Récursivité

Laure Danthony

Table des matières

1 Définitions et cadre d'étude

1.1 Différents types de fonctions

DÉFINITION 1
Si on se donne $f : \mathcal{A}^* \to \mathcal{A}^*$, plusieurs cas se présentent :

- aucune MT ne calcule f : f est dite **non calculable** ;
- si $\exists i, \varphi_i$ calcule f, alors :
  - si $Dom(f) = \mathcal{A}^*$, f est dite **totale récursive** (tr)
  - si $Dom(f) \subsetneq \mathcal{A}^*$, f est dite **partielle partiellement récursive** (ppr)

1.2 Systèmes acceptables de programmation

DÉFINITION 2
Un **système acceptable de programmation** est une suite $\varphi_i, i \in \mathbb{N}$ d'algorithmes de ($\mathbb{N} \to \mathbb{N}$ ou $\mathcal{A}^* \to \mathcal{A}^*$) tels que :

- l'ensemble des fonctions calculées par les φ_i est l'ensemble des fonctions Turing-calculables ;
- il existe un algorithme universel ;
- il existe un théorème s-n-m croissant.

REMARQUE 1 Les MT forment un sap.

2 Théorème de l'arrêt et théorème de Rice

2.1 Théorème de l'arrêt

THÉORÈME 1 (HALT THEOREM) *Il n'existe pas d'algorithme φ_a tel que :*

- *φ_a s'arrête sur toute entrée $< x, y >$;*
- *φ_a s'arrête sur 1 si $\varphi_x(y)$ s'arrête ;*
- *φ_a s'arrête sur 0 si $\varphi_x(y)$ ne s'arrête pas.*

2.2 Théorème de Rice

THÉORÈME 2 (RICE THEOREM) *Soit X une partie de l'ensemble des fonctions ppr (ou tr), alors en notant $A = \{x/\varphi_x \in X\}$:*

- *soit $A = \emptyset$*
- *soit $A = \mathbb{N}$*
- *soit χ_A est non récursive*

COROLLAIRE 1 *On ne peut pas décider si deux machines de Turing calculent la même fonction.*

2.3 Théorème de Rice généralisé

DÉFINITION 3
Soit y un entier de codage $\sigma(y) = (x_1, \ldots, x_k, x_{k+1})$, la **fonction germe** engendrée par y, notée $f_y^\star$ est définie par $f_y^\star(x_1, \ldots, x_k) = x_{k+1}$ et indéfinie partout ailleurs.

THÉORÈME 3 *$P_C = \{x | \varphi_x$ calcule une fonction de $C\}$ est récursivement énumérable ssi C est l'ensemble des fonctions germes d'un ensemble récursivement énumérable.*

3 Théorèmes de la récursion de Kleene

3.1 Théorèmes de la récursion

THÉORÈME 4 *Pour toute fonction f ppr, il existe $n \in \mathbb{N}$ tel que :*

$$\varphi_n = \varphi_{f(n)}$$

REMARQUE 2 On a donc l'existence de l'appel récursif dans tout sap.

THÉORÈME 5 *Pour toute fonction f ppr, il existe une fonction g récursive totale strictement croissante telle que :*

$$\forall x, \varphi_{g(x)} = \varphi_{f\left(g(x)\right)}$$

COROLLAIRE 2 *Soit f ppr, alors $\{n/\varphi_n = \varphi_{f(n)}\}$ est infini (autrement dit, il y a autant de programmes récursifs que l'on veut).*

COROLLAIRE 3 *L'ensemble $\{(x,y)/\varphi_x = \varphi_y\}$ est coupé par le graphe de toute fonction ppr. Autrement dit :*

$$\forall a \in \mathbb{N}, \{(x,y)/\varphi_x = \varphi_y\} \cap \{(x,y)/y = \varphi_a(x)\} \neq \emptyset$$

3.2 Application

Soit $\Psi : \mathbb{N} \to \mathcal{F} = \{$ppr à une variable$\}$ telle que l'application :

$$< x, y > \mapsto \left(\Psi\ x\right)(y)$$

soit ppr (Ψ est un logiciel qui écrit des programmes).

PROPOSITION 1 $\exists a, \Psi(a) = \varphi_a$

COROLLAIRE 4 *$\exists a$ tel que φ_a calcule la fonction constante de valeur a.*

REMARQUE 3 On peut donc faire en sorte que le logiciel s'écrive lui-même.