Chapitre 2

Chapitre 2 : Une brève introduction à la calculabilité avec la machine de Turing et les fonctions primitives récursives

2.1 Introduction

Un programme peut être considéré comme la décomposition de la tâche à réaliser en une séquence d'instructions élémentaires (manipulant des données élémentaires) compréhensibles par l'automate programmable que l'on désire utiliser. Cependant, chaque automate programmable (en pratique chaque processeur) se caractérise par un jeu d'instructions élémentaires qui lui est propre. Dans la pratique, cette diversité est résolue par l'existence de compilateurs ou d'interpréteurs qui traduisent les instructions du langage (évolué) mis à disposition des programmeurs en plusieurs instructions élémentaires (langage machine) du processeur utilisé. Cependant, cette solution n'est pas suffisante pour les besoins de l'informatique théorique, qui requière une représentation unifiée de la notion d'automate programmable, permettant de démontrer des résultats généraux (décidabilité, complexité, ...) vrais pour l'ensemble des automates programmables concrets que l'on peut envisager. A cette fin a été développée la notion de machine de Turing, qui constitue une abstraction (et une formalisation) de la notion d'automate programmable.

la machine de Turing est un modèle de machine théorique fondamental pour la théorie de la calculabilité et de la complexité. Malgré sa simplicité extrême, on peut démontrer qu'elle est capable de simuler toute opération

réalisable par n'importe quel processeur, aussi puissant soit-il. Elle résume de manière saisissante le concept d'ordinateur et constitue un support idéal pour raisonner autour de la notion d'algorithmique.

Il existe de nombreuses formulation de cette machine, et si celle qui va suivre ne correspond pas exactement à celle que vous connaissez, c'est sans grande importance, car il est très facile de montrer qu'elles sont équivalentes.

2.2 La définition formelle d'une machine de Turing (MT)

Une machine de Turing est présentée comme un quintuplet $\langle Q, A, q_0, b, d \rangle$ où :

- Q est un ensemble fini d'états,
- A est un ensemble fini d'alphabet,
- $q_0 \in Q$ représente l'état initial,
- b est un symbole n'appartenant pas à A appelé symbole blanc,
- d est une fonction de transition de Q x B $\longrightarrow$ (B $\cup\{<, >\}$) x Q, avec B = A $\cup\{b\}$.

Nous allons donner une description plus physique du fonctionnement d'une machine de Turing. Pour reprendre l'idée même de Alan Turing (le créateur de la machine qui porte aujourd'hui son nom). Une machine de Turing n'est rien d'autre qu'une machine à écrire modifiée. C'est-à-dire au lieu de travailler sur une feuille de papier, la machine de Turing opère sur une bande infinie à gauche et à droite (indexée par l'ensemble **Z** des entiers relatifs) à l'aide d'une tête de Lecture/ Écriture. La bande est constituée de plusieurs cellules, chaque cellule contenant un symbole de l'alphabet A ou un symbole blanc. On dit qu'une cellule est vierge si le symbole qu'elle contient est le symbole blanc. La tête de Lecture/Écriture est capable :

- D'écrire un symbole dans la cellule courante (celle sur laquelle la tête se pointe),
- D'effacer le contenu de la cellule courante (équivaut à écrire le symbole blanc),
- De se déplacer d'une cellule vers la gauche ou vers la droite ($< et >$ dans la définition),
- De lire le symbole contenu dans la cellule courante.

Exemple. La machine vue dans la figure suivante est une machine qui lit un « b » dans l'état p, passe à l'état q, écrit un « a » à la place du « b » et déplace sa tête de lecture à gauche.

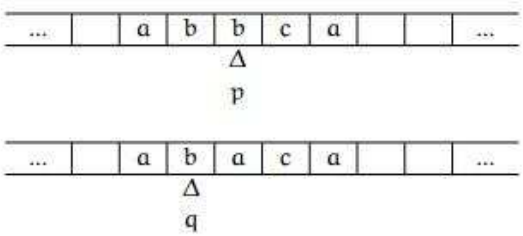

FIGURE 2.1 – Exemple d'une machine de Turing

Du point de vue du modèle, les trois premières opération sont représentées par l'ensemble B $\cup\{<,>\}$, c'est-à-dire on écrit un symbole de B (si ce symbole est b, on considère qu'on efface la cellule), ou on déplace la tête de Lecture/ Écriture d'une cellule vers la gauche ou la droite. Notons que dans la majorité des ouvrages, ces opérations sont représentées par l'ensemble B x $\{<,>\}$,i.e. on écrit un symbole de B et on déplace la tête de Lecture/ Écriture vers la gauche ou la droite.

La fonction de transition d et le contenu initial de la bande qui vont déterminer la séquence des opérations exécutées par la machine ; Au début de l'exécution, la machine est dans l'état initial q_0 et la tête de lecture/ Écriture est par convention placée sur la cellule indexée par 0 ; L'égalité d(q, s) = (o, q′) se lit : si la machine de Turing est dans l'état q et que la cellule contient le symbole s (on dit que la tête de Lecture/ Écriture lit le symbole s), alors la machine de Turing réalise l'opération o et passe à l'état q′.

On appelle instruction d'une machine de Turing , tout quadruplet (q, s, o, q′) tel que d(q, s) = (o, q′). Se donner un ensemble d'instructions, donc de quadruplets de Q x B x (B $\cup\{<,>\}$) x Q revient à définir le graphe d'une fonction de transition, les deux approches sont donc équivalentes. L'ensemble des instructions est appelé *programme* de la machine de Turing. Quand la machine est dans un état q, qu'elle lit un symbole s et que la fonction d n'est

pas définie pour le couple (q, s), on dit que la machine s'arrête et que l'état q est un état d'arrêt.

Exemple applicatif. Écrire la machine de Turing qui reconnait les mots de longueurs paires sachant que l'alphabet = $\{a, \#\}$, si la longueur du mot est paire la machine écrit T sur la bande sinon elle écrit F.

Solution
q_0 a D q_1
q_1 a D q_2
q_2 # T q_3
q_3 T arrêt
q_2 a D q_4
q_4 a D q_2
q_4 # F q_4
q_1 # F q_4
q_4 F arrêt

2.3 Les fonctions primitives récursives

Les fonctions primitives récursives ont été introduites par Godel en 1934 dans son travail sur l'incomplétude. Elles permettent de décrire des fonctions dont il est clair que le calcul termine toujours. Elles correspondent ainsi aux fonctions qui peuvent être calculées sans l'instruction "While" dans un langage typé comme Pascal, c'est à dire avec des " if-then-else" et des "for". L'objectif de cette petite classe est de se convaincre de cette expressivité et d'observer une partie de ses limitations.

Informellement, ces fonctions calculables sont celles que l'on peut définir sur l'ensemble des entiers naturels $\mathbb{N}$ par récurrence :
$\bigcup_{k\in\mathbb{N}} F_k$ avec $F_k = \mathbb{N}^k \longrightarrow \mathbb{N}$

Elles sont comme suit :
- Les fonctions de base,
- La composition de fonctions,
- le mécanisme de récursion.

2.3.1 Définition des fonctions primitives récursives de base

- La fonction zéro() ou $\boldsymbol{O}() \in F_0$ tel que $\boldsymbol{O}() = 0$,
- La fonction successeur succ(x) ou $\sigma(n) \in F_1$ tel que $\sigma(n) = n+1$,
- La fonction de projection $proj_i^k$ ou $\pi_i^k \in F_k$, $k \geqslant 1, 1 \leqslant i \leqslant k$ tel que $\pi_i^k(n_1, ..., n_k) = n_i$.
  Exemple : $\pi_2^3(5, 6, 8) \longrightarrow 6$.

2.3.2 Définition de la composition de fonctions

Soient $g \in F_l$ et $h_1, ..., h_l \in F_k$. La composition de g et des fonctions $h_1, ..., h_l$ est définie par $f \in F_k$ tel que :
f(n) = $g(h_1(n), ..., h_l(n))$ avec n = $(n_1, ..., n_k)$
Exemple : dans cet exemple, on représente la composition par $\circ$, on peut aussi la représenter par Comp.
succ $\circ$ Zéro $\longrightarrow$ 1,
succ $\circ$ succ $\circ$ Zéro $\longrightarrow$ 2,
succ $\circ$ π_3^3(0, 2, 5)$\longrightarrow$ 6.

2.3.3 Définition de la récursion primitive

Soient $g \in F_k$ et $h \in F_{k+2}$, la fonction $f \in F_{k+1}$ tel que :
- f(n, 0) = g(n),
- f(n, m+1)= h(n,m,f(n,m))

Est la fonction définie à partir de g et h par récursion primitive.
Remarque : La fonction f est calculable si g et h sont calculables.

On peut aussi représenter la récursion par le schéma récursif suivant :

Fonction f(x, k)
Si : k=0
F(x)
Sinon :
G(x, k-1, f(x, k-1))
Fin
Fin

On a remplacé les fonctions g par F et h par G pour dire qu'il y a plusieurs représentations formelles. Si F et G sont primitives récursives alors f

est primitives récursives, f fait la récursivité sur l'argument k, tant dis que l'argument x est une donnée supplémentaire.

2.3.4 Définition des fonctions primitives récursives

D'une manière générale, les fonctions primitives récursives sont toutes les fonctions que l'on peut construire à partir des fonctions de base par composition et récursion primitive.

Exemple : Montrez que la fonction +(x, k) est primitive récursive.

En se basant sur le schéma récursif vu précédemment, on peut avoir :

Fonction +(x, k)
Si : k=0
F(x)
Sinon :
G(x, k-1, f(x, k-1))
Fin
Fin

Donc pour F on prend la fonction identité $\pi_1^1(x)$ et pour la fonction G on prend succ $\circ\pi_3^3$ c'est à dire :

Fonction +(x, k)
Si : k=0
$\pi_1^1(x)$
Sinon :
succ $\circ\pi_3^3$(x, k-1, f(x, k-1))
Fin
Fin

Exemple applicatif : +(5,3)
+(5,3) = succ $\circ\pi_3^3$(5,2,+(5,2)) =
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,+(5,1)) =
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,succ $\circ\pi_3^3$(5,0,+(5,0))=
succ $\circ\pi_3^3$(5,2,succ $\circ\pi_3^3$(5,1,succ $\circ\pi_3^3$(5,0,π_1^1(5))))
Comme : $\pi_1^1(5)$=5 alors :
succ $\circ\pi_3^3$(5,0,+(5,0)) = 6 et
succ $\circ\pi_3^3$(5,1,+(5,1)) = 7 et enfin

succ $\circ\pi_3^3$(5,2,+(5,2)) = 8

On remarque que nous avons réussi à construire la fonction + par la règle de récursion à partir des deux fonctions primitives récursives :

- La fonction F = π_1^1 qui est une fonction de base (la projection)
- La fonction G = $succ \circ \pi_3^3$ qui est une composition de deux fonctions de base la projection et la fonction successeur.

Remarque 1 : Dans la récursion, l'ordre des arguments de la fonction G(x, k-1, f(x, k-1)) n'est pas important, on peut trouver dans d'autres documents G(x,f(x, k-1), k-1).

Remarque 2 : On peut aussi résoudre le problème en utilisant la règle de récursion comme suit :
+(x,0) = x + 0 = x = π_1^1
+(x, y+1) = +(x,y)+1 = succ(+(x,y)) = $succ(\pi_2^3(x, +(x, y), y)) = succ \circ \pi_2^3(x, +(x, y), y)$.

Remarque 3 : L'objectif de la partie "Fonction primitives récursives" est d'abord de voir un autre modèle de calcul que la machine de Turing avec ses différentes formes et de montrer que les fonctions primitives récursives et la machine de Turing ont le même pouvoir expressif avec bien sur le λ-calcul (qui n'est pas dans le programme).

2.4 Remarques pour les TD

Afin d'avoir une représentation de la MT beaucoup plus algorithmique, nous écrirons les instructions de la manière suivante :

« état courant » « symbole » « opération » « état suivant ».

« opération » peut être un déplacement vers la droite « D » ou vers la gauche « G » ou bien le remplacement du symbole lu par un autre. Exemple : q_0 s D q_1 qui représente un déplacement vers la droite après la lecture du symbole s, q_0 s s$'$ q_1 dans ce cas il n' y a pas un déplacement par contre il y a un remplacement du symbole s par le symbole s$'$.

Au départ, la tête de L E est toujours placée sur un symbole différent du symbole blanc et on ne boucle pas sur l'état initial.

2.5 Exercices

Exercice 1 soient les instructions suivantes d'une MT quelconque :

q_0 s_0 D q_1
q_0 s_1 s_2 q_2
q_1 s_0 D q_1
q_1 s_2 D q_1
q_1 s_1 D q_2
q_1#
q_2 s_2 G q_3
q_3 (s_0/s_1/s_2) G q_3
q_3#

Dérouler cette MT sur la séquence suivante : #$s_0s_0s_2s_1s_2s_1s_2s_0s_2$# sachant que le # est le symbole blanc.

Exercice 2 Écrire la MT qui étant donné un mot sur le ruban composé des symboles a et b, détermine si le mot se termine par un b ou non. Elle écrit à la fin du mot un T si vrai et un F sinon. Le blanc = $.

Exercice 3 Modifier la MT précédente pour qu'elle vérifie si le mot en entrée se termine par le même symbole de départ (qu'il soit a ou b).

Exercice 4 Les valeurs en base 10 correspondants aux codes ASCII des lettres sont :

- A : 65 ; B : 66 ; C : 67 ; etc,
- a : 97 ; b : 98 ; c : 99 ; etc.

Pour passer du code ASCII d'une lettre minuscule à celui de la majuscule correspondante, il suffit de transformer le 3 ème bit en partant de la gauche de 1 à 0. Par ailleurs, les deux premiers bits sont toujours égaux à « 01 » et les 5 derniers bits ne sont pas modifiés.

A : 65 = 01000001 et a : 97 = 01100001
C : 67 = 01000011 et c : 99 = 01100011
Écrire la MT qui transforme une lettre minuscule en lettre majuscule.

Exercice 5 Écrire la MT qui reconnait la séquence 0001 dans un mot sachant que le ruban contient plusieurs mots et l'alphabet $\Sigma = \{0, 1, \#\}$. Il y a plusieurs mots sur le ruban séparés par un seul #. Deux # successifs désignent la fin de la séquence.

Exercice 6 Écrire la MT qui vérifie si un mot donné sur le ruban contient la séquence de caractères suivants : « aab ». Le blanc = # et il y a plusieurs mots sur le ruban séparés par un seul #. Deux # successifs désignent la fin. A = $\{a, b, \#\}$. On écrit un T ou un N.

Exercice 7 Écrire la MT qui remplace le "0" qui vient après deux "1" par un "1". A = $\{0, 1, \#\}$, q_0 est l'état initial et le ruban contient une seule séquence.

Exercice 8 Écrire la MT qui transforme le mot sur le ruban écrit sur $\{a, b, \#\}$ de telle sorte que tous les "a" soient au début. Exemple : aabbaba devient aaaabbb.

Exercice 9 Montrez que les fonctions suivantes sont primitives récursives :

1. La fonction plus = x + y,
2. La fonction Sigma = $\Sigma_{i=0}^{x} i$,
3. La fonction prédécesseur (pred(x)),
4. La fonction différence (diff(x,y)) telle que diff(x,y) = $\begin{cases} x-y \; si \; x>y \\ 0 \; si \; x \leq y \end{cases}$,
5. La fonction différence absolue $|x - y| = \begin{cases} x-y \; si \; x \geq y \\ y-x \; si \; x<y \end{cases}$,
6. La fonction alpha tel que $\alpha(x) = \begin{cases} 1 \; si \; x=0 \\ 0 \; si \; x \neq 0 \end{cases}$,
7. La fonction multiplication mult = x * y,
8. La fonction factorielle Fact(x) = $x!$.