Les machines de Turing sont équivalentes que les fonctions μ-récursives

Julie Parreaux

2018-2019

Référence du développement : Wolper [2, p.135]

Leçons où on présente le développement : 913 (Machines de Turing) ; 912 (Fonctions récursives).
Leçons pour lesquelles on peut l'évoquer : 914 (Décidabilité et indécidabilité) si on évoque la thèse de Church.

1 Introduction

La thèse de Church qui dit que les fonctions calculables sont exactement celles calculables par une machine de Turing ne peut pas être prouvé. Cependant nous pouvons conforter ce résultat en comparant la puissance de calcul des machines de Turing à d'autres modèles de calculs qui permettent de définir des fonctions calculables (les fonctions récursives par exemple). Si les ensembles définit par ces deux modèles sont égaux cela nous conforte dans la thèse de Church.

L'objectif de ce développement est de montrer que les machines de Turing est un modèle de calcul équivalent aux fonctions μ-récursives. Pendant ce développement nous nous concentrons sur la preuve du sens direct du théorème 1 énoncé ci-dessus. Nous évoquerons rapidement le sens réciproque à la fin de ce document.

Théorème. *Les fonctions calculables par une machines de Turing sont exactement les fonctions μ-récursives.*

En particulier, dans la leçon sur les fonctions primitives récursives, ils est important de noter que nous prouvons le théorème dans le cadre de fonctions totales (le résultat analogue se fait trivialement). On joue alors sur la définition de calculabilité des machines de Turing.

Théorème. *Les fonctions totales calculables par une machines de Turing sont exactement les fonctions μ-récursives totales.*

Remarques sur le développement

Ce développement présente la construction d'une fonction μ-récursive à partir du fonctionnement d'une machine de Turing (étude de cas). Cette version n'est pas la plus formelle mais permet de faire apparaître la Gödelisation. Pour avoir une meilleure version, aller voir le Carton [1].

2 Les machines de Turing sont moins expressives que les fonctions μ-récursives.

Théorème. *Toute fonction (totale) calculable par une machine de Turing est aussi μ-récursive (totale).*

Démonstration. Soit M une machine de Turing sur un alphabet Σ calculant une fonction $f_{\mathcal{M}} : \Sigma^* \to \Sigma^*$. Quitte à déterminiser $\mathcal{M}$, on peut supposer que $\mathcal{M}$ est déterministe. On pose une fonction $gd : \Sigma^* \to \mathbb{N}$ codant les mots de Σ^* par des entiers. Nous allons montrer qu'il existe f une fonction μ-récursive tel que $f_{\mathcal{M}}(w) = gd^{-1}(f(gd(w)))$.

Hypothèses sur la machine $\mathcal{M}$ On peut supposer sans perte de généralité qu'il existe $m \in \mathbb{N}^*$ tel que $\mathcal{Q} = \{0; \ldots; m-1\}$ où $\mathcal{Q}$ est l'ensemble des états de $\mathcal{M}$. De plus, on suppose que $\mathcal{M}$ possède un unique état initial $q_{init} = 0$ et un unique état final $q_{final} = m-1$ qui est atteint lorsque $\mathcal{M}$ inscrit le dernier caractère de la solution (celui-ci est alors avant la tête de lecture sur le ruban). De plus, on suppose $\mathcal{M}$ complète de telle sorte que la fonction de transition $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q} \times \Gamma \times \{\leftarrow; \rightarrow\}$ soit total.

Coder des configurations Pour traduire une exécution d'une machine de Turing déterministe à l'aide de fonctions μ-récursives, il nous faut les définir sur des configurations. Celles-ci interpréterons la suite de configurations représentant cette exécution. Nous devons alors coder les configurations : on code les mots sur les rubans par la méthode de Godël et $\{\leftarrow; \rightarrow\}$ par deux entiers distincts. On peut codé les configurations de la machine de Turing à l'aide de la fonction $\mathsf{code\_conf} : \Sigma^* \times \mathbb{N} \times \Sigma^* \to \mathbb{N}^3$ définie telle que $(\alpha, n, \gamma) \mapsto (gd(\alpha), n, gd(\gamma))$. Comme gd est une fonction primitive récursive (elle est une somme finie de multiplications et de puissances), code_conf est également primitive récursive.

Définitions des fonctions primitives récursives utilisées lors du fonctionnement d'une machine de Turing déterministe Le comportement d'une machine de Turing déterministe peut être modélisé par une composition de fonctions primitives récursives sous une minimisation non bornée. Ces fonctions opèrent sur le codage des configurations à l'aide de la fonction code_conf qui code les configurations de $\mathcal{M}$.

INIT(x) donne la configuration initiale de $\mathcal{M}$ pour le mot d'entrée w mis sous forme de Gödel v.

$$\begin{array}{rccc} \text{INIT} : & \mathbb{N} & \to & \mathbb{N}^3 \\ & v & \mapsto & \mathsf{code\_conf}(\epsilon, 0, gd^{-1}(v)) \end{array}$$

La fonction INIT est bien primitive récursive car $\text{INIT} = \circ(\mathsf{code\_conf}, \epsilon, \mathbf{O}, \circ(gd^{-1}, \pi_1^1))$.

CONFIG_SUIVANTE(x) donne la configuration suivante d'une configuration $c = (\alpha a, c_2, b\beta)$ sous forme de Gödel que l'on note $x = (x_1, x_2, x_3)$ par $\mathcal{M}$. On définit la fonction $\text{CONFIG\_SUIVANTE} : \mathbb{N}^3 \to \mathbb{N}^3$ par :

$$\text{CONFIG\_SUIVANTE}(x_1, x_2, x_3) \mapsto \begin{cases} \mathsf{code\_conf}(\alpha a b', q', \gamma) & si\ \delta(c_2, b) = (q', b', \rightarrow) \\ \mathsf{code\_conf}(\alpha, q', ab'\gamma) & si\ \delta(c_2, b) = (q', b', \leftarrow) \end{cases}$$

La fonction CONFIG_SUIVANTE est une fonction primitive récursive. La fonction de transitions, δ est primitive récursive car elle est à support fini. En effet, une fonction à support fini peut toujours être écrite comme somme finie de fonctions indicatrices. Ces dernières étant, elles-même, primitives récursives par le prédicat ÉGALITÉ. On en conclut, que les fonctions à support fini peuvent être réécrite à l'aide d'une conditionnelle et de fonctions primitives récursives.

Soient (x_1, x_2, x_3) la configuration encodée de la configuration $c = (c_1, c_2, c_3)$. On note $\delta(c_2, b) = (q', b', \delta_3)$ où $\delta_3 \in \{\leftarrow; \rightarrow\}$. On récupère ces valeurs à l'aide d'une projection sur le résultat de cette fonction (ceci est primitif récursif). On a alors

$$\text{CONFIG\_SUIVANTE}(x_1, x_2, x_3) = \begin{cases} (\circ(\mathsf{code\_conf}, \alpha a b', q', \gamma)) & si\ \delta_3 = \rightarrow \\ (\circ(\mathsf{code\_conf}, \alpha, q', ab'\gamma)) & si\ \delta_3 = \leftarrow \end{cases}$$

qui est alors primitive récursive.

CONFIG(x, n) donne la configuration après n étapes de calculs d'une configuration c sous forme de Gödel que l'on note x dans $\mathcal{M}$. On définit $\text{CONFIG} : \mathbb{N}^3 \times \mathbb{N} \to \mathbb{N}^3$ tel que :

$$\text{CONFIG}(x, n) \mapsto \begin{cases} x & si\ n = 0 \\ \text{CONFIG\_SUIVANTE}(\text{CONFIG}(x, n-1)) & sinon \end{cases}$$

qui est primitive récursive car $\text{CONFIG} = rec(\pi_1^3, \circ(\text{CONFIG\_SUIVANTE}, \pi_3^3))$;

STOP(x) est un prédicat qui teste si la configuration donnée $c = (c_1, c_2, c_3)$ dont la représentation de Gödel est $x = (x_1, x_2, x_3)$ est finale.

$$\begin{array}{rccc} \text{STOP} : & \mathbb{N}^3 & \to & \{0; 1\} \\ & (x_1, x_2, x_3) & \mapsto & x_2 = m - 1 \end{array}$$

La fonction STOP est bien récursive primitive car $\text{STOP} = \circ(egal, \pi_2^3, m-1)$. (Dans le cadre de la leçon 912 : notons que le prédicat est primitif récursif car la machine de Turing calcul la fonction f et donc elle s'arrête nécessairement car f est totale. Dans le cadre d'une fonction partielle le prédicat n'est plus sûr et on tombe sur une fonction primitive récursive partielle.)

SORTIE(x) donne la valeur du ruban de $\mathcal{M}$ lorsqu'elle est dans une configuration finale $c_{\text{finale}} = (c_1, c_2, c_3)$ dont la représentation de Gödel est $x = (x_1, x_2, x_3)$.

$$\begin{array}{rccc} \text{SORTIE} : & \mathbb{N}^3 & \to & \mathbb{N} \\ & (x_1, x_2, x_3) & \mapsto & x_1 \end{array}$$

La fonction SORTIE est bien récursive primitive car $\text{SORTIE} = \pi_1^3$.

Représentation de $\mathcal{M}$ et conclusion La fonction f recherché est alors $f(x) = \text{SORTIE}(\text{CONFIG}(\text{INIT}(x), \text{NB\_PAS}(x)))$ où $\text{NB\_PAS}(x) = \mu i.\text{STOP}(\text{CONFIG}(\text{INIT}(x), i))$. Comme NB_PAS est une minimisation non bornée d'une fonction primitive récursive $f(x,i) = \circ(\text{STOP}, \circ(\text{CONFIG}, \circ(\text{INIT}, \pi_1^2), \pi_2^2))$, elle est $\mu - recursive$. Donc f est μ-récursive par composition de fonctions primitives récursives et μ-récursive. □

3 Les machines de Turing sont plus expressives que les fonctions μ-récursives

Théorème. *Toute fonction μ-récursive est aussi calculable par une machine de Turing.*

Idée de la preuve. Se fait par induction sur la structure des expressions primitives récursives.

Fonction nulle : Machine qui remplace les arguments par 0.

Fonction sucesseur Machine qui remplace incrémente d'un le dernier nombre d'écrit.

Projection Recopier le $i$ème argument en première place et effacer le reste.

Composition Remplacer sur la deuxième machine ses arguments par le résultat de ceux-ci par la première.

Récursivité Boucle pour.

Minimisation non bornée Boucle tant que.

□

4 Gödelisation

Pour représenter des chaînes de caractères par des entiers ou vise-versa, on peut raisonner comme suit : les chaînes de caractères est un ensemble dénombrable donc il existe une bijection entre les entiers et les chaînes de caractères. On peut alors coder les entiers (ou les chaînes de caractères) à l'aide de cette bijection (ou de sa réciproque). Attention : ce raisonnement est correct mais pas suffisant. En effet, dans le contexte de la calculabilité, il faut de plus que cette bijection (et sa réciproque) soit calculable (par une procédure effective). Nous nommerons cette représentation, le représentation effective des entiers ou des chaînes de caractères.

Lemme. *Il existe une représentation effective des chaînes de caractères par les entiers.*

Démonstration. Les représentations binaire (sur l'alphabet $\Sigma = \{0,1\}$) ou décimale sont des exemples de représentations effectives. Nous allons montrer que la représentation binaire est bien une représentation effective en montrant qu'il existe des fonctions μ-récursives (qui sont en réalité des fonctions primitives récursives) qui permettent de calculer la représentation binaire.

BINLONG(n) Calcul la longueur binaire de n (primitive récursive car c'est une minimisation bornée (par n) de division par deux qui donne le résultat) ;

CHIFFRE(n, m) Donne la $m$ième chiffre de l'entier n si $m \leq \text{BINLONG}(n)$ et 0 sinon (primitive récursive car c'est une minimisation bornée (par m) de division par deux qui donne le résultat).

Ces deux fonctions sont bien primitives récursives. □

Remarque : notons que cette représentation bien que calculable par une machine de Turing n'est pas nécessairement la plus agréable à manipuler. Une représentation unaire, vue comme une composition de fonction successeur, peut être beaucoup plus intéressante pour écrire sur le ruban de notre machine de Turing.

La représentation inverse est la plus délicate, c'est elle qui porte le nom de Gödélisation car introduite par le logicien Kurt Gödel.

Lemme. *Il existe une représentation effective des entiers par les chaînes de caractères.*

Démonstration. Soit Σ un alphabet contenant k symboles. Une représentation simple consiste à attribué à chacune des lettres de l'alphabet un nombre entre 0 et $k-1$ (si $a \in \Sigma$, on note $gd(a)$ son codage par un entier). Puis on code la chaîne de caractère à partir de ce codage : soit $w = w_1 \dots w_n$ une chaîne de caractères, alors $gd(w) = \sum_{i=0}^{l} k^(l-i) gd(w_i)$ qui correspond au naturel dont la représentation en base k est $gd(w_1) \dots gd(w_n)$.

Attention : ce codage n'est pas univoque puisque $gd(aa) = 00 = 0 = gd(a)$. Pour corriger cette représentation, il faut que le codage ne doit pas utilisé 0. Pour cela on code les lettres de l'alphabet entre 1 et k et on a $gd(w) = \sum_{i=0}^{l} (k+1)^(l-i) gd(w_i)$ □

Remarque : Il est bon de remarqué que la représentation donnée par gd n'est pas injective. En effet, par définition 0 ne sera jamais atteint. Cependant, comme nous cherchons une représentation des chaînes de caractères et non des entiers, cette non injectivité n'est pas gênante.

Références

[1] O. Carton. *Langages formels, calculabilité et complexité*. Vuibert, 2008.

[2] P. Wolper. *Introduction à la calculabilité*. Dunod, 2006.