Notes du cours Calculabilité-Complexité
Premier semestre 2005-06

Hubert Comon-Lundh
comon@lsv.ens-cachan.fr

10 octobre 2006

2 Machines de Turing et problèmes indécidables

2.1 Notations

Un *mot* sur l'alphabet Σ est une suite (éventuellement vide) d'éléments de Σ. Sauf précision contraire, on ne considère que des mots finis.

Si A, B sont des ensembles de mots sur l'alphabet Σ,

- $A + B$ est l'ensemble $A \cup B$
- a (où $a \in \Sigma$) est l'ensemble réduit au mot a
- ϵ est le mot vide
- $A \cdot B$ est l'ensemble $\{w_1 w_2 \mid w_1 \in A, w_2 \in B\}$
- A^* est l'ensemble des mots w tels qu'il existe un entier k (éventuellement nul) et k mots $w_1, \ldots, w_k \in A$ tels que $w = w_1 \cdots w_k$.

2.2 Machines de Turing

Il existe de très nombreux modèles de calcul, représentant plus ou moins bien des architectures matérielles, des circuits ou des langages. Les machines de Turing constituent néanmoins le modèle de référence incontournable, pour des raisons historiques. Il s'agit d'un modèle de très bas niveau dans lequel on ne dispose que d'une structure de données (les mots) et, d'une seule instruction (GOTO) en dehors des lectures et écritures.

Il existe plusieurs variantes des MT... Elles sont tous équivalentes par la thèse de Church, du point de vue de l'expressivité. Mais pas du point de vue de la complexité. On donne une def. ici qui sera ensuite modifiée pour la def. des classes de complexité.

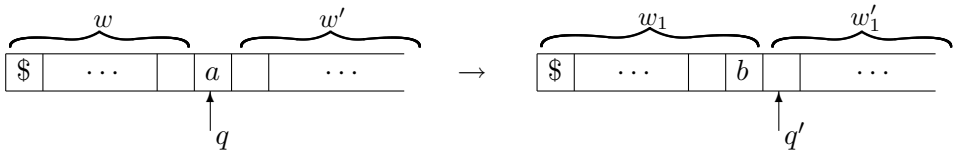
Definition 2.1 *Une* Machine de Turing *est un tuple* $(Q, q_0, \Sigma, \delta, \{B, \$\})$ *où*

- Q *est un ensemble fini d'états*
- $q_0 \in Q$ *est l'état initial*
- Σ *est un alphabet fini*
- $B, \$$ *sont deux symboles spéciaux distincts appartenant à* Σ *(blanc et marqueur de début).*
- δ*, fonction de transition :* $Q \times \Sigma \to Q \cup \{\textbf{\textit{accept}}, \textbf{\textit{reject}}\} \times \Sigma \times \{\leftarrow, \rightarrow, \downarrow\}$ *telle que* $\delta(q, \$) = (q', \$, \rightarrow)$ *: on ne se déplace jamais à gauche du marqueur de début et on ne peut pas l'effacer.*

Remarques :

1. δ est une fonction : son domaine de définition peut n'être qu'un sous-ensemble strict de $Q \times \Sigma$. Dans la suite, on supposera complétée la définition de δ par $\delta(q, a) = (\textbf{reject}, a \downarrow)$ sur les couples où elle est indéfinie
2. Cette définition correspond aux machines *à un seul ruban* : les machines à plusieurs rubans seront abordées ultérieurement
3. Il n'y a pas d'état final dans cette définition. On pourrait aussi considérer **accept** comme unique état final ; il n'y a pas de transition depuis un état final.
4. On accepte dans cette définition de ne pas faire de mouvement (c'est pratique pour compacter quelques définitions)
5. Dans d'autres définitions, on utilise un alphabet "de travail" distinct de l'alphabet d'entrée. Pour les questions que nous abordons, cette distinction n'est pas pertinente.

Mouvement droit



Mouvement gauche

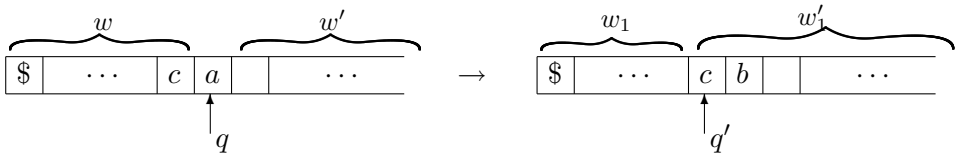


FIG. 1 – Configurations et mouvements des Machines de Turing

6. Nos machines sont *déterministes* : δ est une fonction et non une relation. Pour l'heure, cela nous suffit. Les machines non-déterministes seront introduites quand cela sera nécessaire.

Definition 2.2 *Une* configuration *de la machine* $M = (Q, q_0, \Sigma, \delta, \{B, \$\})$ *est un triplet* (w, q, w') *où* $w, w' \in \Sigma^*, q \in Q \cup \{\textbf{\textit{accept}}, \textbf{\textit{reject}}\}$ *et* $w' \neq \epsilon$.

Étant donné $w_0 \in \Sigma^*$, *la* configuration initiale sur l'entrée *(ou la* donnée*)* w_0 *est* $(\epsilon, q_0, \$w_0)$.

Les configuration finales *sont celles de la forme* (w, q, w') *telles que* $q \in \{\textbf{\textit{accept}}, \textbf{\textit{reject}}\}$.

M peut faire un mouvement *de* (w, q, aw') *vers* (w_1, q_1, w'_1), *ce que l'on note* $(w, q, aw') \vdash (w_1, q_1, w'_1)$ *ssi on est dans l'un des cas suivants :*

- $w_1 = wb, w'_1 = w'B$ *si* $\delta(q, a) = (q_1, b, \rightarrow)$
- $w_1 = w, w'_1 = bw'$ *si* $\delta(q, a) = (q_1, b, \downarrow)$
- $w = w_1 c$, $w'_1 = cbw'$ *si* $\delta(q, a) = (q_1, b, \leftarrow)$.

Les mouvements gauche et droit sont représentés dans la figure 1.

Definition 2.3 *Un* calcul *d'une machine M sur un mot w est une suite de configurations* γ_n *telle que* $\gamma_0 = (\epsilon, q_0, \$w)$ *et* $\forall n > 0.\ \gamma_{n-1} \vdash \gamma_n$.

La suite est finie si et seulement si elle s'achève sur l'un des états $\{\textbf{accept}, \textbf{reject}\}$.

Exemple 1 *Soit M la machine dont la fonction de transition est donnée par la table :*

	$\$$	0	1	B
q_0	$q_0, \$, \rightarrow$	$q_0, 0, \rightarrow$	$q_1, 0, \rightarrow$	$\textbf{\textit{accept}}, 0, \downarrow$
q_1		$q_0, 1, \rightarrow$	$q_1, 1, \rightarrow$	$\textbf{\textit{accept}}, 1, \downarrow$

On montre un calcul de la machine sur le mot d'entrée 0110 *...*

2.3 Fonctions calculables, langages décidables

Si la machine M s'arrête sur la donnée x, soit (w_1, q, w_2) la configuration finale ($q \in \{\textbf{accept}, \textbf{reject}\}$). On note $M(x)$ le mot obtenu en retirant à $w_1 w_2$ le \$ de tête et les blancs de fin de mot. Si M ne s'arrête pas sur la donnée x, par convention, $M(x) = \perp$.

Definition 2.4 *Si f est une application de $D \subseteq (\Sigma \setminus B)^*$ dans Σ^*, M* calcule *f si, pour tout $w \in D$, $M(w) = f(w)$. Si une telle machine de Turing existe, f est dite* calculable.

La notion de fonction calculable s'étend aux entiers. Les entiers sont codés en base 2 par des mots de $0 + 1(0 + 1)^*$; on note $\overline{n}$ le codage de $n \in \mathbb{N}$. Une fonction f de $\mathbb{N}$ dans $\mathbb{N}$ est *calculable* si la fonction qui, pour tout n dans le domaine de définition de f, associe à $\overline{n}$ le mot $\overline{f(n)}$ est calculable.

Pour les fonctions à deux arguments (de $\Sigma^* \times \Sigma^*$ dans Σ^*), on se ramène aux fonctions de $(\Sigma \uplus \{\#\})^*$ dans $(\Sigma \uplus \{\#\})^*$, définies seulement pour les mots $w\#w'$...

Proposition 2.1 *Si les entiers sont des mots de $0 + 1(0 + 1)^*$, les fonctions suivantes sont calculables :*

- $(n, m) \mapsto n + m$
- $(n, m) \mapsto n \times m$
- $(n, m) \mapsto n^m$ *(sauf pour $n, m = 0$)*

Definition 2.5 *Si $L \subseteq (\Sigma \setminus \{B\})^*$, Une M.T. M* décide *L si, pour tout $w \in (\Sigma \setminus \{B\})^*$, il existe un (unique) calcul γ_k tel que $\gamma_0 = (\epsilon, q_0, \$w)$ et, pour un certain n, $\gamma_n = (w_1, \textbf{\textit{accept}}, w_2)$ si $w \in L$ et $\gamma_n = (w_1, \textbf{\textit{reject}}, w_2)$ si $w \notin L$. L est* récursif *s'il existe une M.T. M qui décide L.*

Definition 2.6 *M* accepte *L si, pour tout mot $w \in (\Sigma \setminus \{B\})^*$, ou bien $w \in L$ et il existe un calcul fini s'arrêtant sur une configuration **accept**, ou bien $w \notin L$ et la machine ou bien s'arrête en échec, ou bien ne s'arrête pas sur w. L est* récursivement énumérable *s'il existe une machine de Turing M qui accepte L*

$L(M)$ est alors l'ensemble des mots acceptés par M...

Remarques :

1. L récursif ssi la fonction caractéristique de L est calculable
2. Si L est récursif, alors L est récursivement énumérable.

Théorème 2.2 *Il existe des langages non récursivement énumérables*

On peut utiliser un argument de cardinalité. Mais aussi un langage explicite : w_i est une énumération des mots, M_i une énumération des machines de Turing(ces ensembles étant dénombrables, on admet ici connue une numérotation), $L = \{w_i \mid w_i \notin L(M_i)\}$ n'est pas r.e. En effet, s'il existait une machine M_L telle que, $w \in L$ ssi M_L s'arrête sur w avec succès, alors soit $M_i = M_L$:

$$w_i \notin L(M_i) \Leftrightarrow w_i \in L \Leftrightarrow w_i \in L(M_i)$$

Ce qui est absurde.