2. FONCTIONS RECURSIVES

La fonction d'Ackermann

2.1 Il s'agit dans cette sous-section de donner un exemple de fonction calculable au sens intuitif du terme, qui n'est pas récursive primitive, ce qui justifiera le travail supplémentaire demandé au lecteur dans la suite. La fonction que nous allons définir et que nous appellerons **la fonction d'Ackermann**, bien que ce soit une légère variante de la fonction originellement définie par Ackermann, est une fonction à deux variables que nous noterons ξ et qui est définie comme suit :

i) pour tout entier x, $\xi(0,x) = 2^x$;

ii) pour tout entier y, $\xi(y,0) = 1$;

iii) pour tous entiers x et y, $\xi(y+1,x+1) = \xi(y,\xi(y+1,x))$.

Pour chaque entier n, appelons ξ_n la fonction $\lambda x.\xi(n,x)$. Alors $\xi_0(x) = 2^x$, et on voit facilement, à partir de la clause iii) ci-dessus, que, pour tout n positif, ξ_n est définie par récurrence à partir de ξ_{n-1} par

$$\xi_n(0) = 1 \text{ et } \xi_n(x+1) = \xi_{n-1}(\xi_n(x)).$$

Cela montre d'abord qu'il y a une seule fonction ξ satisfaisant les conditions imposées, et de plus, que toutes les fonctions ξ_n sont récursives primitives (faire une récurrence sur n). En revanche, rien ne nous permet d'affirmer que la fonction ξ elle-même l'est, et c'est heureux car on va montrer qu'elle ne l'est pas. Pourtant, on peut effectivement calculer $\xi(x,y)$ pour n'importe quelles valeurs de x et y, comme le lecteur peut s'en convaincre facilement. Il nous faut maintenant montrer quelques lemmes faciles mais ennuyeux concernant cette fonction ξ.

2.2 LEMME 1 : *Pour tout* n *et pour tout* x, $\xi_n(x) > x$.

☺ On va utiliser un raisonnement faisant intervenir deux récurrences emboîtées : par récurrence sur n, on montre que, pour tout x, $\xi_n(x) > x$. C'est clair pour $n = 0$. Fixons $n > 0$ et supposons l'assertion

pour tout entier x, $\xi_{n-1}(x) > x$

vraie. On montre alors l'assertion

pour tout entier x, $\xi_n(x) > x$.

Pour cela, on fait maintenant une récurrence sur x. C'est clair pour $x = 0$ puisque $\xi_n(0) = 1$. On suppose donc $\xi_n(x) > x$ et on va montrer $\xi_n(x+1) > x+1$. On sait que $\xi_n(x+1) = \xi_{n-1}(\xi_n(x))$, et donc, par la première hypothèse de récurrence, on voit que :

$$\xi_n(x+1) > \xi_n(x) \text{ soit } \xi_n(x+1) \geqslant \xi_n(x) + 1.$$

Or, d'après la seconde hypothèse de récurrence, $\xi_n(x) > x$. Le lemme en découle.

☹

LEMME 2 : *Pour tout entier* n, *la fonction* ξ_n *est strictement croissante.*

☺ C'est clair pour n égal à 0. Ensuite, cela découle immédiatement du lemme 1 et de la formule $\xi_n(x+1) = \xi_{n-1}(\xi_n(x))$.

☹

LEMME 3 : *Pour tout* $n \geqslant 1$ *et pour tout* x, $\xi_n(x) \geqslant \xi_{n-1}(x)$.

☺ C'est clair pour $x = 0$. Pour $x+1$, puisque $\xi_n(x) \geqslant x+1$ et que ξ_{n-1} est croissante, $\xi_{n-1}(\xi_n(x)) \geqslant \xi_{n-1}(x+1)$, et il suffit d'appliquer la formule

$$\xi_n(x+1) = \xi_{n-1}(\xi_n(x)).$$

☹

Si k est un entier, notons ξ_n^k la fonction ξ_n itérée k fois (c'est-à-dire $\xi_n^0 = \lambda x.x$, $\xi_n^1 = \xi_n$, et $\xi_n^{k+1} = \xi_n \circ \xi_n^k$). Le lemme suivant est une collection d'évidences :

LEMME 4 : *Les fonctions* ξ_n^k *sont toutes strictement croissantes. De plus, pour tous* m, n, k, h *et* x, $\xi_n^k(x) < \xi_n^{k+1}(x)$ *et* $\xi_n^k(x) \geqslant x$, $\xi_n^k \circ \xi_n^h = \xi_n^{k+h}$ *et, si* $m \leqslant n$, $\xi_m^k(x) \leqslant \xi_n^k(x)$.

2.3 Donnons maintenant une définition :

DEFINITION : *Soient* $f \in \mathfrak{F}_1$ *et* $g \in \mathfrak{F}_p$. *On dit que* f **domine** g *s'il existe un entier* A *tel que pour tout* $(x_1,x_2,\ldots,x_p)$, $g(x_1,x_2,\ldots,x_p) \leqslant f(\sup(x_1,x_2,\ldots,x_p,A))$.

En particulier, lorsque f est strictement croissante, f domine g si et seulement si $g(x_1,x_2,\ldots,x_p) \leqslant f(\sup(x_1,x_2,\ldots,x_p))$ sauf pour un nombre fini de p-uples $(x_1,x_2,\ldots,x_p)$.

Appelons C_n l'ensemble des fonctions qui sont dominées par au moins une itérée de ξ_n :

$$C_n = \{ g \ ; \text{ il existe } k \text{ tel que } \xi_n^k \text{ domine } g \}.$$

Il est bien clair que les fonctions suivantes appartiennent à C_0 : les fonctions projections P_p^i, les fonctions constantes, la fonction successeur S, la fonction $\lambda x_1 x_2 \ldots x_p . \sup(x_1, x_2, \ldots, x_p)$, la fonction $\lambda xy . x + y$ et les fonctions $\lambda x . kx$ où k est un entier quelconque. De plus, la fonction ξ_n appartient à C_n. D'autre part, si f et g appartiennent toutes deux à $\mathfrak{F}_p$, si $g \in C_n$ et si pour tous $x_1, x_2, \ldots, x_p$, $f(x_1, x_2, \ldots, x_p) \leqslant g(x_1, x_2, \ldots, x_p)$, alors $f \in C_n$. Nous allons montrer :

LEMME 5 : *Pour tout entier* n, *l'ensemble* C_n *est clos par composition.*

Soient $f_1, f_2, \ldots, f_m$ des fonctions à p variables de C_n et g une fonction à m variables de C_n. Il s'agit de montrer que $g(f_1, f_2, \ldots, f_m)$ est aussi dans C_n. On sait qu'il existe des entiers $A, A_1, A_2, \ldots, A_m, k, k_1, k_2, \ldots, k_m$ tels que, pour tous $y_1, y_2, \ldots, y_m$,

$$g(y_1, y_2, \ldots, y_m) \leqslant \xi_n^k(\sup(y_1, y_2, \ldots, y_m, A)),$$

et pour tous $x_1, x_2, \ldots, x_p$ et pour tout i compris entre 1 et m,

$$f_i(x_1, x_2, \ldots, x_p) \leqslant \xi_n^{k_i}(\sup(x_1, x_2, \ldots, x_p, A_i)).$$

Posons $B = \sup(A, A_1, A_2, \ldots, A_m)$ et $h = \sup(k_1, k_2, \ldots, k_m)$. En utilisant le lemme 4, on voit alors que, pour tous $x_1, x_2, \ldots, x_p$:

$$g(f_1(x_1, x_2, \ldots, x_p), f_2(x_1, x_2, \ldots, x_p), \ldots, f_m(x_1, x_2, \ldots, x_p)) \leqslant \xi_n^k(\xi_n^h(\sup(x_1, x_2, \ldots, x_p, B))),$$

et donc

$$g(f_1(x_1, x_2, \ldots, x_p), f_2(x_1, x_2, \ldots, x_p), \ldots, f_m(x_1, x_2, \ldots, x_p)) \leqslant \xi_n^{k+h}(\sup(x_1, x_2, \ldots, x_p, B)).$$

LEMME 6 : *Pour tous entiers* n, k *et* x,

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

Par récurrence sur k ; pour k égal à 0 ou 1, c'est clair. Si c'est vrai pour k, ça l'est pour $k + 1$:

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

(l'inégalité découle de l'hypothèse de récurrence, la dernière égalité de la définition de ξ).

LEMME 7 : *Soient* $g \in \mathfrak{F}_p$ *et* $h \in \mathfrak{F}_{p+2}$ *et on suppose de plus que* h *et* g *sont toutes deux dans* C_n $(n \geqslant 0)$. *Alors la fonction* f *définie par récurrence à partir de* g *et* h *appartient à* C_{n+1}.

Traduisons les hypothèses. Tout d'abord la définition de f :

$$f(x_1, x_2, \ldots, x_p, 0) = g(x_1, x_2, \ldots, x_p) ;$$

$$f(x_1, x_2, \ldots, x_p, y + 1) = h(x_1, x_2, \ldots, x_p, y, f(x_1, x_2, \ldots, x_p, y)) ;$$

ensuite les conditions de domination : il existe A_1, A_2, k_1, k_2 tels que, pour tous $x_1, x_2, \ldots, x_p, y, z$,

$$g(x_1, x_2, \ldots, x_p) \leqslant \xi_n^{k_1}(\sup(x_1, x_2, \ldots, x_p, A_1)) ;$$

$$h(x_1, x_2, \ldots, x_p, y, z) \leqslant \xi_n^{k_2}(\sup(x_1, x_2, \ldots, x_p, y, z, A_2)).$$

On va maintenant montrer par récurrence sur y que, pour tous $x_1, x_2, \ldots, x_p, y$:

$$(*) \qquad f(x_1, x_2, \ldots, x_p, y) \leqslant \xi_n^{k_1 + y k_2}(\sup(x_1, x_2, \ldots, x_p, y, A_1, A_2)).$$

C'est clair pour $y = 0$; si c'est vrai pour y, ça l'est pour $y + 1$:

$$f(x_1, x_2, \ldots, x_p, y + 1) = h(x_1, x_2, \ldots, x_p, y, f(x_1, x_2, \ldots, x_p, y)) ;$$

$$f(x_1, x_2, \ldots, x_p, y + 1) \leqslant \xi_n^{k_2}(\sup(x_1, x_2, \ldots, x_p, y, f(x_1, x_2, \ldots, x_p, y), A_2)).$$

Donc, en utilisant l'hypothèse de récurrence $(*)$ et le lemme 4,

$$f(x_1, x_2, \ldots, x_p, y + 1) \leqslant \xi_n^{k_2}(\xi_n^{k_1 + y k_2}(\sup(x_1, x_2, \ldots, x_p, y, A_1, A_2))),$$

ce qui démontre notre assertion ; maintenant, en utilisant le lemme 6, on obtient

$$f(x_1, x_2, \ldots, x_p, y) \leqslant \xi_{n+1}(\sup(x_1, x_2, \ldots, x_p, y, A_1, A_2) + k_1 + k_2 y).$$

Or la fonction $\lambda x_1 x_2 \ldots x_p y . \xi_{n+1}(\sup(x_1, x_2, \ldots, x_p, y, A_1, A_2) + k_1 + k_2 y)$ s'obtient par composition à partir de fonctions de C_{n+1} ; elle est donc elle-même dans C_{n+1}, de même que f.

Nous sommes maintenant en mesure d'énoncer :

2.4 COROLLAIRE : *L'ensemble* $\bigcup_{n \in \mathbb{N}} C_n$ *contient toutes les fonctions récursives primitives.*

En effet, d'une part cet ensemble contient les fonctions constantes, les fonctions projections et la fonction successeur ; d'autre part, il est clos par composition et pour les définitions par récurrence.

On en arrive au théorème principal de cette section :

THEOREME : *La fonction d'Ackermann n'est pas récursive primitive.*

☺ Raisonnons par l'absurde et supposons la fonction d'Ackermann récursive primitive ; il en est de même de la fonction $\lambda x.\xi(x,2x)$. Il existe donc des entiers n,k, et A tels que, pour tout $x > A$, $\xi(x,2x) \leqslant \xi_n^k(x)$. Donc pour tout $x > A$, on a :

$$\xi(x,2x) \leqslant \xi_n^k(x) \leqslant \xi_{n+1}(x+k) \quad \text{(lemme 6)},$$

et, si $x > \sup(A,k,n+1)$, $\xi_{n+1}(x+k) < \xi_{n+1}(2x) < \xi_x(2x) = \xi(x,2x)$ (lemme 4), ce qui est absurde.
☺

En fait, on peut voir que la fonction $\lambda x.\xi(x,x)$ domine toutes les fonctions récursives primitives.

Le schéma μ et les fonctions partielles récursives

2.5 Il nous faut donc définir une classe plus large, que nous appellerons la classe des fonctions récursives. On le fera en admettant un nouveau schéma de définition, le schéma μ (non borné). L'idée est la suivante : soit A un sous-ensemble de $\mathbb{N}^{p+1}$; alors la fonction $f \in \mathfrak{F}_p$ que ce schéma μ permet de définir est la fonction qui à $(x_1,x_2,\ldots,x_p)$ fait correspondre le plus petit entier z tel que $(x_1,x_2,\ldots,x_p,z) \in A$. On voit tout de suite la difficulté : que se passe-t-il s'il n'existe pas d'entier z tel que $(x_1,x_2,\ldots,x_p,z) \in A$? Il faut remarquer qu'il ne nous est pas possible de faire ce que l'on a fait pour le schéma μ borné et poser dans ce cas $f(x_1,x_2,\ldots,x_p) = 0$. En effet, en admettant, ce que l'on doit faire, que l'on dispose d'un algorithme permettant de calculer la fonction caractéristique χ_A de A, la seule façon imaginable de calculer $f(x_1,x_2,\ldots,x_p)$ est de calculer $\chi_A(x_1,x_2,\ldots,x_p,0)$, s'arrêter si on trouve 1, sinon calculer $\chi_A(x_1,x_2,\ldots,x_p,1)$, etc., jusqu'à tomber sur la valeur 1. Mais si pour tout entier z, $(x_1,x_2,\ldots,x_p,z) \notin A$, alors le processus ne s'arrête pas, et on ne connaîtra jamais la valeur de $f(x_1,x_2,\ldots,x_p)$. Autrement dit, on ne dispose pas d'algorithme pour calculer f, et on ne peut donc pas admettre ce schéma. Une possibilité serait de le restreindre au cas où, pour tout $(x_1,x_2,\ldots,x_p)$, il existe z tel que $(x_1,x_2,\ldots,x_p,z) \in A$ (on appellera ce schéma le **schéma μ total**). On obtiendrait, comme on le verra par la suite, toutes les fonctions récursives. Mais il est préférable de définir les fonctions récursives partielles ; la raison en est que les théorèmes d'énumération et de points fixes (théorème 3.18 et 4.14 de ce chapitre), essentiels dans cette matière, ne sont vrais que pour cette dernière classe (voir exercice 22).

Exercice 2 :

Soit $f : \mathbb{N}^p \longrightarrow \mathbb{N}$. On dit que f est calculable par machine s'il existe une machine M dont l'alphabet d'entrée est $[0,9] \cap \mathbb{N} = \Sigma$, à p bandes d'entrées, une bande de sortie, et que sur une entrée $\vec{n}$, M s'arrête et accepte exactement sur $f(\vec{n})$ en sortie.

Montrons que les fonctions récursives sont calculables par machine, et on aura :

L récursivement énumérable
$\Rightarrow L$ Turing reconnaissable.

- zero est calculable par :
  $\langle \Sigma, \Sigma_\square, \{0,1\}, (\Sigma_\square \times \{0\}) \times (\{0\} \times \{1\}), 0, 1 \rangle$
- Une machine pour succ. a été donnée dans la série 2.
- $Proj^p_i$ est la fonction de transition d'une machine calculant $Proj^p_i$.

Montrons que les fonctions Turing-calculables sont closes par composition, récursion et minimalisation.

- Composition : par une induction simple, il suffit de montrer que $(g$ et h T.-calculables$) \Rightarrow (g \circ h$ T.-calculable$)$
  Soit $g : \mathbb{N} \longrightarrow \mathbb{N}$, $h : \mathbb{N}^p \longrightarrow \mathbb{N}$, Γ_g et Γ_h calculant respectivement g et h.
  Soit $\Gamma_{g \circ h}$ la machine à p bandes d'entrée, une bande de travail et une de sortie, qui fait tourner Γ_g sur les p entrées, notant la sortie sur la bande de travail, puis Γ_h avec la bande de travail en entrée, notant sur la sortie. On a bien que $\Gamma_{g \circ h}$ calcule $g \circ h$.

• récursion : Soit $g : \mathbb{N}^p \to \mathbb{N}$, $h : \mathbb{N}^{p+2} \to \mathbb{N}$,
et Π_g et Π_h calculant g et h respectivement.
$\Pi_{rec(g,h)}$ est la machine suivante :
$p+1$ bandes d'entrée, 2 bandes de travail $(p+2)$, $(p+3)$
1 bande de sortie $(p+4)$. Le calcul est :

- On lance Π_g avec $(1) \to (p)$ en entrée, $(p+4)$ en sortie
- écrire 0 sur $(p+2)$
(*) comparer $(p+2)$ et $(p+1)$ si il y a égalité, accepter.
- sinon appliquer Π_{succ} à $(p+2)$
- ~~lancer~~ recopier $(p+4)$ sur $(p+3)$
- effacer $(p+4)$
- lancer Π_h avec $(1) \to (p)$, $(p+2)$, $(p+3)$ en entrée,
$(p+4)$ en sortie
- revenir à (*)

• minimalisation : Soit $g : \mathbb{N}^{p+1} \to \mathbb{N}$ calculée par Π_g
$\Pi_{min(g)}$ est : p bandes d'entrée $(1) \to (p)$, 1 bande de sortie $(p+1)$, une bande de travail $(p+2)$

- écrire 0 sur $(p+1)$
(*) lancer Π_g avec $(1) \to (p+1)$ en entrée, $(p+2)$ en sortie.
- si $(p+2) = 0$, accepter.
- sinon lancer Π_{succ} sur $(p+1)$
- revenir à (*)

Montrons maintenant que l'on a :

L T-reconnaissable $\Rightarrow$ L récursivement énumérable

Pour cela, il faut trouver une fonction récursive dont le domaine est précisément L.

La fonction conçue doit converger lorsque la machine s'arrête, diverger ailleurs.
Posons, par commodité, M une machine reconnaissant L.
Voici l'idée :
M étant un objet fini, composé d'ensembles finis ou d'individus (états, fonction de transition --), on peut trouver (au moins) un codage de M vérifiant que "être un état" ; "être un symbole de travail" ; "être l'image par la fonction de transition" ; etc. ne soient que des notions récursives primitives.
En effet, on pourrait à la rigueur écrire toutes ces notions comme des combinaisons booléennes de possibilités de codes.
Dans la mesure où "être une suite finie d'entiers" est une notion récursive primitive, "être une configuration" l'est aussi, et puisque concat est récursive primitive, "être deux configurations liées" l'est aussi.
Alors la fonction $g_M(\vec{n}, t)$, définie par récurrence, renvoyant le code de la configuration obtenue après un calcul de longueur t sur l'entrée $\vec{n}$ par la machine M, est récursive primitive.
La fonction recherchée est alors :

$$\mu t \left(\text{" } g_M(\vec{n}, t) \text{ est une configuration acceptante "} \right)$$

Montrons les équivalences suivantes.

D'après ce qui précède, on a $(1) \Leftrightarrow (2)$.

Par ailleurs, on a de façon quasi-définitionnelle que $(3) \Rightarrow (2)$ et $(4) \Rightarrow (1)$. En effet,

$(3) \Rightarrow \mathbb{1}_L$ récursive $\Rightarrow \mathbb{1}_L|_L$ et $\mathbb{1}_{\{0\}^*} - \mathbb{1}_L|_{L^c}$ récursives (car L et L^c le sont)

$\Rightarrow L$ et L^c récursivement énumérables.

$(4) \Rightarrow \exists \Pi = \langle Q, \Sigma, \Gamma, \delta, q_0, q_+, q_- \rangle$ reconnaissant L,
$\overline{\Pi} = \langle Q, \Sigma, \Gamma, \delta, q_0, q_-, q_+ \rangle$ reconnaît L^c $\Rightarrow (1)$.

$(1) \Rightarrow 4$: M reconnaît L, Π' reconnaît L^c.

Soit Π'' à deux bandes, sur la première tourne M, et simultanément Π' sur la deuxième. On pose que Π'' s'arrête si et seulement si Π ou Π' s'arrête, et accepte si Π accepte ou Π' rejette, et rejette dans l'autre cas.

on a $x \in L$ ou $x \in L^c \Rightarrow M$ accepte ou Π' accepte

$\Rightarrow \Pi''$ s'arrête et décide L.

$\Rightarrow (4)$.

$(2) \Rightarrow (3)$ Soit L et $L^c \subseteq \mathbb{N}^k$ récursivement énumérables.

Soit (f, f') deux fonctions récursives de domaines L et L^c respectivement.

D'après ce qui précède, on a $f(\vec{n}) = \mu m \, (R(\vec{n}, m))$, avec R récursive primitive. Aussi, en posant $G_f(\vec{n}, m)$ ssi $\forall x < m \to R(\vec{n}, x) \wedge R(\vec{n}, m)$, on a que le graphe G_f de f est récursif. De même pour $G_{f'}$.

Posons $g(\vec{n}) = \mu m \left(G_f(\vec{n}, m) \vee G_{f'}(\vec{n}, m) \right)$

g est récursive totale, donc $\mathbb{1}_L = \begin{cases} 1 \text{ si } G_f(\vec{n}, g(\vec{n})) \\ 0 \text{ sinon} \end{cases}$ aussi.

Exercice 3 :

① Soit H une machine de Turing, ayant pour entrée $(\text{code}(T), \text{code}(u))$ un couple d'entiers, et telle que H s'arrête et accepte si T s'arrête sur u, et H s'arrête et rejette si T ne s'arrête pas sur u.
(Ainsi H est un décideur pour le problème de l'arrêt)
Soit alors la machine H', ayant pour entrée des entiers du type $\text{code}(T)$, et telle que
H' s'arrête sur $\text{code}(T)$ ssi H rejette sur $(\text{code}(T), \text{code}(T))$
On a alors :
$H'(\text{code}(H'))$ s'arrête ssi H rejette $(\text{code}(H'), \text{code}(H'))$
ssi $H'(\text{code}(H'))$ ne s'arrête pas.
– contradiction –

$\Rightarrow L_H$ n'est pas T- décidable.

② Soit U une machine de Turing universelle, c'à d qui sur une entrée $(\text{code}(T), \text{code}(u))$:
. s'arrête ssi T s'arrête sur u, et accepte ssi T accepte u.
Posons U' la machine qui est exactement la même que U, sauf que U' accepte si U s'arrête, ne s'arrête pas sinon. Par définition de U, U' reconnaît L_H.

③ L_H^c T- reconnaissable $\wedge$ L_H T- reconnaissable
$\Rightarrow L_H$ T décidable.
Par contraposition, et d'après ①, on a donc
L_H^c non T- reconnaissable ou L_H non T-reconnaissable.
Par ②, on a enfin L_H^c non T- reconnaissable.