Théorie de la complexité – Feuille de Travaux Dirigés n° 1

1 Machines de Turing

Syntaxe. Une machine de Turing est un tuple $(\Sigma, \Gamma, Q, q_i, q_f, \delta)$ où :

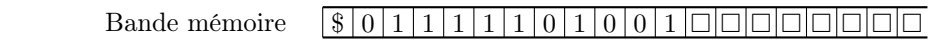
- Σ est l'alphabet d'entrée, il contient les symboles avec lesquels on écrit l'entrée qu'on donne à la machine.
- Γ est l'alphabet de travail, il étend Σ ($\Sigma \subseteq \Gamma$) avec des symboles supplémentaires qu'on peut utiliser pour le calcul. De plus, Γ contient deux symboles spéciaux $\square$ 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).
- $q_f \in Q$ est l'état final (celui dans lequel le calcul s'arrête).
- δ est la fonction de transition, c'est elle qui constitue le programme. On a :

$$\delta : (Q \setminus \{q_f\}) \times \Gamma \to Q \times \Gamma \times \{\triangleleft, \triangleright, \triangledown\}$$

  De plus, on demande que pour tout état $r \in Q \setminus \{q_f\}$, on a $\delta(r, \$) \in Q \times \{\$\} \times \{\triangleright, \triangledown\}$.

Sémantique. Considérons, une machine de Turing $M = (\Sigma, \Gamma, Q, q_i, q_f, \delta)$. Pour tout mot $w \in \Sigma^*$, on va définir l'exécution de M sur le mot d'entrée w. Pour cela, on a besoin de la notion de *configuration*. Une configuration de la machine 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 :



  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.

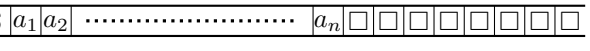
Une configuration est dite *finale* quand l'état de contrôle q est l'état final de M (*i.e.*, $q = q_f$).

Pour toute configuration non finale $C = (q, B, n)$ de M on associe une configuration suivante C'. Soit $a \in \Gamma$ la lettre contenue dans la case n sur la bande mémoire B et $(q', a', d) = \delta(q, a)$ (avec $d \in \{\triangleleft, \triangleright, \triangledown\}$). La configuration C' est définie comme $C' = (q', B', n')$ où B' est la bande obtenue à partir de B en remplaçant la lettre a en case n par a' et,

$$n' = \begin{cases} n-1 & \text{si } d = \triangleleft \\ n & \text{si } d = \triangledown \\ n+1 & \text{si } d = \triangleright \end{cases}$$

On notera $C \to C'$ pour signifier que C' est la configuration suivante de C.

Soit $w = a_1 \cdots a_n \in \Sigma^*$ avec $a_1, \ldots, a_n \in \Sigma$ un mot arbitraire, l'exécution de M sur w est une séquence de configurations définie de la façon suivante. On part de la configuration $C_0 = (q_i, B, 1)$ où B est la bande mémoire suivante contenant le mot w suivit de symboles "$\square$" :



On dit dans ce cas que w est ***l'entrée*** de M. L'exécution de M sur w est la séquence de configurations $C_0, C_1, C_2, \ldots$ telle que $C_0 \to C_1 \to C_2 \to \cdots$. Cette séquence peut être *finie* ou *infinie* suivant que l'on atteint une configuration finale ou non. Si la séquence est finie, on dit que l'exécution de M sur w ***termine***. De plus, dans ce cas, l'exécution ***retourne*** un nouveau mot $v \in \Sigma^*$: v est le plus grand mot de Σ^* écrit sur la partie gauche de la bande mémoire dans la configuration finale de l'exécution (*i.e.*, le plus grand mot de Σ^* qui suit le symbole "\$").

On peut maintenant associer une fonction **partielle** $f : \Sigma^* \to \Sigma^*$ à M. Étant donné $w \in \Sigma^*$, $f(w)$ est égale au mot $v \in \Sigma^*$ retourné par l'exécution de M sur w quand celle-ci termine (quand l'exécution ne termine pas, $f(w)$ est non définie).

Definition 1 Étant donné un alphabet Σ, une fonction (totale) $f : \Sigma^* \to \Sigma^*$ est dite ***calculable*** (ou ***récursive***) quand elle peut s'implémenter par une machine de Turing.

Exercice 1 On utilise l'alphabet $\Sigma = \{0, 1\}$. Décrire une machine de Turing qui, partant du mot vide $\varepsilon \in \Sigma^*$, ne s'arrête pas et écrit 0101010... sur sa bande. ■

Exercice 2 On utilise l'alphabet $\Sigma = \{0, 1\}$. Décrire une machine de Turing qui, pour tout mot $w \in \Sigma^*$ passé en entrée (par ex. 0101100), retourne la séquence duale (par ex. 1010011) : chaque 0 est transformé en 1 et vice-versa. ■

Exercice 3 On utilise l'alphabet $\Sigma = \{0, 1\}$. Décrire une machine de Turing qui, pour tout mot $w \in \Sigma^*$ passé en entrée (par ex. 0101100), retourne le miroir de w (par ex. 0011010). ■

Exercice 4 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 5 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 6 Soit Σ un alphabet. Montrer que pour toute fonction calculable $f : \Sigma^* \to \Sigma^*$, il existe une infinité de machines de Turing qui implémentent f. ■

Exercice 7 Supposons que l'exécution d'une machine de Turing termine au bout de t étapes de calcul (on parle de temps d'exécution) et que le pointeur a visité s cases mémoires pendant ce calcul (on parle d'espace utilisé). Quelle(s) relation(s) existent entre t et s ?

Nous reviendrons à cette question plus tard dans la partie complexité. ■

2 Machines de Turing de décision

On va surtout s'intéresser à des problèmes de décision. C'est-à-dire que plutôt qu'implémenter une fonction $f : \Sigma^* \to \Sigma^*$, on cherche à décider un sous-ensemble fixé $L \subseteq A^*$ appelé *langage* (on veut une procédure qui teste si un mot d'entrée $w \in \Sigma^*$ appartient à L).

Bien sur, une telle procédure peut se reformuler comme l'implémentation d'une fonction $f : \Sigma^* \to \{Vrai, Faux\}$ (qui associe vrai aux mots de L et faux aux autres) et on peut coder un Booléen par un mot. Ainsi, on pourrait donc utiliser les machines classiques définies ci-dessus. Cependant, il sera plus pratique d'utiliser une variante des machines de Turing qui est directement adaptée à ce type de question.

Syntaxe. Une machine de Turing de décision est un tuple $(\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$. Ces objets sont les mêmes que sur une machine classique. La seule différence est que l'état final q_f a disparu et est maintenant remplacé par deux états finaux distincts $q_a, q_r \in Q$ appelés état acceptant et état rejetant, respectivement.

Sémantique. Le sémantique d'une machine de Turing de décision $M = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ est définie de manière similaire à celle des machines de Turing classiques. La différence est qu'étant donné un mot $w \in \Sigma^*$, l'exécution de M sur w peut maintenant se passer de trois façon différentes :

1) Elle ne termine pas.
2) Elle termine dans l'état de contrôle q_a : on dit que M accepte w.
3) Elle termine dans l'état de contrôle q_r : on dit que M rejette w.

En particulier, contrairement aux machines classiques, les machines de Turing de décision ignorent ce qui est écrit sur la bande dans la configuration finale après une exécution qui a terminé.

Definition 2 Soit Σ un alphabet et $L \subseteq A^*$ un langage. On dit que L est ***décidable*** (ou ***récursif***), quand il existe une machine de Turing M telle que pour tout mot $w \in \Sigma^*$:

— si $w \in L$ alors M accepte w.
— si $w \notin L$ alors M rejette w.

Dans ce cas, on dit que la machine M ***décide*** le langage L. Remarquez que si M décide L, la machine M doit s'arrêter sur tout mot d'entrée.

On rappelle qu'on utilise les langages pour coder des problèmes de décision plus abstraits. Pour cette raison on parlera souvent directement de *problème décidable*.

Exercice 8 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 9 On utilise l'alphabet $\Sigma = \{0, 1\}$. Montrer que le langage des palindromes est décidable. ■

Exercice 10 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 11 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 ? ■

3 Machines de Turing à plusieurs bandes

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

$$\delta : (Q \setminus \{q_f\}) \times \Gamma^k \to Q \times (\Gamma \times \{\triangleleft, \triangleright, \triangledown\})^k$$

De plus, on généralise la condition sur les fonctions de transition imposée aux machines classiques de la façons suivante. Soit $q \in Q \setminus \{q_f\}$ et $a_1, \ldots, a_k \in \Gamma$. De plus, soit $(q', (a'_1, d_1), \ldots, (a'_k, d_k)) = \delta(r, a_1, \ldots, a_k)$. Alors, pour tout $j \leqslant k$, si $a_j = \$$, on a $a'_j = \$$ et $d_j \in \{\triangleright, \triangledown\}$.

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 mémoires et k pointeurs. La notion de configuration suivante est définie de manière naturelle à partir de la fonction de transition.

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

1) f s'implémente par une machine de Turing.
2) il existe $k \geqslant 1$ tel que f s'implémente par une machine de Turing à k bandes.

Le Théorème 1 implique, que les machines de Turing à plusieurs bandes définissent aussi la notion de calculabilité. On pourra donc les utiliser librement.

Enfin, on notera que la définition des machines à plusieurs bandes peut s'adapter aisément aux machines de décision. On laisse le soin au lecteur de le faire.

Exercice 12 On utilise l'alphabet $\Sigma = \{0, 1\}$. Refaire l'exercice des palindromes avec une machine de Turing à deux bandes qui n'écrit pas sur sa première bande (l'entrée n'est jamais modifiée). ■

Exercice 13 On utilise l'alphabet $\Sigma = \{0, 1, \#\}$ et on code des entiers en binaire avec $\{0, 1\}$. Décrire une machine de Turing 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 14 On utilise l'alphabet $\Sigma = \{0, 1\}$ pour coder des entiers. Décrire une machine de Turing qui transforme les codages unaires en codages binaires (par ex. 11111 en 101). ■

Exercice 15 On utilise l'alphabet $\Sigma = \{0, 1\}$ pour coder des entiers. Décrire une machine de Turing 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 16 On utilise l'alphabet $\Sigma = \{0, 1, \#\}$ et on code des entiers en binaire avec $\{0, 1\}$. Décrire une machine de Turing 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 17 On utilise l'alphabet $\Sigma = \{0\}$. Décrire une machine de Turing (de décision) qui accepte les mots dont la longueur est un carré. ■

Exercice 18 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. ■