TD: le λ-calcul comme modèle de calcul

Lionel Vaux Auclair

M2 IMD, Logique et automates

1 Entiers de Church

On a déjà vu les termes $\underline{0} = \lambda f.\lambda x.x$, $\underline{1} = \lambda f.\lambda x.f\,x$ et $\underline{2} = \lambda f.\lambda x.f\,(f\,x)$, et on a établi que $\underline{2}\,\underline{2} \to_\beta^* = \lambda f.\lambda x.f\,(f\,(f\,(f\,x))) = \underline{4}$. On peut généraliser :

Définition. *Pour tous termes u et v, on définit l'application itérée de u à v par récurrence :*

$$u^0\,v \coloneqq v$$
$$u^{n+1}\,v \coloneqq u\,(u^n\,v) \quad .$$

*Alors pour tout $n \in \boldsymbol{N}$, on note $\underline{n} \coloneqq \lambda f.\lambda x.f^n\,x$: l'*entier de Church *numéro n.*

Notez qu'on a en particulier $u^m\,(u^n\,v) = u^{m+n}\,v$. On peut alors représenter certaines fonctions sur les entiers par des λ-termes :

Exercice 1. On pose $\underline{\mathsf{succ}} \coloneqq \lambda a.\lambda f.\lambda x.f\,(a\,f\,x)$ et $\underline{\mathsf{add}} \coloneqq \lambda a.\lambda b.\lambda f.\lambda x.a\,f\,(b\,f\,x)$. Démontrez que pour tous $m, n \in \mathbf{N}$, on a $\underline{\mathsf{succ}}\,\underline{n} =_\beta \underline{n+1}$ et $\underline{\mathsf{add}}\,\underline{m}\,\underline{n} =_\beta \underline{m+n}$.

Ainsi $\underline{\mathsf{succ}}$ représente la fonction successeur et $\underline{\mathsf{add}}$ représente l'addition. On peut formaliser cette idée :

Définition. *On dit que le terme s* représente *la fonction $f : \boldsymbol{N}^k \to \boldsymbol{N}$ si, pour tous $(n_1, \dots, n_k) \in \boldsymbol{N}^k$, $s\,\underline{n_1} \cdots \underline{n_k} =_\beta \underline{f(n_1, \dots, n_k)}$.* [1]

Exercice 2. On peut vérifier que plusieurs termes peuvent représenter la même fonction :

1. Démontrez que $\underline{\mathsf{succ}'} =_\beta \lambda a.\lambda f.\lambda x.a\,f\,(f\,x)$ représente aussi la fonction successeur.
2. Démontrez que $I = \lambda a.a$ et $\underline{\mathsf{id}} \coloneqq \lambda a.a\,\underline{\mathsf{succ}}\,\underline{0}$ représentent tous les deux la fonction identité. [2]
3. Démontrez que $\underline{\mathsf{add}'} \coloneqq \lambda a.\lambda b.a\,\underline{\mathsf{succ}}\,b$ représente également l'addition. [3]

De manière générale, puisque les entiers de Church sont les fonctionnelles d'itération, il est assez facile de représenter les fonctions qu'on sait définir par récurrence. Par exemple, la multiplication peut être définie en itérant l'addition : $0 \times n = 0$ et $(m+1) \times n = n + m \times n$; autrement dit on obtient $m \times n$ à partir de 0 en appliquant m fois la fonction qui ajoute n. De même, l'exponentiation est définie en itérant la multiplication : $n^0 = 1$ et $n^{m+1} = n \times n^m$; autrement dit on obtient n^m à partir de 1 en appliquant m fois la fonction qui multiplie par n.

1. Notez qu'on ne demande pas que le terme s représentant f soit en forme normale, ni même normalisable.

2. *Indice :* On montre $\underline{\mathsf{id}}\,\underline{n} =_\beta \underline{n}$ par récurrence sur n : on obtient n en appliquant n fois le successeur à 0.

3. *Indice :* On montre $\underline{\mathsf{add}}\,\underline{m}\,\underline{n} =_\beta \underline{m+n}$ par récurrence sur m : on obtient $m+n$ en appliquant m fois le successeur à n.

Exercice 3. On définit $\underline{\mathsf{mult}} \coloneqq \lambda a.\lambda b.a\,(\underline{\mathsf{add}}\,b)\,\underline{0}$.

1. Montrez que $\underline{\mathsf{mult}}$ représente la multiplication. [4]
2. Définissez un terme $\underline{\mathsf{exp}}$ représentant l'exponentiation en itérant la multiplication.

2 Booléens et tests

On peut également représenter une information binaire par un λ-terme : on pose $\underline{\mathsf{vrai}} \coloneqq \lambda x.\lambda y.x$ et $\underline{\mathsf{faux}} \coloneqq \lambda x.\lambda y.y$.

Exercice 4. Les valeurs booléennes fonctionnent comme des instructions conditionnelles :

1. Donnez un terme $\underline{\mathsf{test}}$ tel que $\underline{\mathsf{test}}\,\underline{\mathsf{vrai}}\,u\,v =_\beta u$ et $\underline{\mathsf{test}}\,\underline{\mathsf{faux}}\,u\,v =_\beta v$, pour tous λ-termes u et v. [5]
2. Donnez des λ-termes qui calculent les opérations booléennes usuelles : $\underline{\mathsf{et}}$, $\underline{\mathsf{ou}}$, $\underline{\mathsf{non}}$ (par exemple $\underline{\mathsf{ou}}\,\underline{\mathsf{vrai}}\,\underline{\mathsf{faux}} =_\beta \underline{\mathsf{vrai}}$).

Ainsi, on peut « programmer » des tests en λ-calcul, par exemple sur les entiers :

Exercice 5.

1. Donnez un terme $\underline{\mathsf{pair}}$ tel que $\underline{\mathsf{pair}}\,\underline{2n} =_\beta \underline{\mathsf{vrai}}$ et $\underline{\mathsf{pair}}\,\underline{2n+1} =_\beta \underline{\mathsf{faux}}$ pour tout $n \in \mathbf{N}$. [6]
2. Donnez un terme $\underline{\mathsf{nul}}$ tel que $\underline{\mathsf{nul}}\,\underline{0} =_\beta \underline{\mathsf{vrai}}$ et $\underline{\mathsf{nul}}\,\underline{n+1} =_\beta \underline{\mathsf{faux}}$ pour tout $n \in \mathbf{N}$. [7]

3 Couples

Il peut parfois être utile de rassembler plusieurs informations dans une seule donnée : typiquement, un couple est la donnée de deux éléments. Pour tous termes $s, t \in \Lambda$, on note $\langle s, t\rangle \coloneqq \lambda z.z\,s\,t$ (en choisissant $z \notin \mathsf{VL}(s,t)$).

Exercice 6. Donnez des termes $\underline{\mathsf{g}}$ et $\underline{\mathsf{d}}$ tels que $\underline{\mathsf{g}}\,\langle s,t\rangle \to_\beta^* s$ et $\underline{\mathsf{d}}\,\langle s,t\rangle \to_\beta^* t$.

Grâce à cette structure, on peut itérer des fonctions qui agissent sur plusieurs données :

Exercice 7.

1. En posant $P \coloneqq \lambda x.x\left(\lambda c.\langle \underline{\mathsf{d}}\,c, \underline{\mathsf{succ}}\,(\underline{\mathsf{d}}\,c)\rangle\right)\langle \underline{0}, \underline{0}\rangle$, montrez que $P\,\underline{0} =_\beta \langle \underline{0}, \underline{0}\rangle$ et $P\,\underline{n+1} =_\beta \langle \underline{n}, \underline{n+1}\rangle$.
2. Déduisez-en un terme $\underline{\mathsf{pred}}$ qui représente la fonction prédécesseur $p : \mathbf{N} \to \mathbf{N}$ telle que $p(0) = 0$ et $p(n+1) = n$.
3. En vous inspirant de l'exemple précédent, trouvez un terme $\underline{\mathsf{fact}}$ tel que $\underline{\mathsf{fact}}\,\underline{n} \coloneqq \underline{n!}$ pour tout $n \in \mathbf{N}$. [8]

4. Notez que $\underline{\mathsf{add}}\,\underline{n}$ représente la fonction qui ajoute n.
5. La réponse à cette question peut être extrêmement simple !
6. *Indice :* Si $b \in \{\underline{\mathsf{vrai}}, \underline{\mathsf{faux}}\}$ alors $\underline{\mathsf{non}}\,(\underline{\mathsf{non}}\,b) =_\beta b$.
7. *Indice :* Pour distinguer entre 0 et $n+1$, on peut itérer une fonction constante.
8. *Indice :* Pour calculer $(n+1)! = (n+1) \times n!$ on a besoin à la fois de n et de $n!$, et donc du couple $(n, n!)$.

4 Points fixes

Jusque là on n'a représenté que des fonctions totales, mais en général les fonctions calculables sont des fonctions partielles : certains calculs peuvent boucler indéfiniment.

La contrepartie en λ-calcul de la boucle non bornée, c'est la récursion, qu'on peut représenter comme le calcul d'un point fixe :

Exercice 8. Soit $Y \coloneqq \lambda x.(\lambda y.x\,(y\,y))\,(\lambda y.x\,(y\,y))$.

1. Montrez que $Y\,s =_\beta s\,(Y\,s)$ pour tout terme s.

2. A-t-on $Y\,s \to_\beta^* s\,(Y\,s)$ en général ?

On appelle Y *l'opérateur de point fixe de Church* : en posant $t \coloneqq Y\,s$, on obtient $t =_\beta s\,t$. On peut montrer que $Y\,s$ est le plus petit point fixe de s, pour un certain ordre lié à la normalisabilité, qu'on ne détaille pas ici.

Exercice 9. Si $x \notin \mathsf{VL}(s)$, on peut considérer que $\lambda x.s$ représente une fonction constante.

1. Quelle est la forme normale de $Y\,\lambda x.s$ quand $x \notin \mathsf{VL}(s)$ et s est en forme normale ?

Par contraste, tout terme t est un point fixe de l'identité : $t =_\beta I\,t$. Donc le plus petit point fixe de I devrait intuitivement être le moins normalisable possible.

2. Montrez que $Y\,I$ n'est pas normalisable.

En dehors de ces deux cas extrêmes, on peut voir que l'opérateur de point fixe permet de définir des fonctions par récursion. Par exemple on peut définir la factorielle récursivement par :

$$n! \coloneqq \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{sinon} \end{cases}$$

et la factorielle est donc un point fixe de la fonctionnelle F qui à toute fonction $f : \mathbf{N} \to \mathbf{N}$ associe la fonction

$$F(f) : n \mapsto \begin{cases} 1 & \text{si } n = 0 \\ n \times f(n-1) & \text{sinon} \end{cases}.$$

Exercice 10. On pose le terme $F \coloneqq \lambda f.\lambda a.\underline{\mathsf{nul}}\,a\,\underline{1}\,(\underline{\mathsf{mult}}\,a\,(f\,(\underline{\mathsf{pred}}\,a)))$. Montrez que $Y\,F$ représente également la factorielle.

Ce type de construction est à la base des langages de programmation fonctionnelle de la famille ML, dont Haskell et OCaml. Et en effet, le λ-calcul est un modèle de calcul complet au même titre que les machines de Turing par exemple : c'est le sujet de la section suivante.

5 Calculabilité

Dans toute la suite, on utilise le mot *fonction* au sens de *fonction partielle*. On généralise la notion de représentation d'une fonction par un terme comme suit :

Définition. *Si f est une fonction de $\boldsymbol{N}^k$ dans $\boldsymbol{N}$, on dit que f est* représentée *par le terme s si, pour tous $(n_1, \ldots, n_k) \in \boldsymbol{N}^k$:*

— $s\,\underline{n_1}\cdots\underline{n_k} =_\beta \underline{f(n_1,\ldots,n_k)}$ *lorsque* $f(n_1,\ldots,n_k)$ *est défini ;*

— *ce terme n'a pas de forme normale de tête sinon.*

On dit qu'une fonction est λ-calculable *s'il existe un* λ*-terme qui la représente.*

On note :

— $0_k : \mathbf{N}^k \to \mathbf{N}$ la fonction constante nulle $\vec{x} \mapsto 0$;

— $\pi_{k,i} : \mathbf{N}^k \to \mathbf{N}$ la i-ème projection $(x_1,\ldots,x_k) \mapsto x_i$;

— $S : \mathbf{N} \to \mathbf{N}$ le successeur $x \mapsto x+1$.

Exercice 11. Donnez des λ-termes qui représentent 0_k, $\pi_{k,i}$ et S.

Si $f_i : \mathbf{N}^k \to \mathbf{N}$ pour $1 \le i \le n$ et $g : \mathbf{N}^n \to \mathbf{N}$, on note $g \circ (f_1,\ldots,f_n) : \mathbf{N}^k \to \mathbf{N}$ leur *composée* : $\big(g \circ (f_1,\ldots,f_n)\big)(\vec{x}) \coloneqq g(f_1(\vec{x}),\ldots,f_n(\vec{x}))$, quand chaque f_i est définie en $\vec{x}$ et g est définie en $(f_1(\vec{x}),\ldots,f_n(\vec{x}))$.

Exercice 12. Montrez que la composée de fonctions λ-calculables est λ-calculable. [9]

Si $f : \mathbf{N}^{k+1} \to \mathbf{N}$, on définit $\mu_k f : \mathbf{N}^k \to \mathbf{N}$ par : $\mu_k f(\vec{x}) = y$ si et seulement si $f(\vec{x},y) = 0$ et, pour tout $y' < y$, $f(\vec{x},y') > 0$ (et $\mu_k f(\vec{x})$ est non définie dans tous les autres cas). On dit que μ_k est l'opérateur de *minimisation*. C'est l'analogue d'une boucle en algorithmique : on incrémente y **tant que** $f(\vec{x},y) > 0$. On a donc :

$$\mu_k f(\vec{x}) = \begin{cases} 0 & \text{si } f(\vec{x},0) = 0 \\ 1 + \mu_k g(\vec{x}) & \text{avec } g(\vec{x},y) = f(\vec{x},y+1) \text{ sinon} \end{cases}.$$

Notez que si $f : \mathbf{N} \to \mathbf{N}$ alors $\mu_0 f : \mathbf{N}^0 \to \mathbf{N}$ est une fonction constante, qu'on identifie avec sa valeur $\mu_0 f \in \mathbf{N}$.

Exercice 13. On pose

$$M \coloneqq \lambda\phi.\lambda f.\underline{\mathsf{nul}}\,(f\,\underline{0})\,\underline{0}\left(\underline{\mathsf{succ}}\left(\Phi\,(\lambda x.f\,(\underline{\mathsf{succ}}\,x))\right)\right)$$

(en supposant $x \notin \mathsf{VL}(s)$).

1. Vérifiez que si s représente $f : \mathbf{N} \to \mathbf{N}$ et $\mu_0 f$ est défini alors $Y\,M\,s =_\beta \underline{\mu_0 f}$.
On admet dans la suite que $Y\,M\,s$ n'est pas normalisable de tête quand $\mu_0 f$ n'est pas défini. [10]

2. Déduisez en, pour tout $k \in \mathbf{N}$, un terme $\underline{\mathsf{min}_k}$ tel que si s représente $f : \mathbf{N}^{k+1} \to \mathbf{N}$ alors $\underline{\mathsf{min}_k}\,s$ représente $\mu_k f$.

L'ensemble des *fonctions* μ*-récursives* (ou plus simplement, *récursives*) est inductivement défini par les règles suivantes :

— S et toutes les fonctions 0_k et π_j^k sont des fonctions récursives ;

— la composée de fonctions récursives est récursive ;

— les opérateurs μ_k préservent les fonctions récursives.

9. *Indice :* Pour des fonctions totales, c'est facile. Mais il faut prendre en compte le cas où l'un des f_i n'est pas défini en $\vec{x}$. Pour ça, on peut utiliser l'astuce suivante : montrez que si $f : \mathbf{N} \to \mathbf{N}$ est représentée par s, alors, pour tout terme t, $s\,\underline{n}\,I\,t$ se réduit en t si f est définie en n, et n'a pas de forme normale de tête sinon.

10. Vous pouvez tenter de vous en convaincre après avoir traité la question 1, mais les détails techniques sont un peu subtils : il faut par regarder à quoi ressemble la suite des réductions de tête issues de $Y\,M s$.

Un résultat bien connu en calculabilité est que :

— les fonctions μ-récursives ;

— les fonctions calculables par des machines de Turing ;

— les fonctions λ-calculables ;

sont différentes définitions de la même notion. On vient d'en montrer un bout : les fonctions μ-récursives sont λ-calculables.