Logique et complexité

François Le Maître

9 octobre 2017

Table des matières

Première partie
Calculabilité

Le but de cette première partie est d'arriver à une définition satisfaisante des fonctions calculables par ordinateur. Un premier candidat important est la classe des fonctions récursives primitives que voici.

1 Fonctions récursives primitives

Commençons par décrire les ensembles de départ et d'arrivée des fonctions qui nous intéressent : pour $p \in \mathbb{N}$ on note $\mathcal{F}_p$ l'ensemble des applications de $\mathbb{N}^p$ dans $\mathbb{N}$. On adopte la convention que $\mathbb{N}^0 = \{\emptyset\}$ et on ainsi $\mathcal{F}_0$ est l'ensemble des applications de $\{\emptyset\}$ dans $\mathbb{N}$ qui s'identifie naturellement à $\mathbb{N}$. La classe des fonctions récursives primitives sera un sous-ensemble de

$$\mathcal{F} := \bigcup_{p \in \mathbb{N}} \mathcal{F}_p.$$

Définition 1.0.1. Pour $p \in \mathbb{N}$ et $i \leqslant p$ on note π_p^i l'application de **projection** sur la i-ème coordonnée de $\mathbb{N}^p$ dans $\mathbb{N}$, c'est-à-dire que pour $(x_1, ..., x_p) \in \mathbb{N}^p$,

$$\pi_p^i(x_1, ..., x_p) = x_i.$$

Définition 1.0.2. La fonction **successeur** est la fonction $S : \mathbb{N} \to \mathbb{N}$ donnée par $S(x) = x + 1$.

Définition 1.0.3. Étant donnée $f_1, ..., f_n \in \mathcal{F}_p$ et $g \in \mathcal{F}_n$, la fonction **composée** $g(f_1, ..., f_n)$ est l'élément de $\mathcal{F}_p$ défini par : pour tous $x_1, ..., x_p \in \mathbb{N}$,

$$g(f_1, ..., f_n)(x_1, ..., x_p) = g(f_1(x_1, ..., x_p), ..., f_n(x_1, ..., x_p)).$$

Définition 1.0.4. Étant donnée une fonction $g \in \mathcal{F}_p$ et $h \in \mathcal{F}_{p+2}$, il existe une unique fonction $f \in \mathcal{F}_{p+1}$ telle que pour tous $x_1, ..., x_p, y \in \mathbb{N}$ on ait

$$f(x_1, ..., x_p, 0) = g(x_1, ..., x_p)$$
$$f(x_1, ..., x_p, y + 1) = h(x_1, ..., x_p, y, f(x_1, ..., x_p, y))$$

On appelle f la fonction **définie par récurrence** à partir de g et h.

L'unicité d'une telle fonction se prouve par induction sur y (exercice). L'existence est intuitivement claire mais peut en fait être prouvée à partir des axiomes de la théorie des ensembles (cf. [CL93, Chap. 7, Sec. 3]).

Nous pouvons maintenant définir les fonctions récursives primitives.

Définition 1.0.5. L'ensemble $\mathcal{F}_{rp}$ des fonctions récursives primitives est le plus petit sous-ensemble de $\mathcal{F}$ qui :

i) contient les fonctions constantes $\mathbb{N}^p \to \mathbb{N}$,
ii) contient les projections π_p^i,
iii) contient la fonction successeur S,
iv) est stable par composition,
v) est stable par définition par récurrence.

Un sous-ensemble de $\mathbb{N}^p$ est récursif primitif si sa fonction caractéristique l'est.

1.1 Exemples de base

1.1.1 Addition

L'addition $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ est récursive primitive. En effet on a :
— $x + 0 = x$ (et la projection sur la première coordonnée est bien récursive)
— $x + (y + 1) = S(x + y) = S(\pi_3^3(x, y, x + y))$ et $S \circ \pi_3^3$ est bien récursive.
Par composition, on voit alors que $(x_1, ..., x_p) \mapsto x_1 + \cdots + x_p$ est récursive primitive.

1.1.2 Multiplication

La multiplication $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ est récursive primitive. En effet on a :
— $x \times 0 = 0$
— $x \times (y + 1) = (x \times y) + y$
Par composition, on voit alors que $(x_1, ..., x_p) \mapsto x_1 \times \cdots \times x_p$ est récursive primitive.

1.1.3 Soustraction

On note $x \dot{-} y = \max(x - y, 0)$. Montrons que c'est une fonctions récursive primitive. On commence par voir que la fonction $(x, y) \mapsto x \dot{-} 1$ est récursive primitive. En effet on a la définition par récurrence sur x suivante :
— $0 \dot{-} 1 = 0$
— $x + 1 \dot{-} 1 = x$.
Ensuite, $(x, y) \mapsto x \dot{-} y$ est récursive primitive car
— $x \dot{-} 0 = x$
— $x \dot{-} (y + 1) = (x \dot{-} y) \dot{-} 1$.

1.1.4 Relation d'ordre usuelle

L'ensemble des entiers strictement positifs est récursif primitif puisque sa fonction caractéristique $\chi_{\mathbb{N}^*}$ satisfait la relation de récurrence
— $\chi_{\mathbb{N}^*}(0) = 0$
— $\chi_{\mathbb{N}^*}(x + 1) = 1$.
L'ensemble des couples (x, y) tels que $x > y$ est récursif primitif puisque $x > y \iff x \dot{-} y > 0$.

1.2 Propriétés de stabilité

Proposition 1.2.1. L'ensemble des sous-ensembles récursifs primitifs est clos par union finie, intersection finie et passage au complémentaire.

Démonstration. Remarquons d'abord que si A et B sont récursifs primitifs, alors $A \setminus B$ l'est aussi car $\chi_{A \setminus B} = \chi_A \dot{-} \chi_B$.

L'intersection de deux ensembles A et B récursifs primitifs est également récursive primitive car $\chi_{A \cap B} = \chi_A \chi_B$ et on a vu que la multiplication est récursive primitive.

Remarquons que $\mathbb{N}$ est récursif primitif car sa fonction caractéristique est constante.

On peut alors conclure quant à la stabilité des ensemble récursifs primitifs par union : on a $A \cup B = \mathbb{N} \setminus (\mathbb{N} \setminus A \cap \mathbb{N} \setminus B)$ et les résultats que nous venons d'établir permettent alors de conclure. □

Proposition 1.2.2. L'ensemble des fonctions récursives primitives est stable par permutation des variables : si $f : \mathbb{N}^p \to \mathbb{N}$ est récursive primitive alors pour toute application $\sigma : \{1, ..., n\} \to \{1, ..., p\}$ on a que

$$f_\sigma : (x_1, ..., x_n) \mapsto f(x_{\sigma(1)}, ..., x_{\sigma(n)})$$

est récursive primitive.

Démonstration. Il suffit de remarquer que $f_\sigma = f \circ (\pi_p^{\sigma(1)}, ..., \pi_p^{\sigma(n)})$. □

Exemple 1.2.3. L'ensemble des (x, y) tels que $x = y$ est récursif primitif car c'est l'intersection de l'ensemble des (x, y) tels que $x \leqslant y$ avec l'ensemble des (x, y) tels que $y \leqslant x$.

Proposition 1.2.4. L'ensemble des ensembles récursifs primitif est stable par préimage via des applications récursives primitives : si $f_1, ..., f_n \in \mathcal{F}_p$ sont récursives primitives et $A \subseteq \mathbb{N}^n$ est récursif primitif, alors $\{(x_1, ..., x_p) : (f_1(x_1, ..., x_p), ..., f_n(x_1, ..., x_p)) \in A\}$ est récursif primitif.

Démonstration. Sa fonction caractéristique est $\chi_A(f_1, ..., f_n)$. □

1.2.1 Définition par cas

Proposition 1.2.5. L'ensemble des fonctions récursives primitives est stable par définition par cas portant sur des ensembles récursifs primitifs.

Démonstration. Si $\mathbb{N}^p = A_1 \sqcup \cdots \sqcup A_k$ alors la fonction définie par

$$f(x) = f_i(x) \text{ si } x \in A_i$$

peut se réécrire $f = \chi_{A_1} \times f_1 + \cdots + \chi_{A_k} \times f_k$ qui est bien récursive primitive dès lors que $A_1, ..., A_k$ et $f_1, ..., f_k$ le sont. □

Exemple 1.2.6. La fonction $(x, y) \mapsto \min(x, y)$ est récursive primitive puisque

$$\min(x, y) = \begin{cases} x & \text{si } x < y \\ y & \text{si } x \geqslant y \end{cases}.$$

De même la fonction max est récursive primitive.

1.2.2 Sommes et produits partiels

Proposition 1.2.7. Si $f \in \mathcal{F}_{p+1}$ est récursive primitive, alors les fonctions g et h définies par

$$g(x_1, ..., x_p, t) = \sum_{i=0}^{t} f(x_1, ..., x_p, i)$$

$$h(x_1, ..., x_p, t) = \prod_{i=0}^{t} f(x_1, ..., x_p, i)$$

sont également récursives primities.

Démonstration. La fonction g est définie par récurrence par $g(x_1, ..., x_{p-1}, 0) = f(x_1, ..., x_{p-1}, 0)$ puis

$$g(x_1, ..., x_{p-1}, t+1) = g(x_1, ..., x_{p-1}, t) + f(x_1, ..., x_{p-1}, t+1)$$

donc g est récursive primitive. De même h est récursive primitive car définie par récurrence par $h(x_1, ..., x_p, 0) = f(x_1, ..., x_p, 0)$ puis

$$h(x_1, ..., x_{p-1}, t+1) = h(x_1, ..., x_p, t) + f(x_1, ..., x_p, t+1). \quad \square$$

Exercice. Montrer que $n \mapsto n!$ est récursive primitive.

1.2.3 Schéma μ borné

Soit $A \subseteq \mathbb{N}^{p+1}$. Considérons la fonction $g \in \mathcal{F}_{p+1}$ définie par

$$g(x_1, ..., x_p, t) = \begin{cases} \min(\{z \leqslant t : (x_1, ..., x_p, z) \in A\}) & \text{si } \exists z \leqslant t : (x_1, ..., x_p, z) \in A \\ 0 & \text{sinon.} \end{cases}$$

On dit que la fonction g est définie par schéma μ-borné à partir de A. On notera

$$g(x_1, ..., x_p, z) = \mu\, t \leqslant z : (x_1, ..., x_p, t) \in A.$$

Proposition 1.2.8. Soit $A \subseteq \mathbb{N}^{p+1}$ récursif primitif. Alors la fonction g définie par schéma μ-borné à partir de A est récursive primitive.

Démonstration. On voit que g est définie par récurrence par $g(x_1, ..., x_p, 0) = 0$ puis

$$g(x_1, ..., x_p, z+1) = \begin{cases} g(x_1, ..., x_p, z) & \text{si } \sum_{i=0}^{z} \chi_A(x_1, ..., x_p, i) \geqslant 1; \\ z+1 & \text{si } \sum_{i=0}^{z} \chi_A(x_1, ..., x_p, i) = 0 \text{ et } (x_1, ..., x_p, z+1) \in A; \\ 0 & \text{dans les autres cas.} \end{cases}$$

Ainsi g est bien récursive primitive (on a utilisé le fait que la définition par cas sur ensembles récursifs primitifs et la somme partielle préservent les fonctions récursives primitives). □

Proposition 1.2.9. La classe des ensembles récursifs primitifs est stable par quantification bornée.

Démonstration. Soit $A \subseteq \mathbb{N}^{p+1}$ récursif primitif, alors $\{(x_1, ..., x_p, z) : \forall t \leqslant z, (x_1, ..., x_p, t) \in A\}$ est récursif primitif car sa fonction caractéristique est

$$\prod_{t=0}^{z} \chi_A(x_1, ..., x_p, t).$$

L'ensemble $\{(x_1, ..., x_p, z) : \exists t \leqslant z, (x_1, ..., x_p, t) \in A\}$ est récursif primitif car c'est le complémentaire de $\{(x_1, ..., x_p, z) : \forall t \leqslant z, (x_1, ..., x_p, t) \notin A\}$. □

1.3 Nouveaux exemples

Proposition 1.3.1. La fonction $(x, y) \mapsto q(x, y)$ qui à (x, y) associe le quotient entier de x par y est récursive primitive.

Démonstration. En effet $q(x, y) = \mu\, q \leqslant x : (q+1) \times y > x$. □

Corollaire 1.3.2. La fonction qui à (x, y) associe le reste $r(x, y)$ de la division euclidienne de x par y est récursive primitive.

Démonstration. On a $r(x, y) = x - q(x, y) \times y$. □

Corollaire 1.3.3. L'ensemble des (x, y) tels que y divise x est récursif primitif.

Démonstration. C'est l'ensemble des (x, y) tels que $r(x, y) = 0$. □

Corollaire 1.3.4. L'ensemble des nombres premiers est récursif primitif.

Démonstration. On a x premier ssi $\forall y \leqslant x$, y ne divise pas x ou $y = 1$ ou $y = x$. □

Corollaire 1.3.5. La fonction $\pi(n)$ définie par $\pi(n)$ est le $n+1$-ème nombre premier est récursive primitive.

Démonstration. En effet on la définit par récurrence par $\pi(0) = 2$ puis $\pi(n+1) = \mu y \leqslant \pi(n)! + 1 : y$ est premier et $y > \pi(n)$. □

1.4 Deux codages

Dans cette section, on note quelques bijections utiles permettant de travailler avec des p-uplets d'entiers ou des suites finies.

Définition 1.4.1. On définit $\alpha : \mathbb{N}^2 \to \mathbb{N}$ par $\alpha(n,p) = \frac{1}{2}(n+p+1)(n+p)+p$.

α est récursive primitive, et on voit qu'elle est bijective (faire un dessin). De plus $\alpha(n,p) \geqslant \max(n,p)$. En particulier on peut retrouver (n,p) à partir de $\alpha(n,p)$ en posant

$$\beta^1(x) = \mu z \leqslant x : \exists t \leqslant x, \alpha(z,t) = x \text{ et } \beta^2(x,y) = \mu z \leqslant x : \exists t \leqslant x, \alpha(t,z) = x$$

On a bien $(\beta^1(\alpha(n,p)), \beta^2(\alpha(n,p)))$, autrement dit (β^1, β^2) est la bijection réciproque de α.

Définition 1.4.2. On définit par récurrence sur $p \in \mathbb{N}^*$ des bijections $\alpha_p : \mathbb{N}^p \to \mathbb{N}$ par $\alpha_1(x) = x$ puis $\alpha_{p+1}(x_1, ..., x_{p+1}) = \alpha_p(x_1, ..., x_{p-1}, \alpha(x_p, x_{p+1})))$.

On définit également $\beta_1^1(n) = n$ puis par récurrence sur p, pour $1 \leqslant k \leqslant p$ des fonctions β_p^k : $\beta_{p+1}^1(x) = \beta_p^1(x)$,..., $\beta_{p+1}^{p-1}(x) = \beta_p^{p-1}(x)$ puis $\beta_{p+1}^p = \beta^1(\beta_p^p(x))$ et $\beta_{p+1}^{p+1}(x) = \beta^2(\beta_p^p(x))$.

On montre par récurrence que toutes ces fonctions sont récursives primitives, et par construction $(\beta_p^1, ..., \beta_p^p)$ est la bijection réciproque de α_p. On a $\alpha_2 = \alpha$ et $\beta^1 = \beta_2^1$, $\beta^2 = \beta_2^2$.

Exemple 1.4.3. La suite de Fibonacci est récursive primitive : considérons la fonction suivante définie par récurrence qui encode (u_n, u_{n+1}) ; on a $v(0) = \alpha_2(1,1)$ puis $v(n+1) = \alpha_2\left(\beta_2^2(v(n)), \beta_2^1(v(n)) + \beta_1^2(v(

On va maintenant encoder les suites finies d'entiers, c'est-à-dire $\bigcup_{p\in\mathbb{N}} \mathbb{N}^p$, où par convention $\mathbb{N}^0$ est le singleton suite vide.

Définition 1.4.4. Etant donnée une suite finie d'entiers $(x_1, ..., x_p)$, on note $\Omega(x_1, ..., x_p) = \pi(0)^{x_1+1} \cdots \pi(p)^{x_p+1}$. On pose également $\Omega(\emptyset) = 1$.

On note $\delta(i,x) = \mu z \leqslant x : x$ n'est pas divisible par $\pi(i)^{z+1}$.

Remarquons qu'à p fixé la fonction Ω est récursive primitive. De plus Ω est injective. Enfin $\delta(i,x)$ est l'exposant de $\pi(i)$ dans la décomposition de x en produit de nombres premiers et permet donc de reconstituer $(x_1, ..., x_p)$ à partir de $\Omega(x_1, ..., x_p)$

1.5 Fonction d'Ackerman

Dans cette section, nous définissons une variante de la fonction d'Ackermae. Bien que définie par induction, nous allons voir qu'elle n'est pas récursive primitive.

Définition 1.5.1. La fonction d'Ackerman est la fonction $\xi \in \mathcal{F}_2$ définie par : pour tous $x, y \in \mathbb{N}$
- $\xi(0,x) = 2^x$
- $\xi(y,0) = 1$
- $\xi(y+1, x+1) = \xi(y, \xi(y+1, x))$

Pour $n \in \mathbb{N}$, notons $\xi_n \in \mathcal{F}_1$ la fonction $\xi(n, \cdot)$. Alors on montre par récurrence que ξ_n est récursive primitive : en effet la fonction ξ_{n+1} est définie par récurrence par $\xi_{n+1}(0) = 2^n$ et $\xi_{n+1}(x+1) = \xi_n(\xi_{m+1}(x))$.

Nous commençons maintenant une série de lemmes qui nous mèneront au fait que la fonction d'Ackerman n'est pas récursive primitive.

Lemme 1.5.2. Pour tout n et tout x on a $\xi_n(x) > x$.

Démonstration. On le montre par récurrence sur n. Pour $n = 0$ on a bien $2^x > x$ pour tout $x \in \mathbb{N}$. Ensuite supposons que pour tout $x \in \mathbb{N}$ on a $\xi_n(x) > x$ et montrons par récurrence sur x que pour tout $x \in \mathbb{N}$ on a $\xi_{n+1}(x) > x$. Pour $x = 0$ on a $\xi_{n+1}(0) = 1 > 0$, maintenant supposons $\xi_{n+1}(x) > x$, alors $\xi_{n+1}(x+1) = \xi_n(\xi_{n+1}(x))$. Par hypothèse de récurrence sur n on a alors $\xi_{n+1}(x+1) > \xi_{n+1}(x) > x$ et puisque l'on a affaire à des entiers on a bien $\xi_{n+1}(x+1) > x+1$, ce qui conclut la récurrence sur x puis sur n. □

Lemme 1.5.3. Pour tout $n \in \mathbb{N}$ la fonction ξ_n est strictement croissante.

Démonstration. C'est clairement vrai pour $n = 0$, et pour $n \geqslant 1$ on écrit $\xi_n(x+1) = \xi_{n-1}(\xi_n(x)) > \xi_n(x)$ d'après le lemme précédent, d'où la stricte croissance voulue. □

Lemme 1.5.4. Pour tous $n, x \in \mathbb{N}$ on a $\xi_{n+1}(x) \geqslant \xi_n(x)$.

Démonstration. Pour $x = 0$ c'est vrai car $1 \geqslant 1$. Ensuite, pour $x \geqslant 1$: écrivons $\xi_{n+1}(x) = \xi_n(\xi_{n+1}(x-1))$, alors comme d'après le lemme 1.5.2 $\xi_{n+1}(x-1)) \geqslant x$ et d'après le lemme 1.5.3 ξ_n est croissante, on conclut $\xi_{n+1}(x) \geqslant \xi_n(x)$. □

On va maintenant montrer que la fonction d'Ackerman croit plus vite que toute fonction récursive. Pour ce faire, il faut d'abord préciser ce qu'on entend par croire plus vite.

Définition 1.5.5. On dit qu'une fonction $f \in \mathcal{F}_1$ **domine** $g \in \mathcal{F}_p$ s'il existe $M \in \mathbb{N}$ tel que pour tous $(x_1, ..., x_p)$ sauf un nombre fini d'entre eux, on a

$$g(x_1, ..., x_p) \leqslant f(\max(x_1, ..., x_p, M)).$$

Notons que si la fonction f est strictement croissante, f domine g ssi on a $g(x_1, ..., x_p) \leqslant f(\max(x_1, ..., x_p))$ pour tous les $(x_1, ..., x_p)$ sauf pour un nombre fini d'entre eux. De plus, toujours si f est croissante, on a

$$f(\max(x_1, ..., x_p, M)) = \max(f(x_1), ..., f(x_p), f(M)).$$

On définit par récurrence sur k la fonction ξ_n^k par $\xi_n^0(x) = x$ puis $\xi_n k + 1(x) = \xi_n(\xi_n^k(x))$ (autrement dit ξ_n^k itère k fois la fonction ξ_n). Comme $\xi_n(x) > x$ pour tout x, on a $\xi_n^k < \xi_n^{k+1}$. De plus les fonctions ξ_n^k sont strictement croissantes par le lemme 1.5.3

Soit alors

$$\mathcal{C}_n := \{f \in \mathcal{F} : \exists k, \xi_n^k \text{ domine } f\}.$$

Comme $\xi_{n+1} \geqslant \xi_n$ et les ξ_n sont croissantes, on a $\xi_n^k \leqslant \xi_{n+1}^k$ et donc $\mathcal{C}_n \subseteq \mathcal{C}_{n+1}$.

Lemme 1.5.6. $\mathcal{C}_n$ est clos par composition.

Démonstration. Soient $f_1, ..., f_q \in \mathcal{F}_p \cap \mathcal{C}_n$, soit $g \in \mathcal{F}_q \cap \mathcal{C}_n$. Soit M et k suffisamment grand tel que pour tous $(x_1, ..., x_p)$ et tout $i \in \{1, ..., q\}$ on ait $f_i(x_1, ..., x_p) \leqslant \xi_n^k(\max(M, x_1, ..., x_p))$ et tous $(y_1, ..., y_q)$ on ait $g(y_1, ..., y_q) \leqslant \xi_n^k(\max(M, x_1, ..., x_p))$. Comme ξ_n^k est croissante on a

$$g(f_1(x_1, ..., x_p), ..., f_q(x_1, ..., x_p)) \leqslant \xi_n^k(\max(\xi_n^k(\max(M, x_1, ..., x_p)), M)).$$

Toujours par croissance, $\xi_n^k(\max(M, x_1, ..., x_p)) = \max(\xi_n^k(x_1), ..., \xi_n^k(x_p), \xi_n^k(M))$ et donc

$$g(f_1(x_1, ..., x_p), ..., f_q(x_1, ..., x_p)) \leqslant \max(\xi_n^k\xi_n^k(x_1), ..., \xi_n^k\xi_n^k(x_p)x_p, \xi_n^k\xi_n^k(M), \xi_n^k(M)).$$

Ainsi $g(f_1(x_1, ..., x_p), ..., f_q(x_1, ..., x_p)) \leqslant \xi_n^{2k}(\max(x_1, ..., x_p), M)$ donc $g \in \mathcal{C}_n$. □

La définition par récurrence va nous faire passer de $\mathcal{C}_n$ à $\mathcal{C}_{n+1}$; pour cela il nous faut borner ξ_n^k en fonction de ξ_{n+1}.

Lemme 1.5.7. On a pour tous x et k

$$\xi_n^k(x) \leqslant \xi_{n+1}(x+k).$$

Démonstration. Par récurrence sur k. Pour $k=0$ il n'y a rien à faire ; ensuite $\xi_{n+1}(x+k+1) = \xi_n(\xi_{n+1}(x+k)) \geqslant \xi_n\xi_n^k(x) = \xi_n^{k+1}(x)$. □

Lemme 1.5.8. Si $g, h \in \mathcal{C}_n$ et f est définie par récurrence à partir de g et h alors $f \in \mathcal{C}_{n+1}$.

Démonstration. Soient M, k assez grand pour que $g(x_1, ..., x_p) \leqslant \xi_n^k(\max(x_1, ..., x_k, M))$ et $h(x_1, ..., x_p, y, z) \leqslant \xi_n^k(\max(x_1, ..., x_k, M))$. On montre par récurrence que que

$$f(x_1, ..., x_k, y) \leqslant \xi_n^{k+ky}(\max(x_1, ..., x_k, M))$$

C'est vrai pour $y=0$, et l'hérédité se prouve comme pour la composition : on écrit

$$\begin{aligned} f(x_1, ..., x_p, y+1) &= h(x_1, ..., x_p, y, f(x_1, ..., x_p, y)) \\ &\leqslant \max(\xi_n^k(x_1), ..., \xi_n^k(x_p), \xi_n^k(y), \xi_n^k(\xi_n^{k+ky}(\max(x_1, ..., x_k, M))), M) \\ &\leqslant \xi_n^{k+(k+1)y}(\max(x_1, ..., x_p, M)) \end{aligned}$$

Ceci prouve la récurrence, maintenant on $\xi_n^{k+(k+1)y}(x) \leqslant \xi_{n+1}(x+k+(k+1)y)$, or les fonction $(x_1, ..., x_p, y) \mapsto \max(x_1, ..., x_p) + k + (k+1)y$ est dans $\mathcal{C}_0$ donc par stabilité par composition de $\mathcal{C}_{n+1}$, on a $(x_1, ..., x_p, y) \mapsto \xi_{n+1}(\max(M, \max_i(x_i + k + (k+1)y)))$ qui est dans $\mathcal{C}_{n+1}$, donc f est bien dans $\mathcal{C}_{n+1}$. □

Soit alors $\mathcal{C} = \bigcup_n \mathcal{C}_n$. D'après les lemmes précédents $\mathcal{C}$ est stable par composition et récurrence, et contient les projections, les fonctions constantes et la fonction successeur. Ainsi l'ensemble des fonctions récursives primitives est inclus dans $\mathcal{C}$.

Théorème 1.5.9. *La fonction $g : x \mapsto \xi(x, 2x)$ n'est pas dans $\mathcal{C}$.*

Démonstration. Par l'absurde supposons qu'il existe $n \in \mathbb{N}$ et k tel que g soit dominée par ξ_n^k. Alors pour tout x sauf un nombre fini, on a

$$\xi_x(2x) \leqslant \xi_n^k(x) \leqslant \xi_{n+1}(x+k).$$

Prenons alors x strictement plus grand que k et n et suffisamment grand, alors $\xi_{n+1}(x+k) < \xi_{n+1}(2x) \leqslant \xi_x(2x)$, ce qui contredit l'inégalité précédente. □

2 Fonctions récursives

2.1 Fonctions partielles récursives

Dans cette section, on définit les fonctions récursives. Par rapport aux fonctions primitives récursives, on aura certaines fonctions qui seront seulement partiellement définies. On va en effet s'autoriser des schémas μ non bornés, et la fonction ne sera pas définie en x si on ne peux trouver de y tel que $(x, y) \in A$.

On définit donc $\mathcal{F}_p^*$ comme l'ensemble des fonctions $f : A \to \mathbb{N}$ où $A \subseteq \mathbb{N}^p$. On appelle A le **domaine** de f. On dit que f est **totale** si son domaine est $\mathbb{N}^p$. On note $\mathcal{F}^* = \bigcup_{p\in\mathbb{N}} \mathcal{F}_p^*$.

Définition 2.1.1. Étant donnée $f_1, ..., f_n \in \mathcal{F}_p^*$ et $g \in \mathcal{F}_n^*$, la fonction **composée** $g(f_1, ..., f_n)$ est l'élément de $\mathcal{F}_p^*$ défini par : pour tous $x_1, ..., x_p \in \mathbb{N}$,

$$g(f_1, ..., f_n)(x_1, ..., x_p) = g(f_1(x_1, ..., x_p), ..., f_n(x_1, ..., x_p)),$$

la fonction n'étant définie que là où l'expression ci-dessus est définie. Plus précisément, si A est le domaine de g le domaine de $g(f_1, ..., f_n)$ est $(f_1, ..., f_n)^{-1}(A)$.

Définition 2.1.2. Étant donnée une fonction $g \in \mathcal{F}_p^*$ et $h \in \mathcal{F}_{p+2}^*$, il existe une unique fonction $f \in \mathcal{F}_{p+1}^*$ telle que pour tous $x_1, ..., x_p, y \in \mathbb{N}$ on ait

$$f(x_1, ..., x_p, 0) = g(x_1, ..., x_p)$$
$$f(x_1, ..., x_p, y+1) = h(x_1, ..., x_p, y, f(x_1, ..., x_p, y)),$$

la fonction f n'étant pas définie dès lors que l'un des termes ci-dessus n'est pas défini. On appelle f la fonction **définie par récurrence** à partir de g et h.

Remarquons que par construction si $f(x_1, ..., x_p, y)$ n'est pas défini alors pour tous $z \geqslant y$ on a que $f(x_1, ..., x_p, z)$ n'est pas défini non plus.

Définition 2.1.3. Étant donné une fonction $f \in \mathcal{F}_{p+1}^*$, la fonction g définie par **schéma** μ sur f s'écrit

$$g(x_1, ..., x_p) = \mu y : f(x_1, ..., x_p, y) = 0$$

et est donnée par $g(x_1, ..., x_p) = y$ si pour tout $z \leqslant y$ $f(x_1, ..., x_p, z)$ est définie et non nul, tandis que $f(x_1, ..., x_p, y) = 0$, et sinon n'est pas définie. Si $A \subseteq \mathbb{N}^{p+1}$, on notera $\mu y : (x_1, ..., x_p) \in A = \mu y : 1 \dot{-} \chi_1(x_1, ..., x_p, y) = 0$

Nous pouvons désormais définir l'ensemble des fonctions récursives partielles comme étant le plus petit ensemble de fonctions partielles dans $\mathcal{F}^*$ qui

i) contient les fonctions constantes $\mathbb{N}^p \to \mathbb{N}$,
ii) contient les projections π_p^i,
iii) contient la fonction successeur S,
iv) est stable par composition,
v) est stable par définition par récurrence,
vi) est stable par schéma μ.

Par définition les fonctions récursives primitives sont bien récursives (totales). La stabilité par permutation de variables demeure. La stabilté par définition par cas sera vue plus tard. On verra à la fin de ce chapitre que la fonction d'Ackermann est récursive elle aussi.

Un ensemble $A \subseteq \mathbb{N}^p$ est dit **récursif** si sa fonction caractéristique est récursive (en particulier cette fonction est totale !). Un ensemble est dit **récursivement énumérable** si c'est le domaine d'une fonction récursive.

Proposition 2.1.4. Tout ensemble récursif est récursivement énumérable.

Démonstration. Soit $A \subseteq \mathbb{N}^p$ récursif. Alors $\chi_{A \times \mathbb{N}}$ est récursive (c'est la fonction $(x_1, ..., x_{p+1}) \mapsto \chi_A(x_1, ..., x_p)$). On définit alors f par schéma μ total sur $A \times \mathbb{N}$, c'est-à-dire

$$f(x_1, ..., x_p) = \mu y : (x_1, ...x_p, y) \in A \times \mathbb{N}$$

Il est alors clair que le domaine de f est A. □

On verra plus tard par un argument diagonal qu'il existe des ensembles récursivement énumérables non récursifs. Ceci nécessite de pouvoir énumérer les fonctions récursives, et on va faire ça en utilisant les machines de Turing.

2.2 Machines de Turing

Une machine de Turing consiste en un nombre fini n de bandes infinies parallèles constituées de cases numérotées par les entiers naturels. Chaque case contient un symbole qui appartient à $\Sigma := \{0, 1, \#\}$ où $\#$ est le symbole de début de bande, uniquement présent sur la première case. On a également une tête de lecture qui permet de lire et écrire sur les cases de chacune des bandes situées à sa position.

Une machin de Turing possède également un nombre fini d'états dont on notera Q l'ensemble. Il y a deux états particuliers : l'état initial q_i et l'état final q_f.

Enfin, la table de transition est une application $M : S^n \times Q \to S^n \times Q \times \{-1, +1, 0\}$.

La machine fonctionne de la manière suivante, étant donné un contenu arbitraire des bandes :

— Au temps $t = 0$ elle est dans l'état q_i, la tête de lecture est au dessus des cases 1

— Au temps t, étant dans un état $q \neq q_f$ et sa tête de lecture lisant $(s_1, ..., s_n)$, on considère $M(s_1, ..., s_n, q) = (s'_1, ..., s'_n, q', f)$. La tête de lecture remplace alors $(s_1, ..., s_n)$ par $(s'_1, ..., s'_n)$, la tête de lecture est déplacée dans la direction indiquée par f et enfin la machine se met dans l'état q' au temps $t + 1$

— Au temps t, si $q = q_f$, la machine s'arrête de fonctionner et t est alors le temps de calcul e la machine étant donné le contenu initial des bandes.

La machine commence dans l'état q_i et s'arrête de fonctionner si elle arrive dans l'état q_f. Elle peut très bien ne jamais s'arrêter de fonctionner. Une restriction s'impose : si la machine lit $(d, ..., d)$ elle ne peut le modifier et la tête de lecture ne peut aller à gauche.

On dit qu'une bande représente l'entier n si elle est commence par $\#$, puis est suivie du symbole 1 n fois, puis ne contient plus que des 0.

Définition 2.2.1. On dit qu'une machine de Turing à $n \geqslant p + 1$ bandes **calcule** une fonction $f \in \mathcal{F}_p^*$ si pour tous $(x_1, ..., x_p)$, quand on lance la machine avec les p premières bandes représentant $x_1, ..., x_p$ et les autres représentant 0,

— Si $f(x_1, ..., x_p)$ n'était pas définie, alors la machine ne s'arrête jamais

— Si $f(x_1, ..., x_p)$ est définie, la machine s'arrête et se retrouve avec les p premières bandes qui représentent $x_1, ..., x_p$, la $p + 1$-ème bande qui représente $f(x_1, ..., x_p)$ et les autres bandes qui représentent 0.

On appelle **T-calculable** une fonction calculable par une machine de Turing.

Exemple 2.2.2. La fonction successeur est T-calculable.

2.2.1 Les fonctions récursives sont T-calculables

Lemme 2.2.3. Les fonctions constantes sont T-calculables.

Démonstration. Soit $k \in \mathbb{N}$, montrons que la fonction constante égale à k de $\mathbb{N}^p$ dans $\mathbb{N}$ est T-calculable. On va se donner $k + 2$ états $q_0, ..., q_{k+1}$, q_0 étant l'état initial et q_k l'état final. On pose alors $M(s_1, ..., s_{p+1}, q_0) = (s_1, ..., s_{p+1}, q_1, +1)$ puis pour $i = 1, ..., k$ $M(s_1, ..., s_{p+1}, q_i) = (s_1, ..., s_p, 1, +1)$ et enfin $M(s_1, ..., s_{p+1}, q_{k+1}) = (s_1, ..., s_{p+1}, q_{k+1}, 0)$. □

Lemme 2.2.4. La fonction successeur est T-calculable.

Démonstration. La machine a deux rubans. On a deux états (q_0, q_1) où q_0 est l'état initial, q_1 est l'état final. On a $M(d, d, q_0) = (d, d, q_0, +1)$, puis $M(|, b, q_0) = (|, |, q_0, +1)$ puis $M(b, b, q_0) = (b, |, q_1)$. □

Lemme 2.2.5. Les fonctions projection sont T-calculables.

Démonstration. Soit $p \in \mathbb{N}$ et $k \in \{1, ..., p\}$ La machine a deux rubans. On a deux états (q_0, q_1) où q_0 est l'état initial, q_1 est l'état final. On a $M(d, d, q_0) = (d, d, q_0, \rightarrow)$, puis $M(s_1, ..., s_{k-1}, |, s_{k+1}, ..., s_p, t, q_0) = (s_1, ..., s_{k-1}, |, s_{k+1}, ..., s_p, |, q_0, \rightarrow)$ puis $M(s_1, ..., s_{k-1}, b, s_{k+1}, ..., s_p, t, q_0) = (s_1, ..., s_{k-1}, b, s_{k+1}, ..., s_p, b, q_1)$. □

Lemme 2.2.6. Soient $f_1, ..., f_n \in \mathcal{F}_p^*$ T-calculables, et soit $g \in \mathcal{F}_n^*$ également T-calculable. Alors $g(f_1, ..., f_n)$ est T-calculable.

Démonstration. Soient p_1,...,p_n les nombres de bandes de machines de Turing $M_1, ..., M_n$ calculant f_1,..., f_n respectivement, et p_{n+1} le nombre de bandes de la machine de Turing M_{n+1} calculant g. On ne va pas décrire précisément les tables de transitions et se contenter de décrire le comportement d'une machine de Turing calculant $g(f_1, ..., f_n)$. Elle a $p_1 + ... + p_n + p_{n+1}$ bandes que l'on numérote (i, j) avec $i \in \{1, ..., n+1\}$ et $j \in \{1, ..., p_i\}$. Pour chaque $i \in \{1, ..., n+1\}$, notre machine contient une copie de M_i qui travaille sur les bandes $(1, j)$ avec $j \in \{1, ..., p_i\}$. Au début, la machine copie la bande $(1, 1)$ sur toutes les bandes $(i, 1)$ pour $i = 2, ..., n$ (avec la même méthode que pour la projection).

Ensuite, les machines M_1,...,M_n effectuent leurs calculs une par une à la suite (en particulier à la fin de chaque calcul on ramène la tête de lecture en position 1). Puis on copie le contenu de (i, p_i) sur $(n + 1, i)$ tout en l'effaçant pour chaque $i \in \{1, ..., n\}$. Enfin on fait travailler M_{n+1}. On obtient alors bien le résultat voulu sur la bande $(n + 1, p_{n+1})$. □

Lemme 2.2.7. Soit $f \in \mathcal{F}_{p+1}^*$ définie par récurrence à partir de $g \in \mathcal{F}_p^*$ et $h \in \mathcal{F}_{p+2}^*$ toutes deux T-calculables. Alors f est T-calculable.

Démonstration. Supposons que la machine M_g qui calcule g a p_1 bandes et M_h qui calcule h en a p_2. La machine calculant f a $p_1 + p_2$ bandes numérotées $(1, k)$ pour $k \in \{1, ..., p_1\}$ puis $(2, k)$ pour $k \in \{1, ..., p_2\}$ et enfin $(3, 1)$. La bande contenant le résultat final sera $(2, p_2)$. Au début M_g effectue son calcul, puis le résultat est recopié sur $(2, p + 2)$. On recopie également $(1, i)$ sur $(2, i)$ pour $i = 1, ..., p$.

Ensuite on arrive dans un état q, on recopie le contenu de $(1, p)$ sur $(3, 1)$. On teste si $(2, p + 1)$ est égal à $(3, 1)$, si c'est le cas on s'arrête. Sinon on commence par faire calculer M_h, on augmente $(2, p + 1)$ de 1 et on retourne à l'état q. □

Lemme 2.2.8. Soit $f \in \mathcal{F}_p^*$ définie par schéma μ à partir de g. Si g est T-calculable alors f aussi.

Démonstration. Soit m le nombre de bandes de M_g calculant g, notre nouvelle machine aura m bandes numérotées de 0 à $m - 1$. Au début la bande p contient 0 et représente y. Notre machine fonctionne ainsi : étant en q_i elle calcule $g(x_1, ..., x_p, y)$ qui apparait sur la bande $p + 1$. Elle teste ensuite le premier bit de la bande $p+1$, s'il est nul elle s'arrête, sinon elle incrémente y et retourne en q_i. □

Remarque. Dans les constructions précécentes, il faut bien voir que les machines de Turing construites donnent les mêmes fonctions partielles, en particulier elles ne s'arrêtent pas là où la fonction n'est pas définie.

2.2.2 Les fonctions T-calculables sont récursives

Pour montrer que les fonctions T-calculables sont récursives, on va devoir montrer qu'à chaque étape de calcule, le contenu des bandes dépend de manière récursive primitive de l'état initial. Pour rendre cette assertion précise, on doit d'abord coder l'état d'une machine de Turing. Commençons par le codage des bande : on code 0 par 0, 1 par 1 et # par 2 et le contenu de n bandes $(B_0, ..., B_{n-1}) = ((b_i^j)_{i\in\mathbb{N}})_{j=0}^{n-1}$ est alors codé par l'entier

$\Gamma(B_0, ..., B_{n-1}) = \sum_{j=0}^{n} \sum_{i=0}^{+\infty} 3^{ni+j} b_i^j$. On retrouve le i-ième élément du code x de la j-ème bande bande par la fonction $\gamma(x, i, j) = r(q(n, 3^{ni+j}), 3)$.

Etant donnée une machine du Turing à n bande $B_0, ..., B_{n-1}$ et r états $0, 1, ..., r-1$ (où l'état 0 est initial, l'état 1 est final) et où la tête est en position k et son état est q, sa configuration sera l'entier $\alpha_3(k, e, \Gamma(B_0, ..., B_{n-1}))$.

Lemme 2.2.9. Étant donnée une machine de Turing à n bandes de table de transition M, la fonction qui à $(x_1, ..., x_p, t)$ associe la configuration de la machine au temps t avec pour entrées $(x_1, ..., x_p)$ est récursive primitive.

Démonstration. Il s'agit d'une fonction définie par récurrence, notons la f. Tout d'abord, remarquons que la fonction c_p^n qui à $(x_1, ..., x_p)$ associe le code de n bandes codant la suite $(x_1, ..., x_p, 0, ..., 0)$ est récursive primitive, puisque

$$c_p^n(x_1, ..., x_p) = 2\sum_{j=0}^{n-1} 3^j + \sum_{j=0}^{p-1} \sum_{i=1}^{x_{j+1}} 3^{j+ni}$$

Alors $f(x_1, ..., x_p, 0) = \alpha_3(1, 0, c_p^n(x_1, ..., x_p))$. Puis on va voir que $f(x_1, ..., x_p, t+1)$ est définie par récurrence par cas, les cas étant donnés par la fonction de transition M de notre machine de Turing. Les cas vont porter sur ce que la tête observe et sur l'état actuel de la machine. Posons $y = f(x_1, ..., x_p, t)$ puis $z = \beta_3^3(y)$ on regarde donc

$$M(\gamma(z, \beta_3^1(y), 0), ..., \gamma(z, \beta_3^1(y), q-1), \beta_3^2(y)) = (s_1, ..., s_q, e, f).$$

On pose alors

$$f(x_1, ..., x_p, t+1) = \alpha_3(\beta_3^1(y) + f, e, z + \sum_{j=0}^{n-1} 3^{j+q\beta_3^1(y)} s_j \dot{-} \sum_{j=0}^{n-1} 3^{j+n\beta_3^1(y)} \gamma(z, \beta_2^1(y), j).).$$

Donc f est bien récursive primitive. □

Théorème 2.2.10. *Toute fonction T-calculable est récursive.*

Démonstration. Soit une fonction $g \in \mathcal{F}_p^*$ calculable par une machine de Turing M à q bandes, soit f du lemme précédent qui est récursive primitive. Remarquons qu'il faut au moins t étapes pour que la bande de sortie de M contienne l'entier t, en particulier la fonction $h(x_1, ..., x_p, t)$ qui retourne l'entier codé par la bande de sortie après t étapes est récursive primitive. En effet

$$h(x_1, ..., x_p, t) = [\mu y \leqslant t : r(q(\beta_3^3(f(x_1, ..., x_p, t)), 3^{y\beta_3^1(f(x_1,...,x_p,t))+p}), 3)) = 0] \dot{-} 1.$$

Soit alors

$$T(x_1, ..., x_p) = \mu t : \beta_3^2 f(x_1, ..., x_p, t) = 1$$

T est récursive, et alors $g(x_1, ..., x_p) = h(x_1, ..., x_p, T(x_1, ..., x_p))$. □

2.3 Conséquences

Corollaire 2.3.1. Toute fonction récursive dont le temps de calcul est bornée par une fonction récursive primitive est récursive primitive.

Démonstration. En effet, dans la preuve ci-dessus, la fonction T est alors définie par un schéma μ-borné donc récursive primitive, donc g est récursive primitive. □

La réciproque est vraie, comme on le voit par induction sur les fonctions récursives primitives en reprenant les preuves que les fonctions récursives sont T-calculables.

Corollaire 2.3.2. La classe des fonctions récursives totales est la plus petite classe de fonctions contenant les fonctions récursives primitives close par composition et schéma μ total e (c'est-à-dire qu'on ne s'autorise à faire des schémas μ que lorsque la fonction résultante est totale).

Démonstration. En effet dans la preuve du théorème la fonction T est définie par un schéma μ-total sur une fonction récursive primitive, et g est ensuite obtenue en composant T avec des fonctions récursives primitives. □

On va maintenant passer en revue des conséquences importantes sur les ensembles récursivement énumérables.

Corollaire 2.3.3. Les assertions suivantes sont vraies :

(1) Tout ensemble récursivement énumérable est la projection d'un ensemble récursif primitif.

(2) Toute projection d'un ensemble récursif est récursivement énumérable.

(3) Toute projection d'un ensemble récursivement énumérable est récursivement énumérable.

Démonstration. (1) Soit $A = \mathrm{dom} g$ où $g \in \mathcal{F}_p^*$ est récursive. On reprend les notations de la preuve du théorème. Soit $B = \{(x_1, ..., x_p, t) : \beta_3^2 f(x_1, ..., x_p, t) = 1\}$. Alors B est récursif primitif, et A est la projection sur $\mathbb{N}^p$ de B.

(2) Si $A \subseteq \mathbb{N}^{p+q}$ est récursif, la projection de A sur les p premières coordonnées est le domaine de la fonction récursive $(x_1, ..., x_p) \mapsto \mu y(x_1, ..., x_p, \beta_q^1(y), ..., \beta_q^q) \in A$.

(3) Comme la projection d'une projection est une projection (3) découle de (2) et (1) immédiatement. □

La proposition suivante justifie l'appellation *récursivement énumérable*.

Proposition 2.3.4. Soit A un sous-ensemble de $\mathbb{N}$, alors sont équivalentes

(1) A est récursivement énumérable

(2) A est l'image d'une fonction récursive

(3) A est l'image d'une fonction récursive primitive

Démonstration. (3)⇒(2) est claire, (2)⇒(1) découle de (2) de la proposition précédente car le graphe d'une fonction récursive est récursif et l'image est la projection sur la seconde coordonnée du graphe. Enfin montrons (1)⇒(3). Soit A est récursivement énumérable, soit g récursive telle que $A = \mathrm{dom} g$, alors on fixe une machine de Turing calculant g. On sait d'après la preuve du théorème que la fonction $h(x, t)$ qui retourne l'entier codé par la bande de sortie après t étapes est récursive primitive, et que l'ensemble B des (x, t) tels que la machine soit à l'état final au temps t avec pour entrée x est récursif primitif. Soit $x_0 \in A$. Notre fonction récursive f d'image A est donnée par $f(x) = \begin{cases} h(\beta_2^1(x), \beta_2^2(x)) & \text{si } (\beta_2^1(x), \beta_2^2(x)) \in B \\ n & \text{sinon.} \end{cases}$. □

Proposition 2.3.5. Une fonction partielle est récursive ssi son graphe est récursivement énumérable.

Démonstration. L'ensemble des couples (x, y) tels que $x = y$ est récursivement énumérable, c'est le domaine d'une fonction h. Si f est récursive, son graphe est le domaine de la composée $h(f(x_1, ..., x_p), y)$ qui est récursive, donc le graphe est récursivement énumérable.

Réciproquement si le graphe G de f est récursivement énumérable, alors $\alpha_2(G)$ est également récursivement énumérable, donc par la proposition précédente c'est l'image de $h : \mathbb{N} \to \mathbb{N}$ récursive totale. On pose alors $g(n) = \mu m : \beta_2^1 h(m) = n$ qui est récursive, puis on a $f(n) = \beta_2^2 h(g(n))$. □

2.4 Machine de Turing universelle

On va maintenant construire une fonction récursive qui calcule toutes les fonctions récursives en encodant toutes les machines de Turing possibles. Jusqu'ici on a déjà codé les configurations, reste donc à coder les tables de transition et le nombre de bandes.

Pour chaque $(s_1, ..., s_n) \in \{0, 1, 2\}^n$, on note $r(s_1, ..., s_n) = \sum_{i=0}^{n-1} s_{i+1} 3^i$ puis pour un $q \in \mathbb{N}$

$$r_1(s_1, ..., s_n, q) = \alpha_2(r(s_1, ..., s_n), q)$$

Pour $M(s_1, ..., s_n) = (s'_1, ..., s'_n, q', \epsilon)$ on note $r_2(M(s_1, ..., s_n, q)) = \alpha_3(r(s'_1, ..., s'_n), q', \epsilon+1)$. Le code de la table de transition est alors

$$C(M) = \prod_{(s_1,...,s_n)\in\{0,1,2\}^n, e\in\{0,1,...,q-1\}} \pi(r_1(s_1, ..., s_n, e))^{r_2(M(s_1,...,s_n,e))}.$$

Le code de la machine de Turing à n bandes de table de transition M avec q états numérotés de 0 à $q - 1$ est alors $\alpha_3(n, q, C(M))$.

On peut montrer que l'ensemble des codes de machines de Turing est récursif primitif, il suffit de reprendre les conditions qu'on a énoncées. De plus, si on note I_p l'ensemble des codes de machines de Turing ayant au moins $p + 1$ bandes, alors I_p est récursif primitif.

Théorème 2.4.1. *La fonction* Sit *qui à* $(i, t, x_1, ..., x_p)$ *associe* 0 *si* $i \notin I_p$ *et sinon la configuration de la machine de Turing d'indice* i *au temps* t *avec comme entrées* $(x_1, ..., x_p)$ *est récursive primitive.*

Démonstration. On commence par définir Sit par cas : si $i \notin I_p$ on pose $\text{Sit}(i, t, x_1, ..., x_p) = 0$.

Pour le cas intéressant où $i \in I_p$, il s'agit essentiellement de reprendre la preuve du lemme 2.2.9 sauf que cette fois-ci on a en plus comme paramètre la machine de Turing via son code i. Le nombre de bandes est $\beta_3^1(i)$. Le code des $\beta_3^1(i)$ bandes codant $(x_1, ..., x_p, 0, ..., 0)$ est récursif primitif car donné par la formule

$$c(i, x_1, ..., x_p) = 2 \sum_{j=0}^{\beta_3^1(i)-1} 3^j + \sum_{j=0}^{p-1} \sum_{k=1}^{x_{j+1}} 3^{j+\beta_3^1(i)k}.$$

Alors $\text{Sit}(i, x_1, ..., x_p, 0) = \alpha_3(1, 0, c(i, x_1, ..., x_p))$. Calculons $\text{Sit}(i, x_1, ..., x_p, t + 1)$ en fonction de $s := \text{Sit}(i, x_1, ..., x_p, t)$ La position de la tête de lecture est $k := \beta_3^1(s)$. Le code de ce qu'elle lit est :

$$C := r(q(\beta_3^3(s), 3^{k\beta_3^1(i)}), 3^{\beta_3^1(i)})$$

La table de transition nous renvoie alors l'entier

$$m := \delta(C, \beta_3^3(i))$$

Le nouveau contenu des cases numéros k est alors $\beta_3^1(m)$, le nouvel état est $\beta_3^2(m)$ et la nouvelle position de la tête de lecture est $k + \beta_3^3(m)$. Ainsi

$$\mathrm{Sit}(i, x_1, ..., x_p, t+1) = \alpha_3(k + \beta_3^3(m), \beta_3^2(m), \beta_3^3(s) + 3^{k\beta_3^1(i)}\beta_3^1(m) - 3^{k\beta_3^1(i)}C).$$

Ceci prouve que la fonction est bien récursive primitive comme annoncé car définie par cas puis récurrence à partir de composition de fonctions récursives primitives. □

Définition 2.4.2. On note B^p l'ensembles des uplets $(i, x_1, ..., x_p, t)$ tels que $\beta_3^2(\mathrm{Sit}(i, x_1, ..., x_p, t) = 1$, c'est-à-dire que la machine de Turing de code i a fini son calcul sur $(x_1, ..., x_p)$ au temps t.

On note C^p l'ensemble des uplets $(i, x_1, ..., x_p, t, y)$ tels que $(i, x_1, ..., x_p, t) \in B^p$ et l'entier y est codé sur la bande de sortie.

D'après le théorème précédent B^p est récursif primitif. On pose alors $T(i, x_1, ..., x_p) = \mu t : (i, x_1, ..., x_p, t) \in B^p$. Alors T est récursive.

Soit h la fonction qui à $(i, x_1, ..., x_p, t)$ associe l'entier codé sur la bande de sortie au temps t (et n'est pas définie si $i \notin I_p$). Par la même preuve que pour le lemme 2.2.10 on voit que h est récursive primitive. Ainsi C^p est récursif primitif.

Définition 2.4.3. On définit φ^p par $\phi^p(i, x_1, ..., x_p) = h(i, x_1, ..., x_p, T(i, x_1, ..., x_p)$ qui calcule donc le résultat obtenu par la machine de Turing i avec pour entrées $(x_1, ..., x_p)$, et qui n'est pas définie si un tel calcul ne termine pas où si $i \notin I_p$.

D'après la discussion précédente, on a :

Théorème 2.4.4. *Pour tout $p \geqslant 1$ la fonction φ^p est récursive. Pour toute fonction récursive partielle $f \in \mathcal{F}_p^*$ il existe $i \in I_p$ tel que $f = \varphi_i^p : (x_1, ..., x_p) \mapsto \varphi^p(i, x_1, ..., x_p)$. On dit que i est un **indice** de f.*

2.5 Conséquences

On peut maintenant utiliser notre travail pour montrer que les fonctions récursives sont stables par définition par cas. On se convaincra que la situation est plus compliquée que dans le cas des fonctions récursives primitives car la formule que l'on a donné (preuve de la proposition 1.2.5) donne une fonction dont l'ensemble de définition pourrait être trop petit.

Proposition 2.5.1. Soient $f, g \in \mathcal{F}_p^*$ récursives et soit $A \subseteq \mathbb{N}^p$ récursif. Alors la fonction

$$h : (x_1, ..., x_p) \mapsto \begin{cases} f(x_1, ..., x_p) & \text{si } (x_1, ..., x_p) \in A \\ g(x_1, ..., x_p) & \text{sinon.} \end{cases}$$

est récursive

Démonstration. Soit i indice de f, j indice de g, k indice de χ_A. On considère l'ensemble C^p des $(i, x_1, ..., x_p, t, y)$ tels que $(i, x_1, ..., x_p, t) \in B^p$ et $h(i, x_1, ..., x_p, t) = y$ (la machine de Turing d'indice i a fini son calcul et retourne y). Alors C^p est récursif primitif.

On considère alors l'ensemble C récursif primitif des $(x_1, ..., x_p, t, y)$ tels que $(i, x_1, ..., x_p, t, y) \in C^p$ et $(k, x_1, ..., x_p, 1) \in C^p$ ou $(j, x_1, ..., x_p, t, y) \in C^p$ et $(k, x_1, ..., x_p, 0) \in C^p$.

Alors $h(x_1, ..., x_p) = \mu y : (x_1, ..., x_p, \mu t : (x_1, ..., x_p, t, y) \in C, y) \in C$. □

Théorème 2.5.2. *Un ensemble est récursif ssi il est récursivement énumérable et son complémentaire l'est également.*

Démonstration. Si A est récursif, sont complémentaire l'est aussi, en particulier A et son complémentaire sont récursivement énumérables.

Si $A \subseteq \mathbb{N}^p$ est récursivement énumérable de complémentaire B récursivement énumérable, soit i tel que $A = \text{dom}\varphi^p(i, \cdot)$ et j tel que $B = \text{dom}\varphi^p(j, \cdot)$. Alors l'ensemble D des $(x_1, ..., x_p, t)$ tels que $(i, x_1, ..., x_p, t) \in B^p$ ou $(j, x_1, ..., x_p, t) \in B^p$ est récursif primitif.

La fonction $T(x_1, ..., x_p) = \mu t : (x_1, ..., x_p, t) \in D$ est récursive. On pose alors

$$h(x_1, ..., x_p, t) = \begin{cases} 1 & \text{si } (i, x_1, ..., x_p, t) \in B^p \\ 0 & \text{sinon.} \end{cases}$$

qui est récursive primitive. La fonction $h(x_1, ..., x_p, T(x_1, ..., x_p))$ est alors comme voulue. □

Théorème 2.5.3. *L'ensemble* $\text{dom}\varphi^1(x, x)$ *est récursivement énumérable de complémentaire non récursivement énumérable. En particulier il existe un sous-ensemble de* $\mathbb{N}$ *récursivement énumérable non récursif.*

Démonstration. Soit $A = \text{dom}\varphi^1(x, x)$, alors A est récursivement énumérable par définition. Supposons que son complémentaire B l'est aussi. Mais alors il existe n tel que $B = \text{dom}\varphi^1(n, \cdot)$.

Alors $n \in A \Leftrightarrow \varphi^1(n, n) \Leftrightarrow (n, n) \in B$ ce qui est manifestement contradictoire. □

Définition 2.5.4. Une propriété de p-uplets d'entiers est dite **décidable** si l'ensemble correspondant est récursif. On peut utiliser ce qu'on vient de faire pour voir que le problème de l'arrêt n'est pas décidable : l'ensemble des $(x, y) \in \text{dom}\varphi^1$ n'est pas récursif car sinon l'ensemble ci-dessus le serait aussi.

On dit de même qu'une propriété est **semi-décidable** si l'ensemble correspondant est récursivement énumérable. Cela veut dire que l'on a un algorithme qui, si on n'est pas dans l'ensemble ne s'arrête jamais, et si on est dans l'ensemble termine.

2.6 Trois théorèmes fondamentaux

2.6.1 Théorème s-n-m

Ce théorème permet de calculer de manière effective les indices des fonctions obtenues à partir d'autres fonctions dont on connait les indices (par example si on a les indices de deux fonctions, on saura calculer un indice de leur somme de manière récursive primitive). Son énoncé permet en fait de passer de φ^p à φ^q pour $q < p$ en voyant certains arguments comme des paramètres.

Théorème 2.6.1. *Soit* $n, m \geqslant 1$*, il existe une fonction récursive primitive* $s_m^n : \mathbb{N}^{m+1} \to \mathbb{N}$ *telle que pour tout* $i \in I_{n+m}$*, on ait*

$$\varphi^n(s_m^n(i, x_1, ..., x_m), y_1, ..., y_n) = \varphi^{n+m}(i, x_1, ..., x_m, y_1, ..., y_n)$$

Démonstration. Si $i \notin I_{n+m}$ on pose $s_m^n(i, x_1, ..., x_m) = 0$ (qui n'est pas un indice de machine de Turing).

Sinon, on change la machine de Turing d'indice i en faisant en sorte d'écrire sur les m premières bandes de lectures les entiers $x_1, ..., x_m$, puis en permutant les numéros de bandes de sorte à ce que l'ancienne bande de sortie se retrouve en position n et que les bandes d'entrée correspondant aux

n derniers arguments se retrouvent en premier. Enfin, on efface toutes les bandes correspondant au arguments $x_1, ..., x_m$.

La machine obtenue calcule la fonction $(y_1, ..., y_n) \mapsto \varphi^{n+m}(i, x_1, ..., x_m, y_1, ..., y_n)$. On peut vérifier (mais c'est très fastidieux !) que l'indice de la machine obtenue dépend de manière récursive primitive que i, si on le note $s^n_m(i, x_1, ..., x_m)$ on a le résultat voulu. □

Exemple 2.6.2. Montrons qu'on peut trouver une fonction récursive primitive qui à (i, j) associe un indice de la machine de Turing calculant $x \mapsto \varphi^1(i, x) + \varphi^1(j, x)$. Considérons la fonction

$$(i, j, x) \mapsto \varphi^1(i, x) + \varphi^1(j, x)$$

Elle est récursive, soit donc k un indice de cette fonction. Alors pour tout x, i, j,

$$\varphi^1(i, x) + \varphi^1(j, x) = \varphi^3(k, i, j, x) = \varphi^1(s^1_2(k, i, j), x)$$

Ainsi une fonction qui marche est $(i, j) \mapsto s^1_2(k, i, j)$.

2.6.2 Théorème de Rice

On a déjà vu que $\mathrm{dom}\varphi^1(x, x)$ est un ensemble récursif non récursivement énumérable. On va le réduire récursivement (cf. TD2 pour la définition) à tout ensemble non trivial d'indices pour obtenir :

Théorème 2.6.3. *Soit $\mathcal{G} \subseteq \mathcal{F}^*_1$ un ensemble de fonctions partielles récursives non vide et distinct de l'ensemble des fonctions partielles récursives. Alors*

$$A_{\mathcal{G}} := \{i \in I_1 : \varphi^1_i \in \mathcal{G}\}$$

n'est pas récursif.

Démonstration. Quitte à remplacer $\mathcal{G}$ par son complémentaire, on peut supposer que la fonction vide est dans $\mathcal{G}$. Soit b l'indice d'une machine calculant une fonction qui n'est pas dans $\mathcal{G}$. Considérons la fonction

$$\psi : (x, y) \mapsto \varphi^1(b, y) + \varphi^1(x, x) \dot{-} \varphi^1(x, x)$$

Alors la fonction $\psi(x, \cdot)$ est égale à la fonction vide si $(x, x) \notin \mathrm{dom}\varphi^1(x, x)$ (et donc dans $\mathcal{G}$), et sinon elle est égale à $\varphi^1(b, y)$ (donc pas dans $\mathcal{G}$).

On va calculer un indice de cette fonction : soit k un indice de ψ, alors $\psi(x, y) = \varphi^2(k, x, y) = \varphi^1(s^1_1(k, x), y)$. La fonction qui à x associe $s^1_1(k, x)$ est récursive primitive, et par construction l'image réciproque de $A_{\mathcal{G}}$ est le complémentaire de $\mathrm{dom}\varphi^1(x, x)$. Comme ce dernier n'est pas récursif, $A_{\mathcal{G}}$ n'est pas récursif. □

Remarque. On en déduit une version plus satisfaisante du fait que le problème de l'arrêt soit non décidable : l'ensemble des indices de machines de Turing à 1 paramètre qui terminent pour l'entrée 0 est non récursif.

En faisant la même preuve et en se souvenant que le complémentaire de $\mathrm{dom}\varphi^1(x, x)$ n'est en fait pas récursivement énumérable, on obtient :

Proposition 2.6.4. Soit $\mathcal{G}$ un ensemble de fonctions récursives partielles à 1 variable contenant la fonction vide et distinct de l'ensemble de toutes les fonctions partielles récursives. Alors l'ensemble

$$A_{\mathcal{G}} := \{i \in I_1 : \varphi^1_i \in \mathcal{G}\}$$

n'est pas récursivement énumérable.

En particulier, l'ensemble des indices calculant la fonction vide n'est pas récursivement énumérable. On dit que le problème de savoir si une machine ne va pas s'arrêter pour chaque entrée n'est pas semi-décidable.

2.7 Théorème(s) de point fixe

Ces théorèmes, dus à Kleene, sont parfois appelés théorèmes de la récursion.

On peut montrer (en ajoutant des états inutiles à la machine de Turing) qu'il existe une fonction récursive primitive $\psi^p : \mathbb{N} \to \mathbb{N}$ telle que $\psi(i) > i$ et pour tout $i \in I^p$, $\varphi_i^p = \varphi_{\psi(i)}^p$. Réciproquement, le théorème de point fixe de Kleene affirme que l'on peut trouver un tel indice pour toute fonction ψ récursive totale.

Théorème 2.7.1. *Soit $\alpha : \mathbb{N} \to \mathbb{N}$ récursive totale. Alors il existe $i \in I_p$ tel que*

$$\varphi_i^p = \varphi_{\alpha(i)}^p$$

Démonstration. Considérons la fonction $(y, x_1, ..., x_p) \mapsto \varphi^p(\alpha(s_1^1(y, y)), x_1, ..., x_p)$. Elle a un indice $k \in I_{p+1}$, donc

$$\varphi^{p+1}(k, y, x_1, ..., x_p) = \varphi^p(\alpha(s_1^1(y, y)), x_1, ..., x_p) = \varphi^p(s_1^p(k, y), x_1, ..., x_p)$$

d'après le théorème s-n-m. En posant $i = s_1^p(k, k)$ on obtient le résultat voulu en regardant l'équation ci-dessus pour $y = k$. □

Exemple 2.7.2. On va montrer que la fonction d'Ackerman est récursive. Considérons l'application $\psi \mapsto \psi^*$ de $\mathcal{F}_2^*$ dans $\mathcal{F}_2^*$ donnée par

$$\psi^*(x, y) = \begin{cases} 2^y & \text{si } x = 0 \\ 1 & \text{si } y = 0 \\ \psi(x - 1, \psi(x, y - 1)) & \text{sinon.} \end{cases}$$

On montre par induction que la fonction d'Ackerman est le seul point fixe de $\psi \mapsto \psi^*$. Montrons qu'il y a une fonction récursive (primitive) α qui à un indice de ψ associe un indice de ψ^*. Considérons la fonction de 3 variables

$$\theta(i, x, y) = \begin{cases} 2^y & \text{si } x = 0 \\ 1 & \text{si } y = 0 \\ \varphi^2(i, x - 1, \varphi^2(i, x, y - 1)) & \text{sinon.} \end{cases}$$

Soit k un indice de cette fonction, alors si i est un indice de ψ on voit que $\psi^*(x, y) = \theta(i, x, y) = \varphi^3(k, i, x, y) = \varphi^2(s_2^2(k, i), x, y)$. On pose donc $\alpha(i) = s_2^2(k, i)$, alors le théorème de point fixe dans sa première version nous donne un indice i tel que $\varphi_{\alpha(i)}^2 = \varphi_i^2$, donc la fonction d'Ackerman est d'indice i, en particulier elle est récursive.

On donne maintenant une version où l'indice de α est un paramètre supplémentaire : on peut alors trouver i de manière récursive primitive en fonction de l'indice de α.

Théorème 2.7.3. *Il existe une fonction $h_p : \mathbb{N} \to \mathbb{N}$ récursive primitive telle que dès lors que φ_j^1 est totale, on a $h_p(j) \in I_p$ et*

$$\varphi_{h_p(j)}^p = \varphi_{\varphi_j^1(h_p(j))}^p.$$

Démonstration. On reprend la démonstration précédente avec paramètre : on considère un indice k de la fonction $(j, y, x_1, ..., x_p) \mapsto \varphi^p(\varphi^1_j(s^1_1(y, y)), x_1, ..., x_p)$. Alors

$$\varphi^{p+2}(k, j, y, x_1, ..., x_p) = \varphi^p(\varphi^1_j(s^1_1(y, y)), x_1, ..., x_p) = \varphi^{p+1}(s^1_1(k, j), y, x_1, ..., x_p) = \varphi^p(s^1_1(s^1_1(k, j), y), x_1, ..., x_p).$$

On pose $h_p(j) = s^1_1(k, j)$, alors on a le résultat voulu en posant $y = s^1_1(k, j)$. □

Terminons avec le cas où α prend plusieurs paramètres.

Théorème 2.7.4. *Soit $p \geqslant 1$ et $n \in \mathbb{N}$, soit $\alpha : \mathbb{N}^{n+1} \to \mathbb{N}$ récursive totale. Alors il existe $h : \mathbb{N}^n \to \mathbb{N}$ récursive primitive à valeur dans I_p telle que pour tous $(x_1, ..., x_n)$,*

$$\varphi^p_{\alpha(x_1,...,x_n,h(x_1,...,x_n))} = \varphi^p_{h(x_1,...,x_n)}$$

Références

[CL93] René Cori and Daniel Lascar. *Logique mathématique. Cours et exercices. II : Fonctions récursives, théorème de Gödel, théorie des ensembles, théorie des modèles. Préface de J.-L. Krivine.* Paris : Masson, 1993.