912 — Fonctions récursives primitives et non primitives

Introduction : Les fonctions récursives forment un modèle intuitif de ce que l'on peut effectivement calculer avec un ordinateur

I Fonctions primitives récursives [WOL]

On se restreint à des fonctions à arguments et valeurs dans $\mathbb{N}$.

1) Définitions

* Fonctions de base : Les fonctions primitives récursives de base sont :

- $0() \to \mathbb{N}$, $\to 0$
- Les fonctions du type $\Pi_i^k : \begin{array}{l} \mathbb{N}^k \longmapsto \mathbb{N} \\ n_1, \dots n_k \mapsto n_i \end{array}$ $(1 \leq i \leq k)$
- La fonction successeur $\sigma : \begin{array}{l} \mathbb{N} \\ n \end{array} \begin{array}{l} \\ \longmapsto \end{array} \begin{array}{l} \mathbb{N} \\ n+1 \end{array}$

* Règle de composition

Soit $g : \mathbb{N}^l \to \mathbb{N}$, $h_1, \dots h_l : \mathbb{N}^k \to \mathbb{N}$ alors la composition de g par $h_1, \dots h_l$ est :

$f : \begin{array}{l} \mathbb{N}^k \longrightarrow \mathbb{N} \\ \bar{n} \longmapsto g(h_1(\bar{n}), \dots h_l(\bar{n})) \end{array}$

* Règle de récursion primitive

Soit $g : \mathbb{N}^k \to \mathbb{N}$, $h : \mathbb{N}^{k+2} \to \mathbb{N}$ alors la fonction définie par récursion primitive à partir de g et h est :

$f \begin{array}{l} \mathbb{N}^{k} \longrightarrow \mathbb{N} \\ (\bar{n}, 0) \longmapsto g\,\bar{n} \\ (\bar{n}, m+1) \longmapsto h(\bar{n}, m, f(\bar{n}, m)) \end{array}$

* Fonctions récursives primitives :

Les fonctions récursives primitives sont :

- les fonctions de base
- toutes les fonctions obtenues à partir des fonctions de base par un nombre quelconque d'application des deux règles précédentes

Théorème : Les fonctions récursives primitives sont calculables dans un langage de programmation donné

2) Exemples

- $j() : \begin{array}{l} \emptyset \to \mathbb{N} \\ \ \ \longmapsto j = \overbrace{\sigma \circ \sigma \circ \cdots \circ \sigma}^{j \text{ fois}} (0()) \end{array}$
- $Id : \begin{array}{l} \mathbb{N} \longrightarrow \mathbb{N} \\ x \longmapsto \Pi_1^1(x) \end{array}$
- plus $\mathbb{N}^2 \longrightarrow \mathbb{N}$

  $(n_1, 0) \longmapsto Id(n_1)$

  $(n_1, n_2+1) \longmapsto \sigma(\Pi_3^3(n_1, n_2, plus(n_1, n_2)))$
- produit, somme et factorielle sont aussi primitives récursives
- pred $\begin{array}{l} \mathbb{N} \longrightarrow \mathbb{N} \\ 0 \longmapsto 0 \\ n+1 \longmapsto \Pi_1^2(n, pred\ n) \end{array}$
- $\dot{-}$ $\begin{array}{l} \mathbb{N}^2 \longrightarrow \mathbb{N} \\ n, 0 \longmapsto n \\ (n, m+1) \longmapsto pred(\Pi_3^3(n, m, n \dot{-} m)) \end{array}$
- Le produit borné par n d'une fonction primitive récursive est primitif récursif.

APPEL de FONCTION

↳ récurrence double

FOR

3) Prédicats primitifs récursifs

Définition : · Un prédicat A sur $\mathbb{N}^k$ est un sous-ensemble de $\mathbb{N}^k$.
· Il est dit primitif récursif si 1_A est primitive récursive

Exemples : · $n < m$ est primitif récursif
· les prédicats primitifs sont stables par opérations booléennes
· $n = m$ est primitif récursif

Def + Prop : Quantification bornée : Soit $P \subset \mathbb{N}^{k+1}$ primitif récursif

Alors $\{(\bar{n}, m) \mid \exists i \leq m, (\bar{n}, i) \in P\}$ est primitif récursif

On peut avec ces prédicats définir plus facilement certaines fonctions primitives récursives

Def - Prop : Fonctions primitives récursives définies par cas
Soit $g_1, \dots g_p\ \mathbb{N}^k \to \mathbb{N}$, $A_1, \dots A_p \subset \mathbb{N}^k$ primitifs récursifs
tq $A_i \cap A_j = \emptyset$ si $i \neq j$. Soit $f\ \mathbb{N}^k \to \mathbb{N}$ défini par :

$$f(\bar{n}) = \begin{cases} g_1(\bar{n}) & \text{si } \bar{n} \in A_1 \\ \vdots \\ g_p(\bar{n}) & \text{si } \bar{n} \in A_p \end{cases}$$ alors f est primitive récursive.

Def - Prop : Minimisation bornée : Soit $A \subset \mathbb{N}^{k+1}$ primitif récursif
alors $f(\bar{n}, m) = \begin{cases} \text{le plus petit } i \leq m \text{ tq } \bar{n}, i \in A \\ 0 \text{ si il n'existe pas.} \end{cases}$
est primitive récursive

4) Les limites du modèle

Théorème | il existe des fonctions calculables effectivement qui ne sont pas primitives récursives

Exemples : · Soit $(f_n)_{n \in \mathbb{N}}$ une énumération effective des FPR.
Soit $g\ \mathbb{N} \to \mathbb{N}$ défini par $g(n) = f_n(n)$.
alors g n'est pas primitive récursive, mais on a un algorithme

· La fonction d'Ackermann, définie par :
$\forall m\ A(0, m) = m+1$, $\forall n$, $A(n+1, 0) = A(n, 1)$
$\forall n, m\ A(n+1, m+1) = A(n, A(n+1, m))$

Notation : On pose $A_n(x) = A(n, x)$, $A_n^k(x) = \underbrace{A_n \circ \dots \circ A_n}_{k \text{ fois}}(x)$

Propriétés : · $\forall n\ A_n$ strictement croissante
· $\forall k > 1$, $m \leq n \Rightarrow A_m^k(x) \leq A_n^k(x)$
· $A_n^k(x) \leq A_{n+1}(x+k)$

Théorème : La fonction d'Ackermann n'est pas primitive récursive

OVPT1

II Fonctions μ-récursives et calculabilité [WOL]

1) μ-récursivité et récursivité partielle

Définition : * Un prédicat est dit sûr si $\forall \bar{n} \in \mathbb{N}^k$, $\exists i \in \mathbb{N}$, $(\bar{n}, i) \in A$

* La minimisation non bornée d'un prédicat $A \subset \mathbb{N}^{k+1}$ est une fonction partielle définie par : $\mu_i A(\bar{n}, i) =$
$\mu_i(A)(\bar{n}) :=$ le plus petit i tq $(\bar{n}, i) \in A$

* les fonctions partielles récursives sont les fonctions obtenues à partir des fonctions de base et des minimisations non bornées en appliquant les règles de composition et récursion primitive

* les fonctions μ récursives s'obtiennent de même en restreignant l'usage des minimisations non bornées aux prédicats sûrs

WHILE

[COR] [WOL] p150

2) Fonctions calculables par MT

Définition : * Une machine de Turing M calcule une fonction partielle f si, pour tout mot d'entrée tel que l'exécution s'arrête, $f(w)$ est sur le ruban à l'issue de l'exécution. L'ensemble des tels w est le domaine de définition de f.

* On dit que f est partiellement calculable s'il existe une machine M qui la calcule

* On dit que f est calculable si de plus l'exécution de M s'arrête pour tout mot w

OVP T2 **Théorème** : Les fonctions calculables par MT sont μ-récursives

Théorème : Les fonctions calculables par MT sont exactement les fonctions μ-récursives (à un codage effectif près)

Remarque : On a le même résultat dans le cas partiel.

Conséquence : Les fonctions μ-récursives sont calculables sur un ordinateur

> Thèse de Turing Church : Les fonctions calculables de manière effective par une machine sont exactement les fonctions μ-récursives

P29 [COR] [LAL] p314

III Ensembles récursifs et récursivement énumérables

Déf : Soit $A \subset \mathbb{N}^p$. A est **récursif** si sa fonction caractéristique est μ-récursive

* A est **récursivement énumérable** si c'est le domaine de définition d'une fonction partielle récursive

Notations : On note R l'ensemble des ensembles récursifs et RE l'ensemble des ensembles récursivement énumérables

Exemple : · $\{2n, n \in \mathbb{N}\} = D(\mu_x(2x = n)) \in RE$

· Tout prédicat primitif récursif appartient à R

Propriétés : RE est stable par union et intersection

Théorème : $A \in R \iff A \in RE$ et $\bar{A} \in RE$

Propriété : Soit $A \subset \mathbb{N}$, les conditions suivantes sont équivalentes :

i) $A \in RE$

ii) A est l'image d'une fonction primitive récursive de $\mathbb{N}$ dans $\mathbb{N}$ (ou $A = \emptyset$)

iii) A est l'image d'une fonction partielle récursive de $\mathbb{N}$ ~~dans~~ $\mathbb{N}$

(iv) A est la projection d'un ensemble $B \subset \mathbb{N}^2$ μ-récursif

OVP T3

[WOL] Wolper, Introduction à la calculabilité

[COR] Cori Lascar, Logique mathématique II

[LAL] Lalement, Logique Réduction - Résolution

Cori-Lascar T2 $\boxed{\text{Ensembles RE}}$ Lallement

Propositions: On a l'équivalence, pour $A \subset \mathbb{N}$:

(1) $A \subset RE$

(2) $A = \phi$ OU A est l'image d'une fonction récursive primitive.

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

(4) A est la projection de $B \subset \mathbb{N}^2$, avec B récursif primitif.

Preuve:

(1) $\Rightarrow$ (2): Si $A \neq \phi$, il existe $n \in A$.

Soit $\varphi: A \to \mathbb{N}$ partielle récursive.

Alors il existe une machine de Turing qui calcule φ et qui bloque ou boucle si son entrée n'est pas dans A. On la note M_φ

On pose $B(t, x) =$ "M_φ s'arrête en t étapes sur x"

On considère une bijection récursive primitive de $\mathbb{N}$ et $\mathbb{N}^2$:

$k \mapsto (t_k, x_k)$.

Alors $\{k \in \mathbb{N} \mid B(t_k, x_k)\}$ est récursif primitif

On pose $f: \begin{array}{l} \mathbb{N} \to A \\ k \mapsto x_k \text{ si } B(t_k, x_k). \\ k \mapsto n \text{ sinon} \end{array}$

$B(t_k, x_k)$ est primitif récursif, donc f l'est.

De plus, on a bien $\mathrm{Im} f = A$, car $x \in A \Leftrightarrow \exists t\, B(x, t)$

(2) $\Rightarrow$ (3). Si $A = \phi$, $A = \mathrm{Im}(y \mapsto \mu x\,(x+1=0)) = \phi$.

Sinon A est l'image d'une fonction primitive récursive, donc d'une fonction μ-récursive partielle

$(3) \Rightarrow (4)$ Soit f μ-récursive partielle telle que $\operatorname{Im} f = A$.

$\exists M_i$ une MT qui calcule f.

On définit une famille de prédicats :

$$C(t, x, a) = \text{“} M_i \text{ s'arrête en } t \text{ étapes sur } a \text{ à partir de } x \text{”}.$$

Ces prédicats sont primitifs récursifs par minimisation bornée

On pose $G = \{ (k, a) \mid C(t_k, x_k, a) \}$, qui est récursif primitif

Alors $A = \pi_2^2(G)$.

$(4) \Rightarrow (1)$ Si A est la projection de B récursif primitif

On pose $f : x \mapsto \mu y\, ((x, y) \in B)$, qui est une fonction partielle μ-récursive.

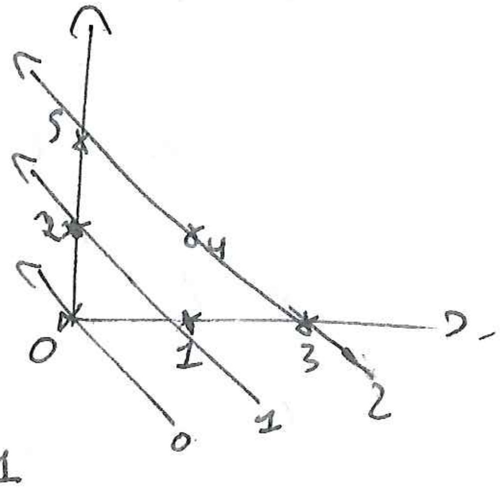
$$x \in \text{Domaine}(f) \Leftrightarrow \exists y,\ (x, y) \in B \Leftrightarrow x \in A$$

Donc $A \subset RE$.

$\square$

Lemme: Bijection récursive primitive de $\mathbb{N}$ à $\mathbb{N}^2$.

→ On va coder la bijection classique :



de $\mathbb{N}$ vers $\mathbb{N}^2$:

$$\text{diagonale}(n) = \left[\mu i \leq n \left(\frac{i(i+1)}{2} > n \right) \right] - 1$$

$$\text{écart}(n) = n - \frac{\text{diagonale}(n)\,(\text{diagonale}(n)+1)}{2}$$

$f : n \mapsto (\text{diagonale}(n) - \text{écart}(n),\ \text{écart}(n))$ est bien récursive primitive

de $\mathbb{N}^2$ vers $\mathbb{N}$: $g : (n, m) \mapsto \dfrac{(n+m)(m+n+1)}{2} + m$ est bien récursive primitive,

et $f \circ g = g \circ f = Id$.

Wolper

$\boxed{\text{Turing} \Rightarrow \text{Ect récursives}}$.

Théorème: Toute fonction calculable par machine de Turing est μ-récursive

Preuve: On considère une fonction $f_M : \Sigma^* \to \Sigma^*$ calculée par une machine de Turing M.

Remarque: + On suppose que l'on dispose d'un encodage des éléments de $\Sigma \cup \Gamma$ en les éléments de $\mathbb{N}$. Exemple: codage en base $|\Sigma \cup \Gamma|$

+ Ceci peut se faire sans perte de généralité

+ On note $cod : \Sigma^* \to \mathbb{N}^*$ ce codage.

Objectif: Trouver f μ-récursive telle que $f_M = cod^{-1} \circ f \circ cod$

Pour cela, on va introduire des fonctions reproduisant le comportement de la machine.

Une configuration de M est un triplet (q, α_1, α_2), q l'état, α_1 le mot avant la tête de lecture, α_2 le mot après la tête (lettre sous la tête incluse).

En la notant (α_1, q, α_2) on obtient, en numérotant les états de M, un triplet d'entiers.

On définit,

⊛ init : à un mot d'entrée codé par x, associe la configuration initiale associée, c'est à dire le codage de $(, q_0, x)$

init : $x \mapsto (cod(\varepsilon), \text{numéro}(q_0), x)$

C'est une fonction μ-récursive par composition.

⊛ config_suivante : $c \mapsto$ la configuration de M une étape après c.

C'est la disjonction d'un nombre fini de transitions, chacune étant μ-récursive, c'est donc une fonction μ-récursive

On pourra prendre par convention que la configuration suivant une configuration acceptante est elle-même.

⊛ config : $(x,0) \mapsto x$
$(x,n) \mapsto$ config_suivante(config(x,n)).

Cette fonction donne la configuration après n étapes à partir de x.
→ Elle est μ-récursive par récursion primitive.

⊛ stop : $c \mapsto 1$ si q final
$c \mapsto 0$ sinon

stop est un <u>prédicat</u> ne dépendant que de l'état, qui appartient à un ensemble fini.

→ C'est une fonction μ-récursive.

⊛ sortie associe à une configuration finale la valeur calculée par M.

→ μ-récursive

Le point délicat est de calculer le nombre d'étape nécessaire pour que M soit en configuration optimale.

Comme M calcule f_M, M s'arrête toujours, et donc le prédicat stop(config(init(x), i)) est sûr.

En posant $f : x \mapsto$ sortie(config(init(x), μi stop(config(init(x), i)))) on obtient une fonction μ-récursive par composition et minimisation non-bornée, qui correspond bien à f_M.

□

Cori-Lascar T2 (adapté)

$\boxed{\text{Ackermann}}$

Le Rey

Def:

$$\begin{cases} A(0,m) = m+1 \\ A(k+1,0) = A(k,1) \\ A(k+1,m+1) = A(k, A(k+1,m)) \end{cases} \qquad (k,m) \in \mathbb{N}$$

A est clairement calculable par procédure effective, mais:

Thm: A n'est pas primitive récursive

On commence par une série de propriétés:

On note $A_n : x \mapsto A(n,x)$

$$\begin{cases} A_0(x) = x+1 \\ A_n(0) = 2 \uparrow^{(n-2)} 3 - 3 \\ A_n(x+1) = A_{n-1}(A_n(x)) \end{cases}$$

Lemme 1: $\forall x, n \in \mathbb{N}, \quad A_n(x) > x$.

Preuve: Par rec sur n: H_n: " $\forall x \in \mathbb{N}, \ A_n(x) > x$ "

* H_0 car $A_0(x) = x+1 > x$
* Soit H_{n-1} pour n fixé. On va montrer H_n par récurrence sur x:
  + $A_n(0) > 0$ est toujours vraie.
  + Si $A_n(x-1) > x-1$ pour x fixé.

$A_n(x) = A_{n-1}(A_n(x-1))$

$A_n(x) > A_n(x-1)$ par hypothèse de la récurrence sur n.

$A_n(x) \geq A_n(x-1)+1$

$A_n(x) > x-1+1$ par hypothèse de rec. sur x.

$A_n(x) > x$.

D'où H_n par récurrence sur x

D'où le lemme 1 par récurrence sur n. ◇

Lemme 2: $\forall n$, A_n est strictement croissante

Preuve: A_0 est str. croissante.

Pour $n \geq 1$, $A_n(x+1) = A_{n-1}(A_n(x))$

$A_n(x+1) > A_n(x)$ } par lemme 1

$\lozenge$

Lemme 3: $\forall n \geq 1$, $\forall x$, $A_n(x) \geq A_{n-1}(x)$

Preuve: Par récurrence sur x.

* $2\uparrow^{(n-1)}3 \geq 2\uparrow^{(n-2)}(3)$, donc vrai pour $x = 0$.

* $A_n(x) \geq x+1$ par L1, et A_{n-1} croissante par L2:

$$A_{n-1}(A_n(x)) \geq A_{n-1}(x+1)$$

$$A_n(x+1) \geq A_{n-1}(x+1).$$

D'où le résultat par réc. sur x.

$\lozenge$

En notant A_n^k les itérés de A_n, on a facilement par récurrence:

Prop: $A_n^k(x) < A_n^{k+1}(x)$ (1)

$A_n^k(x) \geq x$ (2)

$m \leq n \Rightarrow A_m^k(x) \leq A_n^k(x)$ (3)

$A_n^k(x) \leq A_{n+1}(x+k)$ (4)

Preuve du théorème:

On va montrer que $A(x, 2x)$ croît plus vite que toute fonction récursive, et aboutir à une contradiction.

Déf: Pour $g : \mathbb{N}^p \to \mathbb{N}$, et $f : \mathbb{N} \to \mathbb{N}$ strictement croissante, on dit que f domine g si:

$\forall (x_1, \dots, x_p) \subset \mathbb{N}^p$, $g(x_1, \dots, x_p) \leq f(\sup_i (x_i))$

On pose $C_n = \{g \mid g \text{ dominée par } A_n^k \text{ pour un certain } k\}$

et $C = \bigcup_n C_n$. Rem: $C_{n-1} \subset C_n$.

Nous allons montrer que toutes les fonctions récursives primitives sont dans C.

(*) les projections, la fonction $O()$ et successeur sont dans C_0.

(*) Montrons que C_n est clos par composition.

$f_1, \dots, f_m : \mathbb{N}^p \to \mathbb{N}$, $g : \mathbb{N}^m \to \mathbb{N}$ dans C_n.

Alors, il existe $k_1, \dots, k_m$ et k tels que :

$\forall x_1, \dots, x_p, y_1, \dots, y_m$

$$g(y_1, \dots, y_m) \leq A_n^k(\sup_j(y_j))$$
$$f_i(x_1, \dots, x_p) \leq A_n^{k_i}(\sup_j(x_j))$$

$$g(f_1(x_1, \dots, x_p), \dots, f_m(x_1, \dots, x_p)) \leq A_m^k(\sup_j(f_j(x_1, \dots, x_p)))$$

$$\leq A_n^k(A_n^h(\sup_j(x_i))) \quad \text{par (1)}$$

avec $h = \sup_{i=1 \dots m} k_i$

$$\leq A^{k+h}(\sup(x_i)).$$

Donc C_n est stable par composition, et donc C aussi.

(*) Montrons que si $g : \mathbb{N}^p \to \mathbb{N}$ et $h : \mathbb{N}^{p+2} \to \mathbb{N}$ st ds C_n, alors f obtenue par récurrence sur g et h est dans C_{n+1} :

$\exists k_1, k_2 \in \mathbb{N}, \forall x_1, \dots, x_p, y, z$:

$$g(x_1, \dots, x_p) \leq A_n^{k_1}(\sup(x_i))$$
$$h(x_1, \dots, x_p, y, z) \leq A_n^{k_2}(\sup(x_i, y, z))$$

Montrons par récurrence H_y : "$f(x_1, \dots, x_p, y) \leq A_n^{k_1 + y k_2}(\sup(x_i, y))$"

(+) $y = 0$ $\quad f(x_1, \dots, x_p, 0) = g(x_1, \dots, x_p) \leq A_n^{k_1}(\sup(x_i))$.

(+) Si H_y pour y fixé.

$$f(x_1, \dots, x_p, y+1) = h(x_1, \dots, x_p, y, f(x_1, \dots, x_p, y))$$

$$\leq A_n^{k_2}(\sup(x_i, y, f(x_1, \dots, x_p, y))) \quad \text{par hyp. de réc.}$$

$$\leq A_n^{k_2}(\sup(x_i, y, A^{k_1 + y k_2}(\sup(x_i, y))))$$

$f(x_1, \dots, x_p, y) \leq A_n^{k_2}(A_n^{k_1+yk_2}(\sup(x_i, y)))$ par prop (2)

d'où H_{y+1}.

On a H_y pour tout y par récurrence.

$$\left.\begin{aligned} f(x_1, \dots, x_p, y) &\leq A_n^{k_1+yk_2}(\sup(x_i, y)) \\ &\leq A_{n+1}(\sup(x_i, y) + k_1 + y k_2) \end{aligned}\right\} \text{ par prop (4)}$$

$(x_1, \dots, x_p, y) \mapsto A_{n+1}(\sup(x_i, y) + k_1 + y k_2)$ est dans C_{n+1} par composition, donc f l'est aussi.

On obtient que C est stable par récursion.

$\Rightarrow$ Par induction, <u>C contient toutes les fonctions récursives primitives.</u>

Supposons que $A \in C$.

Alors $x \mapsto A(x, 2x)$ est dans C aussi.

Soient n et $k \in \mathbb{N}$

$\forall x \in \mathbb{N}, \quad A(x, 2x) \leq A_n^k(x)$

Par prop (4), on a $A(x, 2x) \leq A_{n+1}(x + k)$

Pour $x > k$, $\quad A_{n+1}(x+k) < A_{n+1}(2x)$ par L_2

Pour $\overset{x>k}{x} > n+1$, $\quad A_{n+1}(x+k) < A_x(2x) = A(x, 2x)$.

Donc, pour tout $x \in [\![\max(k+1, n+2), +\infty[\![$, on a :

a que $A(x, 2x) < A(x, 2x)$.

Absurde.

Donc A n'est pas récursive primitive.

<u>Remarque:</u> On a en quelque sorte donné des classes de croissances pour les fonctions récursives primitives.