Théorie de la complexité – Feuille de cours intégré n° 3
Machines de Turing

1 Machines de Turing

Tout comme un automate ou une grammaire, on utilise une machine de Turing pour spécifier un langage. On commence par présenter la *syntaxe* des machines de Turing.

Définition 1 (Syntaxe des machines de Turing)

Une machine de Turing est un tuple $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ où :

- Σ est l'alphabet d'entrée, il contient les symboles avec lesquels on écrit l'entrée qu'on donne à la machine.
- Γ est l'alphabet de travail, il étend Σ ($\Sigma \subseteq \Gamma$) avec des symboles supplémentaires qu'on peut utiliser pour le calcul. De plus, Γ contient deux symboles spéciaux $\square$ et \$ qui n'appartiennent pas à Σ.
- Q est un ensemble fini d'états.
- $q_i \in Q$ est l'état initial (celui dans lequel le calcul commence).
- $q_a \in Q$ et $q_r \in Q$ sont les états finaux (dans lesquels le calcul s'arrête) dits "état acceptant" et "état rejetant".
- δ est l'ensemble des transitions. C'est lui qui constitue le programme. On a :

$$\delta \subseteq (Q \setminus \{q_a, q_r\}) \times \Gamma \times Q \times \Gamma \times \{\triangleleft, \triangleright, \nabla\}.$$

De plus, on demande que pour tout $(s, a, t, b, D) \in \delta$, si $a = \$$, alors $b = \$$ et $D \in \{\triangleright, \nabla\}$.

On passe maintenant à la définition de la *sémantique* d'une machine de Turing, qui définit quel est le langage spécifié par une machine donnée.

Définition 2 (Sémantique des machines de Turing)

On considère une machine de Turing $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_f, \delta)$. Pour tout mot $w \in \Sigma^*$, on peut associer des exécutions de $\mathscr{M}$ sur w. Celles-ci peuvent être *acceptantes* ou *rejetantes* (il est aussi possible qu'il n'existe pas d'exécution de $\mathscr{M}$ sur w). Le langage défini par $\mathscr{M}$ est celui de tous les mots pour lesquels *il existe* une exécution acceptante.

Configurations. On commence par une notion intermédiaire : les *configurations* de notre machine $\mathscr{M}$. Une configuration de $\mathscr{M}$ est donnée par les trois éléments suivants :

- Un état de contrôle $q \in Q$.
- Une bande mémoire (infinie) B qu'on représente de la façon suivante :

Bande mémoire

\$	0	1	1	1	1	1	0	1	0	0	1	□	□	□	□	□	□	□	□	.....

Chaque case mémoire contient une lettre de Γ. La première case (celle qui est la plus à gauche) contient nécessairement la lettre "\$" (son rôle est de marquer cette case).

- Un entier $n \geqslant 0$ appelé pointeur qui désigne une case sur la bande mémoire (0 désigne la case la plus à gauche, contenant le symbole \$, 1 désigne la case suivante, etc.).

Une configuration est *finale* si l'état de contrôle q est un des deux états finaux de $\mathscr{M}$ (*i.e.*, $q = q_a$ ou $q = q_r$). Une configuration finale peut être *acceptante* (si $q = q_a$) ou *rejetante* (si $q = q_r$).

Configurations suivantes. Considérons deux configurations $C_1 = (q_1, B_1, n_1)$ et $C_2 = (q_2, B_2, n_2)$ de $\mathscr{M}$. On dit que C_2 est une *configuration qui suit* C_1 si il existe une transition $(q_1, a, q_2, b, D) \in \delta$ telle que,

1. les bandes B_1 et B_2 sont identiques sauf éventuellement sur la case n_1 qui contient la lettre "a" dans B_1 et la lettre "b" dans B_2, et,
2. l'une des trois conditions suivantes est satisfaite :
   - $n_2 = n_1 - 1$ et $D = \triangleleft$, ou,
   - $n_2 = n_1$ et $D = \triangledown$, ou,
   - $n_2 = n_1 + 1$ et $D = \triangleright$.

On notera $C_1 \to C_2$ pour indiquer que C_2 est une configuration qui suit C_1. De plus, on note $\xrightarrow{*}$ pour la clôture réflexive transitive de $\to$: étant données deux configurations C et C', on a $C \xrightarrow{*} C'$ si il existe $C_1, \ldots, C_n$ telles que,

$$C = C_1 \to C_2 \to \cdots \to C_n = C'.$$

On appelle la séquence de configurations $C_1, \ldots, C_n$ une *exécution partielle* de $\mathscr{M}$ depuis la configuration $C = C_1$.

Configurations initiales. Enfin, pour tout mot *d'entrée* $w \in \Sigma^*$, on associe une *configuration initiale* de $\mathscr{M}$ qu'on notera $C[w]$. Soient $a_1, \ldots, a_n \in \Sigma$ les lettres telles que $w = a_1 \cdots a_n \in \Sigma^*$. On définit $C[w] = (q_i, B, 1)$ où q_i est l'état initial de $\mathscr{M}$ et B est la bande mémoire contenant le mot w suivit de symboles "□", représentée ainsi :

$C_0 =$ 

Langage accepté par $\mathscr{M}$. Soit $w \in \Sigma^*$ un mot d'entrée pour $\mathscr{M}$.

- Une exécution (finie) de $\mathscr{M}$ sur w est donnée par une configuration *finale* C de $\mathscr{M}$ telle que $C[w] \xrightarrow{*} C$. Celle-ci est acceptante (resp. rejetante) si C est une configuration acceptante (resp. rejetante).
- Une exécution *infinie* de $\mathscr{M}$ sur w est une séquence infinie de configurations $C_1, \ldots, C_n, \ldots$ telle que $C[w] = C_1 \to \cdots \to C_n \to \cdots$.

On dit que w est accepté par $\mathscr{M}$ is *il existe une exécution (finie) acceptante de $\mathscr{M}$ sur w*. Autrement, on dit que w est rejeté par $\mathscr{M}$. Le langage accepté par $\mathscr{M}$, noté $L(\mathscr{M}) \subseteq \Sigma^*$ contient tous les mots qui sont acceptés par $\mathscr{M}$.

Exercice 1 Soit $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ une machine de Turing. Étant donné un mot $w \in \Sigma^*$, énoncer de manière positive ce que veut dire la phrase "w est rejeté par $\mathscr{M}$". ▪

Attention

Étant donnés une machine de Turing $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ et un mot d'entrée $w \in \Sigma^*$, il peut exister plusieurs exécutions de $\mathscr{M}$ sur w. À l'inverse, il se peut qu'il n'existe pas d'exécution de $\mathscr{M}$ sur w.

Définition 3 (Machines totales)

Une machine de Turing $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ est *totale* si pour tout mot $w \in \Sigma^*$, il n'existe pas d'exécution infinie de $\mathscr{M}$ sur w.

On peut maintenant définir les classes de langages associées aux machines de Turing.

Définition 4 (Langages semi-décidables et décidables)

Soit Σ un alphabet et $H \subseteq A^*$ un langage. On dit que,

- H est *semi-décidable* si il existe une machine de Turing $\mathscr{M}$ telle que $H = L(\mathscr{M})$.
- H est *décidable* si il existe une machine de Turing *totale* $\mathscr{M}$ telle que $H = L(\mathscr{M})$.

Exercice 2 On utilise l'alphabet $\Sigma = \{0,1\}$ pour coder des entiers en binaire. Montrer que le langage des mots codant un entier naturel pair est décidable. ▪

Exercice 3 On utilise l'alphabet $\Sigma = \{0,1\}$. Montrer que le langage des palindromes est décidable. ▪

Exercice 4 On utilise l'alphabet $\Sigma = \{a,b,c\}$. Montrer que le langage des mots qui contiennent le même nombre de a, de b et de c est décidable. Montrer ensuite que le langage des mots qui ont la forme $a^n\ b^n\ c^n$ (où $n \in \mathbb{N}$ est un entier naturel quelconque) est également décidable. ▪

Exercice 5 Soit $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$. On considère le langage $L \subseteq \Sigma^*$ de l'ensemble des suites de 42 chiffres qui sont des facteurs de longueur 42 du développement décimal du nombre π. Ce langage est-il décidable ? ▪

Exercice 6 Montrer que tout langage régulier est décidable. ▪

Exercice 7 Montrer que tout langage hors-contexte est décidable. ▪

2 Machines déterministes

Définition 5

Une machine de Turing $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ est dite *déterministe* si pour tout $q \in Q \setminus \{q_a, q_r\}$ et pour tout $a \in \Gamma$, on a,

$$\left| \{(r,b,D) \in Q \times \Gamma \times \{\triangleleft, \triangledown, \triangleright\} \mid (q,a,r,b,D) \in \delta\} \right| = 1$$

Exercice 8 Soit $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ une machine déterministe. Montrer que pour tout mot $w \in \Sigma^*$, il existe *au plus* une exécution de $\mathscr{M}$ sur w. Est-il vrai qu'il en existe toujours une ? ▪

Définition 6 (Fonction associée à une machine déterministe)

Soit $\mathscr{M} = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ une machine déterministe. On associe une fonction partielle $f : \Sigma^* \to \Sigma^*$. Soit $w \in \Sigma^*$, le mot $f(w) \in \Sigma^*$ est défini si il existe une exécution acceptante de $\mathscr{M}$ sur w. Dans ce cas $f(w)$ est le plus grand mot dans Σ^* écrit sur la bande de la machine dans la configuration finale acceptante de cette *unique* exécution. On dit que $f : \Sigma^* \to \Sigma^*$ est la fonction *calculée par* $\mathscr{M}$.

Enfin une fonction partielle quelconque $f : \Sigma^* \to \Sigma^*$ est dite *calculable* si il existe une machine déterministe $\mathscr{M}$ telle que f est la fonction calculée par $\mathscr{M}$.

Exercice 9 On utilise l'alphabet $\Sigma = \{0,1\}$. Décrire une machine déterministe qui, pour tout mot $w \in \Sigma^*$ passé en entrée (par ex. 010110), retourne la séquence duale (par ex. 101001) : chaque 0 est transformé en 1 et vice-versa. ▪

Exercice 10 On utilise l'alphabet $\Sigma = \{0,1\}$. Décrire une machine déterministe qui, pour tout mot $w \in \Sigma^*$ passé en entrée (par ex. 0101100), retourne le miroir de w (par ex. 0011010). ▪

Exercice 11 On utilise l'alphabet $\Sigma = \{0,1\}$ pour coder des entiers en binaire. On considère la fonction $mul_2 : \Sigma^* \to \Sigma^*$ définie par $mul_2(\varepsilon) = \varepsilon$ et pour tout mot non-vide $x \in \Sigma^* \setminus \{\varepsilon\}$, $mul_2(x)$ est égal au codage binaire de l'entier $2 \times x$. Montrer que mul_2 est calculable.

Reprendre l'exercice en utilisant la représentation unaire (par ex. le nombre 5 est codé en représentation unaire par 11111). ▪

Exercice 12 On utilise l'alphabet $\Sigma = \{0,1\}$ pour coder des entiers en binaire. On considère la fonction successeur $succ : \Sigma^* \to \Sigma^*$. C'est-à-dire que $succ(\varepsilon) = \varepsilon$ et pour tout mot non-vide $x \in \Sigma^* \setminus \{\varepsilon\}$, $succ(x)$ est égal au codage binaire de l'entier $x+1$. Montrer que $succ$ est calculable. ▪

Exercice 13 Soit Σ un alphabet. Montrer que pour toute fonction calculable $f : \Sigma^* \to \Sigma^*$, il existe une infinité de machines de Turing qui calculent f. ∎

Exercice 14 Montrer que pour toute machine de Turing $\mathscr{M}$ (déterministe ou non), on peut construire une machine *déterministe* $\mathscr{M}'$ telle que $L(\mathscr{M}') = L(\mathscr{M})$. Montrer que si $\mathscr{M}$ est totale, alors on peut choisir $\mathscr{M}'$ comme une machine déterministe totale également. ∎

3 Machines de Turing à plusieurs bandes

Syntaxe. Soit $k \geqslant 1$. Une machine de Turing à k bandes est un $(\Sigma, \Gamma, Q, q_i, q_a, q_f, \delta)$. Ces objets sont similaires à ceux d'une machine classique. La différence est dans l'ensemble de transitions δ qui doit maintenant gérer k bandes (alors qu'une machine classique n'en gère qu'une seule) :

$$\delta \subseteq (Q \setminus \{q_a, q_r\}) \times \Gamma^k \times Q \times (\Gamma \times \{\triangleleft, \triangleright, \nabla\})^k.$$

De plus, on généralise la condition sur l'ensemble de transitions imposée aux machines classiques de la façon suivante. Pour tout $(q, (a_1, \ldots, a_k), r, (b_1, D_1), \ldots, (b_k, D_k)) \in \delta$ et pour tout $j \leqslant k$, si $a_j = \$$, alors $b_j = \$$ et $D_j \in \{\triangleright, \nabla\}$.

Sémantique. La sémantique des machines à plusieurs bandes est naturellement adaptée de celle des machines de Turing classiques : les configurations d'une machine de Turing à k bandes contiennent k bandes mémoire et k pointeurs. Dans la configuration initiale, les k pointeurs valent 1, les k cases en position 0 contiennent le symbole \$, et le mot d'entrée est écrit sur la première bande à partir de la position 1 (les autres bandes ne contenant que des symboles $\square$ après le \$). Une transition $(q, (a_1, \ldots, a_k), r, (b_1, D_1), \ldots, (b_k, D_k))$ peut s'appliquer lorsque la machine est dans l'état q et que les lettres lues aux cases déterminées par les pointeurs sont $a_1, \ldots, a_k$ sur les bandes 1 à k. L'application de cette transition remplace chaque a_i par b_i et modifie le i-ème pointeur selon la valeur de D_i, ce qui produit une *configuration suivante*.

On rappelle le résultat important (vu en cours) sur les machines de Turing à plusieurs bandes : leur pouvoir d'expression est le même que celle des machines classiques.

Théorème 7

Soit Σ un alphabet et $L \subseteq \Sigma^*$. Les deux propriétés suivantes sont équivalentes :

1. L est accepté par une machine de Turing.
2. il existe $k \geqslant 1$ tel que L est accepté par une machine de Turing à k bandes.

Il existe un résultat similaire pour les machines déterministes et les fonctions calculables.

Théorème 8

Soit Σ un alphabet et $f : \Sigma^* \to \Sigma^*$ une fonction partielle. Les deux propriétés suivantes sont équivalentes :

1. f est la fonction calculée par une machine déterministe.
2. il existe $k \geqslant 1$ tel que f est la fonction calculée par une machine déterministe à k bandes.

Le Théorème 7 implique que les machines de Turing à plusieurs bandes définissent aussi les notion de décidabilité d'un langage et de calculabilité d'une fonction. On pourra donc les utiliser librement.

Exercice 15 On utilise l'alphabet $\Sigma = \{0, 1\}$. Refaire l'exercice des palindromes avec une machine à deux bandes qui n'écrit pas sur sa première bande (l'entrée n'est jamais modifiée). ∎

Exercice 16 On utilise l'alphabet $\Sigma = \{0, 1, \#\}$ et on code des entiers en binaire avec $\{0, 1\}$. Décrire une machine déterministe qui calcule la fonction addition $add : \Sigma^* \to \Sigma^*$. C'est-à-dire que pour tout mot w de la forme $x\#y$ où $x, y \in \{0, 1\}^* \setminus \{\varepsilon\}$ codent deux entiers (par ex. 101#11, pour $x = 5$ et $y = 3$), $add(w)$ est le codage en binaire de l'entier $x + y$ (par ex. 1000 pour $x + y = 8$). ∎

Exercice 17 On utilise l'alphabet $\Sigma = \{0,1\}$ pour coder des entiers. Décrire une machine déterministe qui transforme les codages unaires en codages binaires (par ex. 11111 en 101). ∎

Exercice 18 On utilise l'alphabet $\Sigma = \{0,1\}$ pour coder des entiers. Décrire une machine déterministe qui calcule la fonction exponentielle $f : x \mapsto 2^x$, c'est-à-dire qui prend en entrée un entier naturel x codé en binaire et retourne en sortie le codage binaire de 2^x. ∎

Exercice 19 On utilise l'alphabet $\Sigma = \{0,1,\#\}$ et on code des entiers en binaire avec $\{0,1\}$. Décrire une machine déterministe qui calcule la fonction multiplication $mul : \Sigma^* \to \Sigma^*$. C'est-à-dire que pour tout mot w de la forme $x\#y$ où $x, y \in \{0,1\}^* \setminus \{\varepsilon\}$ codent deux entiers (par ex. 101#11, pour $x = 5$ et $y = 3$), $mul(w)$ est le codage en binaire de l'entier $x \times y$ (par ex. 1111 pour $x \times y = 15$). ∎

Exercice 20 On utilise l'alphabet $\Sigma = \{0\}$. Décrire une machine de Turing qui accepte les mots dont la longueur est un carré. ∎

Exercice 21 Bande bi-infinie On considère une variante des machines de Turing classiques (*i.e.*, à une seule bande) qui utilise une bande bi-infinie. On supprime le symbole spécial \$ et les contraintes associées sur la fonction de transition, de sorte que les machines peuvent lire et écrire aussi loin que nécessaire vers la gauche. Montrer que ces machines peuvent être simulées par les machines classiques. ∎