7.3 Fonctions récursives partielles et ensembles récursivement énumérables

La minimisation est étendue au cas où on n'a pas nécessairement $\forall \vec{n} \exists m.\xi(\vec{n}, m) = 0$. Dans ce cas, la fonction ϕ définie est une fonction partielle :

$$\phi(\vec{n}) = \min\{m \mid (\xi(\vec{n}, m) = 0) \wedge \forall k < m, \xi(\vec{n}, k) \neq \perp\}$$

La composisition de fonctions récursives totales est étendue aux fonctions partielles : la composée n'est définie que lorsque les fonctions composantes le sont. De même, la récursion primitive appliquée à des fonctions partielles : $\phi = \mathsf{Prim}(\xi, \psi)$ est définie en $(m, \vec{n})$ si $m = 0$ et $\xi(\vec{n}) \neq \perp$ ou bien $m > 0$, $\phi(m-1, \vec{n})$ est définie et ψ est définie en $(\phi(m-1, \vec{n}), m-1, \vec{n})$.

Definition 7.3.1 *L'ensemble des fonctions* récursives partielles *(aussi appelées* fonctions récursives partielles*) est le plus petit ensemble de fonctions sur les entiers contenant les fonctions initiales et clos par récursion primitive, composition et minimisation.*

Un ensemble est *récursif* si sa fonction caractéristique est une fonction récursive totale. Un ensemble S est *récursivement énumérable* si la fonction qui vaut 1 quand $n \in S$ et est indéfinie ou vaut 0 quand $n \notin S$ est récursive partielle.

7.3.1 Turing-calculabilité et fonctions récursives

A l'appui de la thèse de Church, les notions de calculabilité définies sur les entiers par les machines de Turing et les fonctions récursives partielles coïncident :

Théorème 7.3.1 *Soit $f : \mathbb{N}^k \to \mathbb{N}$ une fonction partielle. f est récursive partielle (resp. récursive) ssi il existe une machine de Turing M_f telle que M_f s'arrête exactement sur les données du domaine de définition de f et, lorsqu'elle s'arrête sur $n_1, \ldots, n_k$, le ruban contient $f(n_1, \ldots, n_k)$.*

Preuve:
On suppose que les entiers sont représentés (dans une base b) par des mots.

Tout d'abord, si f est récursive partielle (resp. récursive totale), f est calculable par une machine de Turing. On montre que l'ensemble des fonctions partielles calculables par une M.T., contient les fonctions initiales et est clos par récursion primitive, par composition et par minimisation.

— Soient $M_0, M_1, \ldots, M_k$ calculant respectivement $f_0, f_1, \ldots, f_k$. Soit la machine M à $k+1$ rubans qui duplique la donnée sur les rubans puis simule $M_1, \ldots, M_k$ sur les rubans $1, \ldots k$ puis, si tous les calculs s'arrêtent, simule M_0 sur les résultats obtenus. M s'arrête sur la donnée $\vec{x}$ ssi $M_1, \ldots, M_k$ s'arrêtent sur $\vec{x}$ et M_0 s'arrête sur la donnée $f_1(\vec{x}), \ldots f_k(\vec{x})$. M s'arrête donc ssi $f_0(f_1(\vec{x}), \ldots, f_k(\vec{x}))$ est défini et, dans ce cas, le résultat du calcul de M est $f_0(f_1(\vec{x}), \ldots, f_k(\vec{x}))$. On a donc la clôture par composition.

— Pour la récursion primitive : soit

$$f(\vec{x}, n) = \begin{cases} g(\vec{x}) & \text{Si } n = 0 \\ h(f(\vec{x}, n-1), \vec{x}, n-1) & \text{Sinon} \end{cases}$$

Soient M_g et M_h les machines qui calculent respectivement g et h (et ne s'arrêtent pas sur les données hors des domaines de définition). On définit M_f comme suit : $\vec{x}, n$ sont donnés sur le ruban 0. Sur le ruban 1, M_f conserve un compteur (initialement 0). Sur le ruban 2, M_f simule M_g sur la donnée $\vec{x}$.
M_f répète ensuite les opérations suivantes tant que le compteur est inférieur strictement à n :

1. Simuler M_h sur le ruban3, avec les données fournies par le deuxième ruban, $\vec{x}$ (ruban 0) et la donnée du ruban 1
2. Recopier le contenu du ruban 3 sur le ruban 2
3. incrementer le ruban 1

— Pour la minimisation : soit $f(\vec{x}) = \min_n(g(\vec{x}, n) = 0)$ et soit M_g une machine qui calcule g (et ne s'arrête pas sur les valeurs pour lesquelles g n'est pas définie).
M_f a sur son premier ruban la donnée $\vec{x}$, sur un deuxième ruban un compteur (initialement 0). Sur le troisième ruban, M_f répète les opérations suivantes :

1. simuler M_g sur les données des deux premiers rubans
2. si le résultat est nul, alors s'arrêter (et le résultat est donné sur le deuxième ruban), sinon incrémenter le deuxième ruban.

Réciproquement, étant donnée une machine de Turing, on construit une fonction récursive primitive qui simule un mouvement de la machine, puis avec un coup de minimisation on termine. Plus précisément : les mots sont considérés comme des entiers en base $|Q| + |\Sigma| + 1$. Les configurations sont des triplets (q, w_1, w_2) et sont donc codés de manière bijective par des entiers supérieurs ou égaux à 2. Soit C ce codage.

La fonction f_0 qui, à $\vec{n}$ associe le codage de la configuration initiale de la machine sur la donnée $\vec{n}$ est récursive primitive (car l'exponentiation est récursive primitive).

La fonction g qui à un entier n associe associe $C(q', w_1', w_2')$ si $n = C(q, w_1, w_2)$, et $(q, w_1, w_2) \vdash_M (q', w_1', w_2')$ et 0 sinon est une fonction récursive primitive puisque quotient, reste, multiplication, somme et tests d'égalité sont récursives primitives et les fonctions récursives primitives sont closes par branchement conditionnel (quand la condition est elle-ême récursive primitive).

Enfin, la fonction f qui associe à un entier 0 ssi cet entier code une configuration finale de la machine est aussi récursive primitive et la fonction d qui associe au code d'une configuration finale l'entier résultat du calcul est aussi récursive primitive.

On définit alors

$$h(\vec{n}, k) = \begin{cases} f_0(\vec{n}) & \text{Si } k = 0 \\ g(h(\vec{n}, k-1)) & \text{Sinon} \end{cases}$$

qui calcule le codage de la nième configuration de la machine de Turing M dans son calcul sur la donnée $\vec{n}$.

La fonction

$$f_M(\vec{n}) = d(h(\vec{n}, \min_k(f(h(\vec{n}, k)) = 0)))$$

est alors récursive partielle. Son domaine de définition est l'ensemble des entiers sur lesquels M s'arrête. f_M calcule par construction la même fonction que M, sur son domaine de définition.

Corollaire 7.3.1 (Kleene) *Pour toute fonction récursive partielle à k arguments f, il existe deux fonctions récursives primitives g, h telles que, pour tous $x_1, \ldots, x_k$, $f(x_1, \ldots, x_k) = g(x_1, \ldots, x_k, \min_y(h(y, x_1, \ldots, x_k) = 0))$.*

Exercice 190
Montrer que si on remplace la définition de la minimisation (pour les fonctions partielles) par

$$g(\vec{x}) = \min\{n \mid f(n, \vec{x}) = 0 \& \forall m.\ f(m, \vec{x}) \neq \perp\}$$

(Autrement dit, h doit être définie pour tout m et pas seulement pour $m < n$) Alors on ne change pas la définition des fonctions récursives partielles.

Exercice 191 (5)
Montrer que le problème suivant est indécidable :

Donnée : une fonction récursive primitive f

Question : f calcule-t-elle la fonction nulle ?

Exercice 192 (5)
Montrer que, si f est une fonction récursive partielle croissante (dans tous ses arguments ; en particulier si f est indéfinie sur $x_1, \ldots, x_k$, et $y_i \geq x_i$, alors f est indéfinie sur $y_1, \ldots, y_k$), alors son graphe est récursif primitif.

7.3.2 Élimination de la récursion primitive

On veut montrer que la récursion primitive peut être simulée par les autres constructions.

On note $\mathcal{C}$ le plus petit ensemble de fonctions sur les entiers qui contient les fonctions initiales, l'addition, la multiplication, l'égalité, et qui est clos par composition et minimisation. (Donc pas de récursion primitive ici).

Exercice 193
Montrer que les fonctions ou prédicats suivants sont dans $\mathcal{C}$:

1. conjonction, disjonction, négation de prédicats dans $\mathcal{C}$
2. Si $P, f_1, f_2 \in \mathcal{C}$, $f = \lambda n.\text{if} P(n)$ then $f_1(n)$ else $f_2(n)$
3. La soustraction entière : $x \dot{-} y = \max(0, x - y)$
4. L'inégalité $x \geq y$
5. Les foncions J, K, L
6. $D(n, m)$ qui est satisfait si $n \neq 0$ et n divise m
7. $\text{rem}(x, y)$ qui calcule le reste de la division de x par y lorsque $y \neq 0$ (et est indéfini lorsque $y = 0$).
8. $\forall x \leq f(\vec{y}).P(x, \vec{y})$ et $\exists x \leq f(\vec{y}).P(x, \vec{y})$ où $P, f \in \mathcal{C}$.
9. $Prime(n)$ qui est satisfait par les nombres premiers
10. $PP(n)$: "n est une puissance d'un nombre premier".

Lemme 7.3.1 *Il existe une fonction $\beta \in \mathcal{C}$ (la fonction* beta de Gödel*) telle que, pour tout n et toute séquence $a_0, \ldots, a_n$, il existe un entier d tel que, pour tout $i = 0, \ldots, n$, $\beta(d, i) = a_i$.*

Preuve:
On pose $N = \max(n, a_0, \ldots, a_n)$ et $u_i = 1 + (i + 1)N!$. Alors $u_i > a_j$ et $\gcd(u_i, u_j) = 1$ pour $i \neq j$.

Le système d'équations $z = a_i \mod u_i$ a alors une unique solution b telle que $b < u_0 \times \ldots \times u_n$.

On pose enfin $d = J(b, N!)$ et $\beta(x, y) = \text{rem}(K(x), 1 + (y + 1)L(x))$

On vérifie aisément que $\beta \in \mathcal{C}$ à l'aide des exercices précédents.

De plus,

$$\beta(d, i) = \text{rem}(K(d), 1 + (i + 1)L(d)) = \text{rem}(b, 1 + (i + 1)N!) = \text{rem}(b, u_i) = a_i$$

Élimination de la récursion primitive :

Théorème 7.3.2 *$\mathcal{C}$ coïncide avec l'ensemble des fonctions récursives partielles.*

Preuve:
Comme l'addition, la multiplication et l'égalité sont facilement expressibles comme des fonctions récursives primitives, il suffit de montrer que $\mathcal{C}$ est clos par récursion primitive.

Soit

$$f(x,\vec{n}) = \begin{cases} \xi(\vec{n}) & \text{Si } x = 0 \\ \psi(f(x-1,\vec{n}), x-1, \vec{n}) & \text{Sinon} \end{cases}$$

Soit alors

$$S(x,b,\vec{n}) \stackrel{\text{def}}{=} \beta(b,0) = \xi(\vec{n}) \wedge \forall i < x.(\beta(b,i+1) = \psi(\beta(b,i),i,\vec{n}))$$

$S \in \mathcal{C}$ par l'exercice 193.

Montrons alors, par récurrence sur $z \leq x$, que, si $S(x,b,\vec{n}) = 1$, alors $\beta(b,z) = f(z,\vec{n})$.

Si $z = 0$, alors $\beta(b,z) = \xi(\vec{n}) = f(0,\vec{n})$. Sinon, $\beta(b,z) = \psi(\beta(b,z-1), z-1,\vec{n}) = \psi(f(z-1,\vec{n}), z-1,\vec{n})$ (par hypothèse de récurrence) et donc $\beta(b,z) = f(z,\vec{n})$.

De plus, d'après le lemme 7.3.1, si $f(x,\vec{n})$ est défini, il existe un b_0 tel que $S(x,b_0,\vec{n}) = 1$. Donc $\beta(\min_b(S(x,b,\vec{n}) = 1), x) = f(x,\vec{n})$.

Si maintenant $f(i,\vec{n})$ est indéfini. Si $i = 0$, $S(x,b,\vec{n})$ ne vaut jamais 1 et $f'(x,\vec{n})$ est indéfini. Si $i > 0$, alors $\psi(f(i-1,\vec{n}), i-1,\vec{n})$ est indéfini et donc $S(i,b,\vec{n})$ est indéfini. $f'(i,\vec{n})$ est alors indéfini (puisque $S(i,b,\vec{n}) \neq 1$ quel que soit b).

Dans tous les cas, $f(x,\vec{n}) = \beta(\min_b(S(x,b,\vec{n}) = 1), x)$. Donc $f \in \mathcal{C}$.