\mathbf{I} = \lambda x{:}o\,.\,x$, $\mathbf{K} = \lambda x{:}o\,.\,\lambda y{:}o\,.\,x$ et $\mathbf{\Delta} = \lambda x{:}o\,.\,xx$) est faiblement normalisable car : $\mathbf{KI}(\mathbf{\Delta\Delta}) \succ (\lambda y{:}o\,.\,\mathbf{I})(\mathbf{\Delta\Delta}) \succ \mathbf{I}$. En revanche, une stratégie de réduction privilégiant systématiquement la réduction de l'argument $\mathbf{\Delta\Delta}$ — typiquement une stratégie d'appel par valeur — fait entrer le processus de réduction en boucle infinie : $\mathbf{KI}(\mathbf{\Delta\Delta}) \succ \mathbf{KI}(\mathbf{\Delta\Delta}) \succ \mathbf{KI}(\mathbf{\Delta\Delta}) \succ \cdots$ Bien entendu, ce contre-exemple est mal typé ; il ne peut en être autrement puisque tous les termes bien typés dans le λ-calcul simplement typé sont fortement normalisables.

et Δ s'écrit Γ, Δ (cette écriture n'est définie que si $DV(\Gamma) \cap DV(\Delta) = \varnothing$). Enfin on dit qu'un contexte Γ est un *sous-contexte* d'un autre contexte Γ' et on écrit $\Gamma \subset \Gamma'$ si toute déclaration figurant dans Γ figure également dans Γ' (pas nécessairement dans le même ordre).

Le système de types repose sur un unique jugement de typage, noté $\Gamma \vdash M : \tau$ (« dans le contexte Γ, le terme M a pour type τ »). Ce jugement se dérive à partir des trois règles suivantes :

$$\frac{}{\Gamma \vdash x : \tau}\ {\scriptstyle (x:\tau)\in\Gamma} \qquad \frac{\Gamma,\ x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x : \tau \,.\, M : \tau \to \sigma}$$

$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}$$

On notera qu'il y a exactement une règle par construction syntaxique ; aussi dit-on que le système est *dirigé par la syntaxe*. En pratique, cela signifie que la dérivation d'un jugement $\Gamma \vdash M : \tau$ — lorsqu'elle existe — est unique, et que l'enchaînement (et l'ordre d'application) des règles d'inférence se déduit immédiatement de la structure du terme M.

Les propriétés du typage sont les suivantes :

Déclaration des variabes libres Si $\Gamma \vdash M : \tau$, alors $FV(M) \subset DV(\Gamma)$.

Affaiblissement Si $\Gamma \vdash M : \tau$ et $\Gamma \subset \Gamma'$, alors $\Gamma' \vdash M : \tau$.

Substitutivité Si $\Gamma,\ x : \tau \vdash M : \sigma$ et $\Gamma \vdash N : \tau$, alors $\Gamma \vdash M\{x := N\} : \sigma$.

Préservation du type par réduction
Si $\Gamma \vdash M : \tau$ et $M \succ^* M'$, alors $\Gamma \vdash M' : \tau$.

De plus, la présence de l'annotation de type « $: \tau$ » dans l'abstraction entraîne que le type d'un terme, lorsqu'il existe, est unique (dans un contexte donné) :

Unicité du type Si $\Gamma \vdash M : \tau$ et $\Gamma \vdash M : \tau'$, alors $\tau \equiv \tau'$.

Lorsqu'on passe à une présentation à la Curry — en supprimant l'annotation de type sous l'abstraction — cette propriété d'unicité du type disparaît. (Mais les autres propriétés sont conservées.)

Dans le λ-calcul simplement typé à la Church, les problèmes de la vérification et de l'inférence de type sont tous les deux décidables :

Vérification de type
Étant donnés Γ, M et τ, vérifier si le jugement $\Gamma \vdash M : \tau$ est dérivable.

Inférence de type
Étant donnés Γ et M, trouver un type τ tel que le jugement $\Gamma \vdash M : \tau$ est dérivable, ou échouer s'il n'existe pas de tel type.

Il est important de noter que ces deux problèmes restent décidables dans le λ-calcul simplement typé à la Curry, mais que la perte d'information liée à la disparition de l'annotation de type sous l'abstraction nécessite de recourir à des algorithmes d'inférence et de vérification beaucoup plus sophistiqués, lesquels sont basés sur la notion d'unification.

Enfin, le λ-calcul simplement typé vérifie le théorème de normalisation forte

Normalisation forte Tous les termes bien typés du λ-calcul simplement typé sont fortement normalisables.

qui exprime qu'un terme bien typé dans ce calcul a non seulement une forme normale, mais également que toute stratégie de réduction appliquée à ce terme mène invariablement à la forme normale.

4 Extensions

Il est naturel d'envisager l'extension du λ-calcul simplement typé avec diverses constructions, telles que les couples, les variantes, les booléens ou les entiers.

4.1 Produit cartésien

La syntaxe est étendue avec les constructions suivantes :

$$\begin{array}{lrcl} \textbf{Types} & \tau, \sigma & ::= & \cdots \mid \tau \times \sigma \\ \textbf{Termes} & M, N & ::= & \cdots \mid \langle M, N \rangle \mid \pi_1(M) \mid \pi_2(M) \end{array}$$

On étend la relation de réduction en une étape (notée $M \succ M'$) en ajoutant deux nouvelles règles de base :

$$\pi_1(\langle M, N \rangle) \succ M \qquad \text{et} \qquad \pi_2(\langle M, N \rangle) \succ N$$

Cette extension de la relation de réduction en une étape modifie en conséquence la relation de réduction $M \succ^* M'$ (qui reste **confluente**) et la relation de conversion $M_1 \simeq M_2$ (qui satisfait toujours la propriété de **Church-Rosser**). Enfin, on étend le système de types en ajoutant trois nouvelles règles de typage :

$$\frac{\Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash \langle M, N \rangle : \tau \times \sigma} \qquad \frac{\Gamma \vdash M : \tau \times \sigma}{\Gamma \vdash \pi_1(M) : \tau} \qquad \frac{\Gamma \vdash M : \tau \times \sigma}{\Gamma \vdash \pi_2(M) : \sigma}$$

Le système ainsi obtenu est encore dirigé par la syntaxe. Il conserve en outre toutes les propriétés du λ-calcul simplement typé, à savoir les propriétés de **déclaration des variables libres**, d'**affaiblissement**, de **substitutivité**, de **préservation du type par réduction** et même la propriété d'**unicité du type**. La **vérification** et l'**inférence de type** restent décidables dans ce système. Enfin, ce système conserve la propriété de **normalisation forte** (pour les règles de réduction étendues).

4.2 Somme directe

La syntaxe est étendue avec les constructions suivantes :

$$\begin{array}{lrcl} \textbf{Types} & \tau, \sigma & ::= & \cdots \mid \tau + \sigma \\ \textbf{Termes} & M, N & ::= & \cdots \mid i_1^{\tau+\sigma}(M) \mid i_2^{\tau+\sigma}(M) \\ & & \mid & \texttt{case}\ N\ \texttt{of}\ \{i_1(x_1) \mapsto M_1 \mid i_2(x_2) \mapsto M_2\} \end{array}$$

On étend la relation de réduction en une étape avec les règles :

$$\begin{array}{lcl} \texttt{case}\ i_1^{\tau+\sigma}(N)\ \texttt{of}\ \{i_1(x_1) \mapsto M_1 \mid i_2(x_2) \mapsto M_2\} & \succ & M_1\{x_1 := N\} \\ \texttt{case}\ i_2^{\tau+\sigma}(N)\ \texttt{of}\ \{i_1(x_1) \mapsto M_1 \mid i_2(x_2) \mapsto M_2\} & \succ & M_2\{x_2 := N\} \end{array}$$

Enfin, on étend le système de types en ajoutant trois règles de typage :

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash i_1^{\tau+\sigma}(M) : \tau + \sigma} \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash i_2^{\tau+\sigma}(M) : \tau + \sigma}$$

$$\frac{\Gamma \vdash N : \tau_1 + \tau_2 \qquad \Gamma,\ x_1 : \tau_1 \vdash M_1 : \sigma \qquad \Gamma,\ x_2 : \tau_2 \vdash M_2 : \sigma}{\Gamma \vdash \texttt{case}\ N\ \texttt{of}\ \{i_1(x_1) \mapsto M_1 \mid i_2(x_2) \mapsto M_2\} : \sigma}$$

Cette extension conserve toutes les propriétés précédentes, de la **confluence** à la **normalisation forte** en passant par la décidabilité de l'**inférence** et de la **vérification** de type. On notera que la propriété d'**unicité du type** est également conservée en raison de la présence de l'annotation de type « $\tau + \sigma$ » sur les constructions $i_1^{\tau+\sigma}(M)$ et $i_2^{\tau+\sigma}(M)$.

Dans le cadre d'une présentation à la Curry[3] évidemment, cette annotation de type disparaît, et avec elle la propriété d'unicité du type.

4.3 Type unité

On étend le système avec les constructions syntaxiques suivantes

Types	τ, σ	$::=$	$\cdots$	$\mid$	Unit
Termes	M, N	$::=$	$\cdots$	$\mid$	$()$

et la règle de typage

$$\frac{}{\Gamma \vdash () : \mathsf{Unit}}$$

(Il n'y pas de règle de réduction associée.)
On notera que les types somme et unité permettent de reconstruire les booléens :

$$\begin{array}{rcl} \mathsf{Bool} & \equiv & \mathsf{Unit} + \mathsf{Unit} \\ \mathsf{true} & \equiv & i_1^{\mathsf{Bool}}(()) \\ \mathsf{false} & \equiv & i_2^{\mathsf{Bool}}(()) \\ \texttt{if}\ N\ \texttt{then}\ M_1\ \texttt{else}\ M_2 & \equiv & \texttt{case}\ N\ \texttt{of}\ \{i_1(\_) \mapsto M_1 \mid i_2(\_) \mapsto M_2\} \end{array}$$

4.4 Type vide

On étend le système avec les constructions syntaxiques suivantes

Types	τ, σ	$::=$	$\cdots$	$\mid$	Empty
Termes	M, N	$::=$	$\cdots$	$\mid$	$\mathsf{raise}^{\tau}(M)$

et la règle de typage

$$\frac{\Gamma \vdash M : \mathsf{Empty}}{\Gamma \vdash \mathsf{raise}^{\tau}(M) : \tau}$$

(Il n'y pas de règle de réduction associée.)

[3] *cf* le langage Caml, où les « variants » s'écrivent simplement $i_1(M)$ et $i_2(M)$.
(Et où par ailleurs la construction `case` $\cdots$ `of` $\cdots$ s'écrit `match` $\cdots$ `with` $\cdots$).

4.5 Entiers naturels

On étend le système avec les constructions syntaxiques suivantes

Types	τ, σ	$::=$	$\cdots$	$\mid$	Nat			
Termes	M, N	$::=$	$\cdots$	$\mid$	$\mathsf{0}$	$\mid$ S	$\mid$	rec^τ

les règles de réduction

$$\begin{array}{lcl} \mathsf{rec}^\tau\ M_0\ M_1\ \mathsf{0} & \succ_\delta & M_0 \\ \mathsf{rec}^\tau\ M_0\ M_1\ (\mathsf{S}\ N) & \succ_\delta & M_1\ N\ (\mathsf{rec}^\tau\ M_0\ M_1\ N) \end{array}$$

et les règles de typage

$$\frac{}{\Gamma \vdash \mathsf{0} : \mathsf{Nat}} \qquad \frac{}{\Gamma \vdash \mathsf{S} : \mathsf{Nat} \to \mathsf{Nat}}$$

$$\frac{}{\Gamma \vdash \mathsf{rec}^\tau : \tau \to (\mathsf{Nat} \to \tau \to \tau) \to \mathsf{Nat} \to \tau}$$

(Dans une présentation à la Curry, rec^τ s'écrit rec.)

Cette extension (qui préserve les propriétés attendues) accroît considérablement l'expressivité calculatoire du système, et permet d'exprimer toutes les fonctions primitives récursives — et d'autres comme la fonction d'Ackermann.

4.6 Système T

On appelle *système T* (de Gödel) le système construit en ajoutant au λ-calcul simplement typé les types produit, les types somme, le type unité, le type vide et le type des entiers naturels comme indiqué précédemment.

Ce système — dans lequel la réduction est **confluente** et satisfait la propriété de **Church-Rosser** — vérifie les propriétés de **déclaration des variables libres**, d'**affaiblissement**, de **substitutivité**, de **préservation du type par réduction** et même d'**unicité du type** (sauf dans la présentation à la Curry). Dans ce système, l'**inférence** et la **vérification de type** sont décidables.

Enfin, le système T vérifie la propriété de **normalisation forte**, qui entraîne que tous les programmes bien typés dans ce formalisme terminent (et cela quelle que soit la stratégie de réduction utilisée).

Une autre propriété essentielle du calcul est que la forme normale N d'un terme M de type τ *dans le contexte vide* a exactement la forme attendue par son type τ, à savoir :
– Si $\tau \equiv \tau_1 \to \tau_2$, alors $N \equiv \lambda x : \tau_1 \,.\, N_2$.
– Si $\tau \equiv \tau_1 \times \tau_2$, alors $N \equiv \langle N_1 ; N_2 \rangle$
– Si $\tau \equiv \tau_1 + \tau_2$, alors soit $N \equiv i_1^{\tau_1+\tau_2}(N_1)$, soit $N \equiv i_2^{\tau_1+\tau_2}(N_2)$.
– Si $\tau \equiv \mathsf{Unit}$, alors $N \equiv ()$.
– Le cas $\tau \equiv \mathsf{Empty}$ est impossible.
– Si $\tau \equiv \mathsf{Nat}$, alors $N \equiv \underbrace{\mathsf{S}\ (\cdots (\mathsf{S}}_{n}\ \mathsf{0}) \cdots)$ pour un certain entier $n \geq 0$.

En particulier, le théorème de normalisation forte entraîne que *dans le contexte vide*, il n'est pas possible d'écrire un programme de type Empty.