μ-recursive $\Longrightarrow$ λ-définissable

Leçons : 912 ; 929

Références : LASSAIGNE, DE ROUGEMONT, *Logique et fondements de l'informatique* (p.193) [?] et LESESVRE–MONTAGNON–LE BARBENCHON–PIERRON, *131 développements pour l'oral* (p. 696) [?]

Prérequis :

- la définition de fonctions μ-récursives
- combinateurs de point fixe (Turing [1], Curry [2] par exemple) [3]

Introduction :

On va montrer que toute fonction μ-récursive est λ-définissable, c'est un étape pour valider la thèse de Church qui nous dit que toute fonction calculable par une procédure effective est en fait une fonction calculable par une machine de Turing. En effet, on va pouvoir remplacer machine de Turing par λ-calcul ou fonctions μ-récursives quand on aura prouver que ces trois modèles de calcul sont équivalents. On prouve ici une implication de cette équivalence.

On représentera les entiers par [4]

$$[\![0]\!] := \lambda x.x$$
$$[\![n+1]\!] := \langle F, [\![n]\!] \rangle$$

avec

$$F := \lambda xy.y$$
$$\langle M, N \rangle := \lambda x.xMN$$

Définition 1. On dit qu'une fonction $f : \mathbb{N}^p \to \mathbb{N}$ est λ-définissable si et seulement si il existe un terme F de λ-calcul tel que

$$\forall n_1, \dots, n_p \in \mathbb{N}, \quad F[\![n_1]\!][\![n_2]\!] \dots [\![n_p]\!] = [\![f(n_1, \dots, n_p)]\!]$$

On dira que F représente f.

Théorème 1. Toute fonction μ-récursive est λ-définissable.

Démonstration.
Étape 1 : *Les fonctions de base* [5] *sont λ-définissables.*

La fonction zero est représentée par

$$\lambda x.[\![0]\!]$$

La fonction succ est représentée par

$$\lambda x.\langle F, x \rangle$$

La fonction π_i^n est représentée par

$$\lambda x_1 x_2 \dots x_n.x_i$$

1. $\Theta := AA$ où $A := \lambda xf.f(xxf)$
2. $Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
3. L'avantage du point fixe de Turing sur celui de Curry, c'est qu'on a une β-réduction directe, alors que pour celui de Curry, il faut réduire dans des sens différents (c'est une β-équivalence)
4. c'est une définition que l'on trouve dans le Barendregt [**?**]
5. La fonction nulle zero, la fonction qui donne le successeur succ et la projection sur la $i^{\text{ème}}$ coordonnée d'un vecteur à n éléments π_i^n.

Étape 2 : *Clos par composition*

Si f une fonction à p arguments est représentée par un terme F et les p fonctions $g_1, \dots, g_p$ sont représentées par $G_1, \dots, G_p$, alors

$$\begin{aligned}[\![f(g_1(\vec{n}), \dots, g_p(\vec{n}))]\!] &= F[\![g_1(\vec{n})]\!] \dots [\![g_p(\vec{n})]\!] \\ &= F(G_1[\![\vec{n}]\!]) \dots (G_p[\![\vec{n}]\!])\end{aligned}$$

Donc

$$\lambda\vec{x}.F(G_1\vec{x}) \dots (G_p\vec{x})$$

représente la composition de f avec les fonctions g_i.

Étape 3 : *Clos par récursion primitive*

Soit φ le schéma de récursion primitive :

$$\varphi(\vec{n}, 0) = f(\vec{n})$$

$$\varphi(\vec{n}, k) = g(\vec{n}, k, \varphi(\vec{n}, k-1)$$

On se donne F qui représente f et G qui représente g.

On veut montrer que φ est λ-définissable.

On voudrait trouver un terme H tel que

$$H\ \vec{x}\ k = \ \mathsf{if}\ (\mathsf{is\_zero}\ k)\ \mathsf{then}\ F\ \vec{x}\ \mathsf{else}\ G\ \vec{x}\ k\ (H\ \vec{x}\ (\mathsf{pred}\ k))$$

avec

$$\mathsf{if.then.else} = \lambda xtf.xtf$$

$$\mathsf{is\_zero} = \lambda x.(xT)$$

$$\mathsf{pred} = \lambda x.(xF)$$

On va donc avoir besoin d'un combinateur de point fixe θ tel que

$$\forall K, \quad \theta K = K(\theta K)$$

La fonction φ est donc représentée par

$$\theta\ H$$

avec

$$H := (\lambda h\vec{x}k.\ \mathsf{if}\ (\mathsf{is\_zero}\ k)\ \mathsf{then}\ F\ \vec{x}\ \mathsf{else}\ G\ \vec{x}\ k\ (h\ \vec{x}\ (\mathsf{pred}\ k))$$

En effet

$$\begin{aligned}[\![\varphi(\vec{n}, 0)]\!] &= \theta\ H\ [\![\vec{n}]\!]\ [\![0]\!] \\ &= H\ (\theta\ H)\ [\![\vec{n}]\!]\ [\![0]\!] \\ &= F\ [\![\vec{n}]\!] \\ &= [\![f(\vec{n})]\!]\end{aligned}$$

et pour tout $k > 0$,

$$\begin{aligned}[\![\varphi(\vec{n}, k)]\!] &= \theta\ H\ [\![\vec{n}]\!]\ [\![k]\!] \\ &= H\ (\theta\ H)\ [\![\vec{n}]\!]\ [\![k]\!] \\ &= G\ [\![\vec{n}]\!]\ [\![k]\!]\ (\theta\ H\ [\![\vec{n}]\!]\ (\mathsf{pred}\ [\![k]\!])) \\ &= G\ [\![\vec{n}]\!]\ [\![k]\!]\ (\theta\ H\ [\![\vec{n}]\!]\ [\![k-1]\!]) \\ &= G\ [\![\vec{n}]\!]\ [\![k]\!]\ [\![\varphi(\vec{n}, k-1)]\!] \\ &= [\![g(\vec{n}, k, \varphi(\vec{n}, k-1))]\!]\end{aligned}$$

Étape 4 : *Clos par minimisation non bornée*

Soit $\varphi(\vec{n}) = \mu_i(f(\vec{n}, i) = 0)$ le schéma de minimisation non bornée. On se donne F qui représente f.

Montrons que φ est λ-définissable.

Il faut trouver un terme qui fait un test sur le prédicat $f(\vec{n}, i) = 0$ et qui renvoit i si le prédicat est évalué à vrai et sinon teste sur $i + 1$, etc.

Soit P un prédicat, on pose

$$H_P = \lambda hk.\ \mathsf{if}\ (P\ k)\ \mathsf{then}\ k\ \mathsf{else}\ h\ (\mathsf{succ}\ k))$$

On utilise à nouveau un combinateur de point fixe

$$\begin{aligned} \theta\ H_P\ [\![0]\!] &= H_P\ (\theta\ H_P)\ [\![0]\!] \\ &= \mathsf{if}\ (P\ [\![0]\!])\ \mathsf{then}\ [\![0]\!]\ \mathsf{else}\ \theta\ H_P\ (\mathsf{succ}\ [\![0]\!]) \\ &= \mathsf{if}\ (P\ [\![0]\!])\ \mathsf{then}\ [\![0]\!]\ \mathsf{else}\ \theta\ H_P\ [\![1]\!] \end{aligned}$$

On pose

$$\mu\ P = \theta\ H_P\ [\![0]\!]$$

Ainsi on peut représenter φ par le terme

$$\lambda\vec{x}.\Big(\mu\big(\lambda k.(\mathsf{is\_zero}\ (F\ \vec{x}\ k))\big)\Big)$$

□

Remarques :

Ce théorème est une implication dans l'équivalence des modèles de calcul entre Machines de Turing, fonctions μ-récursives et λ-calcul. En effet, combiné avec le développement Machine de Turing $\Longrightarrow$ μ-recursif, il suffit de prouver que toute fonction λ-définissable est calculable par une Machine de Turing.

Astuces de l'agrégatif :

Il faut bien comprendre le sens des termes notamment pour la minimisation non bornée qui n'est pas forcément la partie la plus immédiate.

Il faut voir les λ-termes T et F comme les booléens true et false. En effet, pour la conditionnelle $\mathsf{if\ \ then\ else}$: $\lambda btf.btf$, le booléen b qui sera soit T soit F renverra soit t soit f puisque T renvoie le premier argument et F le deuxième argument.

Le combinateur de point fixe permet de simuler une boucle while, car il permet de réitérer une fonction sans connaitre le nombre d'itérations. On a utilisé un combinateur de point fixe pour simuler la récursion primitive mais l'expressivité du combinateur de point fixe est plus grande puisqu'elle peut simuler la minimisation non bornée. On peut prendre comme combinateur de point fixe celui de Curry Y ou celui de Turing Θ avec

$$Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \quad \text{et} \quad \Theta := AA \text{ où } A := \lambda xf.f(xxf)$$

par exemple. Un des avantages du point fixe de Turing réside dans le fait qu'on a une β-réduction directe, alors que pour celui de Curry, il faut réduire dans des sens différents (c'est une β-équivalence). L'avantage de celui de Curry est qu'il est plus naturel à définir. En effet,

$$\begin{aligned} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))K &\to_\beta (\lambda x.K(xx))(\lambda x.K(xx)) \\ &\to_\beta K((\lambda x.K(xx))(\lambda x.K(xx))) \\ &\leftarrow_\beta K(\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))K) \end{aligned}$$

Questions possibles :

- Montrer que à $\vec{x}$ fixé, $\lambda k.(\mathsf{is\_zero}\ (X\ [\![\vec{x}]\!]\ [\![k]\!]))$ représente $\chi(\vec{x}, k) = 0$.
- Donner un λ-terme qui additionne deux entiers.
- Donner une autre représentation des entiers. entiers de Church
- Montrer que toute fonction λ-définissable est Turing-calculable.
- Montrer que les combinateurs de Curry et de Turing sont bien des combinateurs de point fixe.