Fonctions récursives primitives et non primitives. Exemples.

912

Pré-requis : machine de Turing + langages acceptés et décidés par une machine de Turing + fonctions calculées par une machine de Turing.

Définition : [2, p. 145] Un modèle de calcul est un système qui permet d'associer à une entrée x une sortie y en appliquant un nombre fini d'opérations élémentaires.

I - Les fonctions récursives primitives.

A - Syntaxe et sémantique

Définition : [3, p. 121] Le langage $\mathcal{L}_{PR}$ des fonctions primitives récursives est défini par induction :
* $\mathbb{0}$ est une expression d'arité 0 ;
* $\mathbb{S}$ est une expression d'arité 1 ;
* Π_i^m où $m \in \mathbb{N}^*$ et $i \in \{1, \dots, m\}$ est d'arité m ;
* Si F est une expression d'arité m et $G_1, \dots, G_m$ sont des expressions d'arités p alors : $\circ(F, G_1, \dots, G_m)$ est une expression d'arité p ;
* Si F est une expression d'arité m et G est une expression d'arité $m+2$, $\mathrm{rec}(F, G)$ est une expression d'arité $m+1$.

Définition : On définit la fonction $[\![\ ]\!] : \mathcal{L}_{PR} \to \bigcup_{k \in \mathbb{N}} \mathcal{F}(\mathbb{N}^k, \mathbb{N})$ par induction structurelle où $\mathcal{F}(\mathbb{N}^k, \mathbb{N})$ représente les fonctions totales de $\mathbb{N}^k$ dans $\mathbb{N}$.
* $[\![\mathbb{0}]\!] : \mathbb{N}^0 \to \mathbb{N}$, $() \mapsto 0$
* $[\![\mathbb{S}]\!] : \mathbb{N} \to \mathbb{N}$, $x \mapsto x+1$
* $[\![\Pi_i^m]\!] : \mathbb{N}^m \to \mathbb{N}$, $(x_1, \dots, x_m) \mapsto x_i$
* Si F est une expression d'arité m et $(G_1, \dots, G_m)$ sont des expressions d'arité p alors $[\![\circ(F, G_1, \dots, G_m)]\!] : \mathbb{N}^p \to \mathbb{N}$, $\vec{x} \mapsto [\![F]\!]([\![G_1]\!](\vec{x}), \dots, [\![G_m]\!](\vec{x}))$
* Si F est une expression d'arité m et G est une expression d'arité $m+2$, alors $[\![\mathrm{rec}(F, G)]\!] = g$ où $g : \mathbb{N}^{m+1} \to \mathbb{N}$ est la fonction définie par récurrence par :
$$g(\vec{x}, k) = \begin{cases} [\![F]\!](\vec{x}) & \text{si } k = 0 \\ [\![G]\!](\vec{x}, k-1, g(\vec{x}, k-1)) & \text{si } k > 0 \end{cases}$$

Définition [3, p. 122] : Les fonctions primitives récursives sont les fonctions qui sont une expression de $\mathcal{L}_{PR}$ dénote.

Exemples [3, p. 122] :
* $\mathrm{plus} : \mathbb{N}^2 \to \mathbb{N}$, $(n_1, n_2) \mapsto n_1 + n_2$ est primitive récursive : $\mathrm{plus} = [\![\mathrm{rec}(\Pi_1^1, \circ(\mathbb{S}, \Pi_3^3))]\!]$
* $\mathrm{fact} : \mathbb{N} \to \mathbb{N}$, $n \mapsto n!$ est primitive récursive : $\mathrm{fact} = [\![\mathrm{rec}(\circ(\mathbb{S}, \mathbb{0}), \circ(*, \Pi_2^3, \circ(\mathbb{S}, \Pi_1^3)))]\!]$ où $*$ est le produit de deux nombres entiers (primitive récursive).
* $\mathrm{signe} : \mathbb{N} \to \{0, 1\}$, $n \mapsto \begin{cases} 0 \text{ si } n = 0 \\ 1 \text{ sinon} \end{cases}$ est primitive récursive : $\mathrm{signe} = [\![\mathrm{rec}(\mathbb{0}, \circ(\mathbb{0}, \mathbb{0}^3))]\!]$ où $\mathbb{0}^3 = \circ(\mathrm{rec}(\mathbb{0}, \Pi_2^2), \Pi_1^k)$.

* Flèche de Knuth : $a \uparrow^1 b = a^b$
$$\forall n \geq 1, \ a \uparrow^n b = \begin{cases} 1 & \text{si } b = 0 \\ a \uparrow^{n-1} (a \uparrow^n (b-1)) & \text{sinon} \end{cases}$$

B - Les prédicats. [3, p. 125]

Définition : Un prédicat d'arité k, avec $k \geq 0$, est une fonction de $\mathbb{N}^k$ dans $\{\text{vrai}, \text{faux}\}$.

Remarque : De façon équivalente, un prédicat est définit par un sous-ensemble de $\mathbb{N}^k$ correspondant aux éléments de $\mathbb{N}^k$ pour lequel il est vrai

Exemple : Le prédicat pair : $\mathbb{N} \to \{\text{vrai}, \text{faux}\}$, $n \mapsto \begin{cases} \text{vrai si } \exists k \text{ tel que } 2k = n \\ \text{faux sinon} \end{cases}$
Il correspond au sous-ensemble des naturels pairs.

Définition : La fonction caractéristique d'un prédicat $P \subseteq \mathbb{N}^k$ est la fonction $f : \mathbb{N}^k \to \{0, 1\}$ telle que $f(\vec{n}) = \begin{cases} 0 \text{ si } \vec{n} \notin P \\ 1 \text{ si } \vec{n} \in P \end{cases}$.

Définition : Un prédicat est primitif récursif si sa fonction caractéristique est primitive récursive.

Exemples :
* Le prédicat zéro n'est vrai que pour l'entier 0. Il est primitif récursif car zéro = 1 - signe.
* Le prédicat égal est primitif récursif car égal$(n, m) = 1 - (\mathrm{signe}(m - n) + \mathrm{signe}(n - m))$ où signe est définie sur $\mathbb{Z}$ et vaut 0 si $n \leq 0$.

Explication : Les fonctions à support fini sont primitives récursives par somme finie de produit de fonctions caractéristiques

Définition : La minimisation bornée d'un prédicat $q(\vec{n}, i)$ se dénote par $\mu i \leq m \, q(\vec{n}, i)$ est est définie par :
$$\mu i \leq m \, q(\vec{n}, i) = \begin{cases} k \text{ plus petit } i \leq m \text{ tel que } q(\vec{n}, i) = 1 \\ 0 \text{ si il ne existe pas de tel } i. \end{cases}$$

Proposition : La minimisation d'un prédicat primitif récursif est aussi primitive récursive.

C - Les limites des fonctions primitives récursives.

Théorème : [3, p. 129] Il existe des fonctions calculables qui ne sont primitives récursives.

Idée de la démonstration : argument diagonal.

Définition : [1, p. 184] On définit la fonction d'Ackermann, notée A, de $\mathbb{N}^2$ dans $\mathbb{N}$ par :
$$\begin{cases} A(0, m) = m+1 \\ A(k+1, 0) = A(k, 1) \\ A(k+1, m+1) = A(k, A(k+1, m)) \text{ pour } m \geq 0 \end{cases}$$

Théorème : [1, p. 184] La fonction d'Ackermann n'est pas primitive récursive.

Remarque : Les fonctions primitives récursives ne permettent pas de décrire les fonctions calculables.

Lien avec les langages de programmation : Les fonctions primitives récursives sont aussi expressives que les langages ne contenant que une boucle FOR.

II - Les fonctions μ-récursives.

Définition : [3, p. 131] Une minimisation non-bornée d'un prédicat $q(\vec{n}, i)$ se dénote par $\mu i.\, q(\vec{n}, i)$ et est définie par

$$\mu i.\, q(\vec{n}, i) = \begin{cases} \text{le plus petit } i \text{ tel que } q(\vec{n}, i) = 1 \\ 0 \text{ si un tel } i \text{ n'existe pas.} \end{cases}$$

Définition : Le langage $\mathcal{L}_{\mu R}$ des expressions des fonctions μ-récursives est défini par induction :

* Expressions de $\mathcal{L}_{PR}$
* Si F est d'arité $p+1$, alors $mu(F)$ est une expression d'arité p.

Définition : On définit la fonction $[\![\cdot]\!] : \mathcal{L}_{\mu R} \to \bigcup_{n \in \mathbb{N}} \mathcal{F}_{partielle}(\mathbb{N}^n, \mathbb{N})$ par induction structurelle où $\mathcal{F}_{partielle}(\mathbb{N}^n, \mathbb{N})$ est l'ensemble des fonctions partielles de $\mathbb{N}^n \to \mathbb{N}$.

* La sémantique des expressions de $\mathcal{L}_{PR}$ reste la même.
* Si F est d'arité $p+1$ alors $[\![mu(F)]\!] : \mathbb{N}^p \to \mathbb{N}$, $\vec{x} \mapsto \mu i.\, [\![F]\!](\vec{x}, i)$

où $\mu i.\, [\![F]\!](\vec{x}, i) = \begin{cases} \text{le plus petit } i \in \mathbb{N} \text{ tel que } [\![F]\!](\vec{x}, i) \neq 0 \text{ si un tel } i \text{ existe.} \\ \text{non définie sinon} \end{cases}$

Définition : Une fonction partielle $\mathbb{N}^k \to \mathbb{N}$ est μ-récursive partielle si une expression de $\mathcal{L}_{\mu R}$ la dénote.

Exemples : [3, p. 136]

* La fonction $f(n) = \begin{cases} n/2 \text{ si } n \text{ est pair} \\ \text{non définie sinon} \end{cases}$ est une fonction μ-récursive partielle.
* La fonction d'Ackermann est une fonction μ-récursive partielle.

Définition : Un prédicat $q(\vec{n}, i)$ est dit sûr si $\forall \vec{n}\ \exists i\, q(\vec{n}, i) = 1$. [3, p. 131]

Définition : [3, p. 131] Une fonction $\mathbb{N}^k \to \mathbb{N}$ est μ-récursive totale si c'est une fonction μ-récursive partielle dont toutes les minimisations non-bornées sont appliquées à des prédicats sûrs.

Exemple : La fonction d'Ackermann est une fonction μ-récursive totale.

Proposition : [3, p. 138] Soit f une fonction μ-récursive de $\mathbb{N}$ dans $\mathbb{N}$, qui admet une fonction inverse, f^{-1}, bien définie et totale. Alors f^{-1} est μ-récursive.

Exemple : L'inverse de la fonction d'Ackermann est une fonction μ-récursive totale.

Lien avec les langages de programmation : La minimisation non-bornée introduit l'expressivité de la boucle WHILE. Les fonctions μ-récursives sont aussi expressives que les langages contenant la boucle FOR et la boucle WHILE.

III - Lien avec la calculabilité.

A - Lien avec la thèse de Church.

Thèse de Church : Les fonctions calculables sont exactement les fonctions μ-récursives.

* Etude des machines de Turing.

Théorème [1, p. 185] Une fonction totale est μ-récursive totale si et seulement si elle est calculable par une machine de Turing. DEV ⇐

Corollaire : [3, p. 137] Une fonction partielle est μ-récursive partielle si seulement si elle est calculable par une machine de Turing.

Application : On peut reformuler la thèse de Church à l'aide de machine de Turing : les fonctions calculables sont exactement les fonctions calculables par une machine de Turing.

* Etude du λ-calcul [2, p. 185]

Définition : L'ensemble L des termes du λ-calcul est le plus petit ensemble tel que :

* les variables $x, y, z, \ldots$ sont des termes ;
* si u et v sont des termes, $(u\,v)$ est un terme ;
* si x est une variable et t est un terme, $\lambda x.t$ est un terme.

Le terme $(u\,v)$ représente l'application de la fonction u à l'argument v. Le terme $\lambda x t$ représente la fonction qui, à la variable x, associe le terme t : ce terme est obtenu par abstraction sur la variable x.

Exemple : Codage des entiers

* $0 = \lambda(fx).x$
* $1 = \lambda(fx).fx$
* $2 = \lambda(fx).f(fx)$
* $n = \lambda(fx).\underbrace{f(f(\ldots(fx)\ldots))}_{n \text{ fois}} = \lambda(fx).f^n x$ avec f itéré n fois

Ici f représente la fonction successeur.

Théorème : Les fonctions μ-récursives sont exactement celles qui sont représentables par un terme du λ-calcul

Corollaire : Les fonctions calculables par une machine de Turing

sont exactement celles qui sont représentables par un terme du λ-calcul.

B- Les langages récursivement énumérables

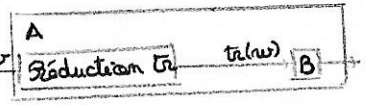
Définition : [3, p.140] La classe de décidabilité R est l'ensemble des langages décidables par une machine de Turing.
La classe de décidabilité RE est l'ensemble des langages acceptés par une machine de Turing.

Problème : Arrêt
entrée : une machine de Turing déterministe $\mathcal{M}$, un mot w.
sortie : oui si $\mathcal{M}(w)$ s'arrête ; non, sinon

Théorème : Le problème de l'arrêt est dans RE et est indécidable.

Définition : Une réduction d'un problème A à un problème B est une fonction tr calculable telle que pour toute instance w de A, w est une instance positive de A si et seulement si $tr(w)$ est une instance positive de B. On dit que A se réduit à B si une telle réduction existe.

Théorème : Si A se réduit à B, alors :
* B décidable implique A décidable.
* A indécidable implique B indécidable.



Problème : Pavage de Wang.
entrée : une famille finie de tuiles de la forme  , avec a, b, c, d appartenant à un ensemble fini.
sortie : oui si le jeu de tuiles permet de paver le plan ; non sinon.

Application : Le problème du pavage de Wang est indécidable.

Théorème [1, p.162] (Rice) : Pour toute propriété non triviale P sur les langages récursivement énumérables, le problème de savoir si le langage $L(M)$ d'une machine de Turing M vérifie P est indécidable.

Application : $P = \emptyset$: le problème du langage vide est indécidable.
Problème : Langage vide
entrée : une machine de Turing M
sortie : oui si $L(M) = \emptyset$; non sinon

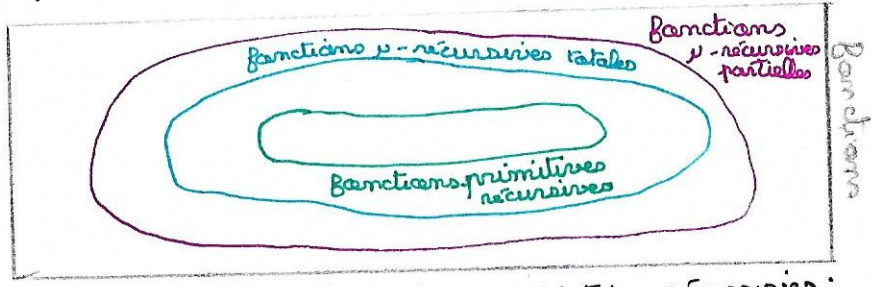
Définition : [4, p.41] Soit $A \subset \mathbb{N}^p$; on dit que A est récursif si sa fonction caractéristique est μ-récursive totale
On dit que A est récursivement énumérable si c'est le domaine de définition d'une fonction μ-récursive partielle.

Théorème [4, p.49] (Rice version fonctions récursives)
Soit $\mathcal{X}$ un ensemble de fonctions μ-récursives partielles à une variable, que l'on suppose non-vide, et distinct de l'ensemble de toutes les fonctions μ-récursives partielles. Alors, l'ensemble $A = \{x ; \lambda x\, \varphi(x) \in \mathcal{X}\}$ n'est pas récursif.

Théorème (Caractérisation des fonctions récursivement énumérables) Soit $A \subseteq \mathbb{N}$. Les assertions suivantes sont équivalentes :
1- $A \in RE$.
2- $A = \pi_2^2(B)$ avec $B \subseteq \mathbb{N}^2$ primitif récursif.
3- $A = \emptyset$ ou A est l'image d'une fonction primitive récursive.
4- A est l'image d'une fonction μ-récursive.

DEV

Ouverture :
* On a l'inclusion des classes de fonctions suivantes :
primitive récursive $\subseteq$ μ-récursive totale $\subseteq$ μ-récursive partielle



* On a une hiérarchie des fonctions primitives récursives : la hiérarchie de Grzegorczyk.

Références :

1- Cartan ; Langages formels, calculabilité et complexité.
2- Rougemont, Sassaigne ; Logique et fondement de l'informatique. Logique du 1er ordre, calculabilité et λ-calcul.
3- Wolper ; Introduction à la calculabilité.
4- Carl, Lascar ; Logique mathématique, tome 2