Domaines: introduction

La *non-terminaison* est inhérente au calcul, et tous les langages de programmation (Turing-complets) permettent de définir des programmes qui bouclent.

En théorie de la récursivité, on étudie les fonctions *partielles* de $\mathbb{N}$ dans $\mathbb{N}$.

En théorie des domaines, on *réifie* la non terminaison: $\perp$ dénote un calcul qui ne termine pas (diverge).

Les fonctions partielles de $\mathbb{N}$ dans $\mathbb{N}$ deviennent de fonctions *totales* de $\mathbb{N}_\perp = \{\perp, 0, 1, 2, \ldots, n, \ldots\}$ dans $\mathbb{N}_\perp$.

Tous les modèles universels de calcul (Machine de Turing, λ-calcul, machines à régistres, algorithmes de Markov...) permettent de définir le même ensemble de fonctions de $\mathbb{N}_\perp$ dans $\mathbb{N}_\perp$: les *fonctions récursives partielles* (d'où la $\ll$solidité$\gg$ de la Thèse de Church).

Mais que se passe-t-il aux *types supérieurs*? Que est-ce que une fonction(nelle) calculable de type $(\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$?

Des calculs infaisables

problème 1, version Machine de Turing:

Construire une Machine de Turing M qui sur l'entrée n renvoie 0 si M_n diverge sur 0, 1 sinon.

problème 1, version λ-calcul typé:

λ-définir la fonctionnelle $\Phi : (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ suivante:

$$\Phi(f) = \begin{cases} 0 & \text{si } f(0) = \perp \\ 1 & \text{sinon} \end{cases}$$

Monotonicité du calcul

Toute fonctionnelle définissable de type $(\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ est monotone croissante par rapport aux ordres sur $\mathbb{N}_\perp$ et $\mathbb{N}_\perp \to \mathbb{N}_\perp$ définis par:

$$x \leq y \text{ si } \ (x = \perp \ \text{ ou } x = y)$$

$$f \sqsubseteq g \text{ si } \ (\forall x \in \mathbb{N}_\perp \ f(x) \leq g(x))$$

(ce dernier est l'ordre *point à point*, ou *extensionnel*, associé au premier).

Des calculs infaisables, 2

problème 2, version Machine de Turing:

Construire une Machine de Turing M qui sur l'entrée n renvoie 0 si M_n calcule la fonction identité, diverge sinon.

problème 2, version λ-calcul typé:

λ-définir la fonctionnelle $\Phi : (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ suivante:

$$\Phi(f) = \begin{cases} 0 & \text{si } \forall x \in \mathbb{N}_\perp \ f(x) = x \\ \perp & \text{sinon} \end{cases}$$

Remarque: Φ est monotone.

Continuité du calcul

Deux formulations équivalentes:

1. Si Φ est definissable et $\Phi(f) \neq \perp$, alors il existe une fonction *à graphe fini* $f_0 \sqsubseteq f$ telle que $\Phi(f_0) = \Phi(f)$.

2. Toute fonctionnelle définissable préserve les limites filtrantes par rapport a l'ordre $\sqsubseteq$.

En résumé, si Φ est définissable:

- $f \sqsubseteq g \Rightarrow \Phi(f) \leq \Phi(g)$
- $\Phi(\bigsqcup \Delta) = \bigvee_{f \in \Delta} \Phi(f)$

Modèle *continu* ou *de Scott*.

D'autres calculs infaisables, en fonctionnel pur

problème 3, version Machine de Turing:

Construire une Machine de Turing M qui sur l'entrée n renvoie 0 si

M_n renvoie 0 sur 0 ou M_n renvoie 0 sur 1,

diverge sinon.

problème 3, version λ-calcul typé:

λ-définir la fonctionnelle $\Phi : (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ suivante:

$$\Phi(f) = \begin{cases} 0 & \text{si } f(0) = 0 \text{ ou } f(1) = 0 \\ \perp & \text{sinon} \end{cases}$$

Remarque: Φ est continue.

Stabilité du calcul fonctionnel

Si Φ est definissable et $\Phi(f) \neq \perp$, alors il existe une fonction *à graphe fini* $f_0 \sqsubseteq f$ telle que

- $\Phi(f_0) = \Phi(f)$.
- Pour tout $g \sqsubseteq f$, si $\Phi(g) \neq \perp$ alors $f_0 \sqsubseteq g$.

f_0 répresente l'information (finie) *nécessaire*, au dessous de f, pour que Φ produise un résultat.

Modèle *stable.*

D'autres calculs infaisables, en fonctionnel pur (2)

problème 4, version λ-calcul typé:

λ-définir la fonctionnelle $\Phi : (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ suivante:

$$\Phi(f) = \begin{cases} 0 & \text{si } f(0) = 0 \text{ ou } f(1) = 1 \\ 0 & \text{si } f(1) = 0 \text{ ou } f(2) = 1 \\ 0 & \text{si } f(0) = 1 \text{ ou } f(2) = 0 \\ \perp & \text{sinon} \end{cases}$$

Remarque: Φ est stable.

Stabilité ≪forte≫ du calcul fonctionnel

L'information minimale f_0 relative a tout élément f tel que $\Phi(f) \neq \perp$ doit pouvoir être engendré de façon déterministe à partir du $\perp$ de l'espace $\mathbb{N}_\perp \to \mathbb{N}_\perp$.

Dans l'exemple, sur quel argument va-t-on tester f d'abord?

Modèle *fortement stable*

On n'aboutira au modèle complet pour PCF qu'à travers la Sémantique de Jeux, mais en cours de route on rencontrera des exemples significatifs du va et vient entre syntaxe et sémantique:

$$\begin{array}{ccc}
\text{langage} & & \text{modèle} \\
\\
\mathrm{PCF}^{\|} & \longleftrightarrow & \text{modèle continu} \\
\\
\mathrm{StPCF} & \longleftrightarrow & \text{modèle stable} \\
\\
\mathrm{PCF} + \mathtt{H} & \longleftrightarrow & \text{modèle fortement stable}
\end{array}$$