Chapitre 6

Machines de Turing et problèmes indécidables

6.1 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 $w = w_1 \cdots w_k$.

6.2 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 6.2.1 *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. Si δ n'est pas définie partout, 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.
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 6.2.2 *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 configurations 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 :*

— $w_1 = wb$ *si* $\delta(q, a) = (q_1, b, \rightarrow)$ *et* $w_1' = w'$ *si* $w' \neq \epsilon$ *et* $w_1' = B$ *sinon*

— $w_1 = w, w_1' = bw'$ *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 6.1.

Definition 6.2.3 *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* $\forall n > 0.\ \gamma_{n-1} \vdash \gamma_n$.

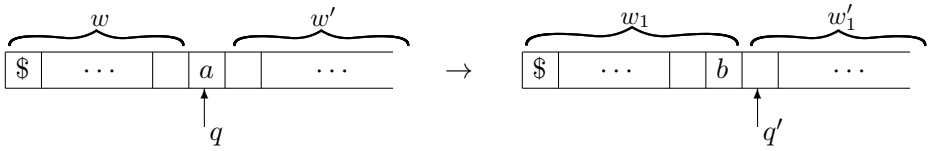
La suite est finie si et seulement si elle s'achève sur l'un des états $\{\textbf{accept}, \textbf{reject}\}$.

Exemple 6.2.1 Soit M la machine dont la fonction de transition est donnée par la table :

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

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

Mouvement droit



Mouvement gauche

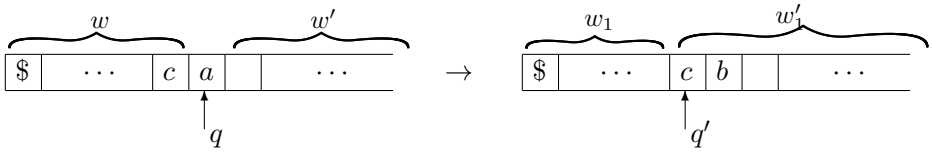


FIGURE 6.1 – Configurations et mouvements des Machines de Turing

Exercice 153 (3)
Donner explicitement la table d'une machine de Turing qui, étant donné un mot $w \in \{0,1\}^*$, accepte w si w contient au moins autant de 0 que de 1 et rejette sinon. (On prendra soin de démontrer que la machine fait bien ce qu'elle est censée faire).

6.3 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 6.3.1 *Si f est une application de $(\Sigma \setminus \{B, \$\})^*$ dans $(\Sigma \setminus \{B, \$\})^*$, M* calcule *f si, pour tout $w \in (\Sigma \setminus \{B, \$\})^*$, $M(w) = f(w)$. Si une telle machine de Turing existe, f est dite* calculable.

On peut étendre cette définition aux fonctions partielles : si f est une application de $D \subseteq (\Sigma \setminus \{B, \$\})^*$ dans $(\Sigma \setminus \{B, \$\})^*$, f est calculable si la fonction f' de $(\Sigma' \setminus \{B, \$\})^*$ dans $(\Sigma' \setminus \{B, \$\})^*$ définie par :
— $\Sigma' = \Sigma \uplus \{\textbf{erreur}\}$
— $f'(w) = \textbf{erreur}$ si $w \notin D$
— $f'(w) = f(w)$ si $w \in D$
est 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 6.3.1 *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 6.3.2 *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 6.3.3 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.

Exercice 154 (3)
Parmi les 3 fonctions suivants (de $\mathbb{N}$ dans $\{0,1\}$) , 2 sont calculables et, pour la 3ème, on ne sait pas actuellement si elle est calculable ou non. Dire (en le justifiant) quelles sont les deux fonctions calculables.

$$f_1(x) = \begin{cases} 0 & \text{Si Dieu existe} \\ 1 & \text{Sinon} \end{cases}$$

$$f_2(x) = \begin{cases} 0 & \text{Si le développement décimal de } \pi \text{ contient au moins } x \text{ 1 consécutifs} \\ 1 & \text{Sinon} \end{cases}$$

$$f_3(x) = \begin{cases} 0 & \text{Si le développement décimal de } \pi \text{ contient exactement } x \text{ 1 consécutifs} \\ 1 & \text{Sinon} \end{cases}$$

Exercice 155
On considère le modèle de calcul des machines de Turing à ruban bi-infini : la définition est la même que la définition 6.2.1, excepté qu'il n'y a pas de symbole spécial \$ (et donc pas d'hypothèse sur les règles correspondantes). Une configuration initiale est un triplet (ϵ, q_0, wB). Une configuration finale est un triplet (w,q,w') avec $q \in \{\textbf{accept}, \textbf{reject}\}$. La relation de transition est donnée comme dans la définition 6.2.2, excepté que, lors d'un mouvement gauche, $w_1 = B$ lorsque $w = c$.

La définition de fonction calculable est la même que pour les machines de Turing avec un ruban infini d'un seul côté : on ignore les blancs dans le résultat.

Montrer qu'une fonction est calculable dans ce modèle si et seulement si elle est calculable par une machine de Turing.

Exercice 156 (Castors affairés)
Pour $n \in \mathbb{N}$, soit $\mathcal{E}_n$ l'ensemble des machines de Turing à ruban bi-infini, sur l'alphabet $\{1, B\}$, à n états (+**accept**,**reject**) qui acceptent le mot vide. Si $M \in \mathcal{E}_n$, on note $f(M)$ le nombre de 1 inscrits sur le ruban quand, sur la donnée ϵ, M s'arrête en acceptant.

On considère la fonction SCORE de $\mathbb{N}$ dans $\mathbb{N}$ définie par $\mathsf{Score}(n) = \max\{f(M) | M \in \mathcal{E}_n\}$.

1. Calculer $\mathsf{Score}(2)$.
2. Montrer que, pour $n \geq 4$, $\mathsf{Score}(n) > 2n$
3. Montrer que Score n'est pas calculable.
4. Montrer que $\mathsf{Score}(3) \geq 6$. (En fait $= 6$).

Théorème 6.3.1 *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.

Proposition 6.3.2 *Si $f : (\Sigma \setminus \{B, \$\})^* \to (\Sigma \setminus \{B, \$\})^*$ est calculable, alors :*

1. *Pour toute fonction g calculable, $g \circ f$ est calculable*
2. *Si L est récursif, alors $f^{-1}(L)$ est récursif*
3. *Si L est récursivement énumérable, alors $f^{-1}(L)$ est récursivement énumérable.*

Si M_f est une machine qui calcule f et M_g calcule g (resp. décide L, resp. accepte L), on construit une machine $M_{g\circ f}$ comme suit : $Q_{g\circ f} = Q_g \uplus Q_f \uplus \{q_i\}$, $q_{0,g\circ f} = q_{0,f}$, $\delta_{g\circ f}$ est définie par :

- Si $q \in Q_f$ et $\delta_f(q,a) = (q',b,d)$ avec $q' \in Q$, alors $\delta_{g\circ f}(q,a) = \delta_f(q,a)$
- Si $q \in Q_f$ et $\delta_f(q,a)$ est indéfini ou bien $\delta_f(q,a) = (q',b,d)$ avec $q' \in \{\mathbf{accept}, \mathbf{reject}\}$, alors $\delta_{g\circ f}(q,a) = (q_i,a,\downarrow)$ (resp. (q_i,b,d))
- $\delta_{g\circ f}(q_i,a) = (q_i,a,\leftarrow)$, pour tout $a \neq \$$
- Si $\delta_g(q_{0,g},\$) = (q,b,d)$, alors $\delta(q_i,\$) = (q,b,d)$
- Si $\delta_g(q,a) = (q',b,d)$ avec $q \in Q_g$, alors $\delta_{g\circ f}(q,a) = (q',b,d)$.

Si $w \in (\Sigma \setminus \{B,\$\})^*$, alors, par hypothèse, M_f s'arrête après n_w étapes et $M_f(w) = f(w)$. La machine $M_{g\circ f}$ arrive après n_w étapes dans une configuration (w_1, q_i, w_2) telle que $w_1 \cdot w_2 = f(w)$. Après au plus $|M_f(w)|$ étapes, la machine $M_{g\circ f}$ est dans la configuration qui suit la configuration initiale de M_g sur $f(w)$.

6.4 Variantes, réductions

Il y en a beaucoup. Par exemple :

Théorème 6.4.1 *Pour toute machine de Turing M, on peut construire une machine de Turing M' qui n'a que deux états et telle que $L(M) = L(M')$*

L'idée est la suivante : On suppose les états de M totalement ordonnés : soient $q_1,\ldots,q_n$ ces états. À une configuration $a_1\cdots a_n, q_i, a_{n+1}\cdots a_m$ de M correspond une configuration $(q_0,a_1,\lhd)\cdots(q_0,a_n,\lhd)(q_i,a_{n+1},\alpha)(q_0,a_{n+2},\rhd)\cdots$ où $\alpha \in \{\lhd_d, \rhd_d, \lhd_i, \rhd_i\}$

Une transition $q_i, a \mapsto q_j, a', \rightarrow$ correspond par exemple aux transitions :

$$\begin{array}{rcll}
Q_0,(q_i,a,\alpha) & \mapsto & Q_1,(q_{j-1},a',\lhd_d),\rightarrow & \\
Q_1,(q_k,b,\rhd) & \mapsto & Q_1,(q_{k+1},b,\rhd_i),\leftarrow & \\
Q_1,(q_k,b,\rhd_i) & \mapsto & Q_1,(q_{k+1},b,\rhd_i),\leftarrow & \\
Q_1,(q_k,b,\lhd_d) & \mapsto & Q_1,(q_{k-1},b,\lhd_d),\rightarrow & \text{Si } k \geq 1 \\
Q_1,(q_0,b,\lhd_d) & \mapsto & Q_0,(q_0,b,\lhd),\rightarrow &
\end{array}$$

Une transition $q_i, a \mapsto q_j, a', \leftarrow$ correspond aux transitions :

$$\begin{array}{rcll}
Q_0,(q_i,a,\alpha) & \mapsto & Q_1,(q_{j-1},a',\rhd_d),\leftarrow & \\
Q_1,(q_k,b,\lhd) & \mapsto & Q_1,(q_{k+1},b,\lhd_i),\rightarrow & \\
Q_1,(q_k,b,\lhd_i) & \mapsto & Q_1,(q_{k+1},b,\lhd_i),\rightarrow & \\
Q_1,(q_k,b,\rhd_d) & \mapsto & Q_1,(q_{k-1},b,\rhd_d),\leftarrow & \text{Si } k \geq 1 \\
Q_1,(q_0,b,\rhd_d) & \mapsto & Q_0,(q_0,b,\rhd),\leftarrow &
\end{array}$$

Il faut en plus une phase d'initialisation (qui n'utilise que Q_0) et considérer les blancs comme $(q_0,B,\rhd)$ dans les transitions ci-dessus.

Au total on utilisera un alphabet $\Sigma' = \Sigma \uplus \{q_0\} \times \Sigma \uplus (Q \uplus \{q_0\}) \times \Sigma \times \{\lhd, \rhd, \lhd_d, \rhd_d, \lhd_i, \rhd_i\}$

On peut aussi (mais pas en plus) limiter l'alphabet à deux symboles (en plus de \$, B)...

6.4.1 Machines de Turing à k rubans

Definition 6.4.1 *Une* machine de Turing à k rubans *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 ensemble fini de symboles*
- $B, \$ \in \Sigma$
- δ *est une application de* $Q \times \Sigma^k$ *dans* $(Q \cup \{\textbf{\textit{accept}}, \textbf{\textit{reject}}\}) \times (\Sigma \times \{\leftarrow, \downarrow, \rightarrow\})^k$ *telle que, si* $\delta(q, (\ldots \$ \ldots)) = (q', (\ldots (a, d) \ldots))$, *alors* $a = \$$ *et* $d \in \{\downarrow, \rightarrow\}$. *(Aucune des têtes de lecture ne se déplace à gauche du marqueur de début correspondant et le \$ n'est jamais effacé).*

Représentation graphique ...

Exemple 6.4.1

	$\$, \$$	$0, \$$	$1, \$$	$\#, \$$	$0, B$	$1, B$	B, B	$0, 0$	$0, 1$	$1, 0$	$1, 1$	$B, 0$	$B, 1$	$B, \$$
q_0	q_0 $\$, \rightarrow$ $\$, \downarrow$	q_0 $0, \rightarrow$ $\$, \downarrow$	q_0 $1, \rightarrow$ $\$, \downarrow$	q_0 $B, \rightarrow$ $\$, \rightarrow$	q_0 $B, \rightarrow$ $0, \rightarrow$	q_0 $B, \rightarrow$ $1, \rightarrow$	q_1 $B, \leftarrow$ $B, \leftarrow$							
q_1	q_2 $\$, \rightarrow$ $\$, \rightarrow$	q_1 $0, \leftarrow$ $\$, \downarrow$	q_1 $1, \leftarrow$ $\$, \downarrow$					q_1 $0, \leftarrow$ $0, \leftarrow$	q_1 $0, \leftarrow$ $1, \leftarrow$	q_1 $1, \leftarrow$ $0, \leftarrow$	q_1 $1, \leftarrow$ $1, \leftarrow$	q_1 $B, \leftarrow$ $0, \leftarrow$	q_1 $B, \leftarrow$ $1, \leftarrow$	q_1 $B, \leftarrow$ $\$, \downarrow$
q_2					q_2 $0, \rightarrow$ $B, \downarrow$	q_2 $1, \rightarrow$ $B, \downarrow$		q_2 $0, \rightarrow$ $0, \rightarrow$	q_2 $0, \rightarrow$ $1, \rightarrow$	q_2 $1, \rightarrow$ $0, \rightarrow$	q_3 $0, \rightarrow$ $0, \rightarrow$	q_2 $0, \rightarrow$ $B, \rightarrow$	q_2 $1, \rightarrow$ $B, \rightarrow$	
q_3					q_2 $1, \rightarrow$ $B, \downarrow$	q_3 $0, \rightarrow$ $B, \downarrow$		q_2 $1, \rightarrow$ $0, \rightarrow$	q_3 $0, \rightarrow$ $1, \rightarrow$	q_3 $0, \rightarrow$ $0, \rightarrow$	q_3 $1, \rightarrow$ $0, \rightarrow$	q_2 $1, \rightarrow$ $B, \rightarrow$	q_3 $0, \rightarrow$ $B, \rightarrow$	

La machine à deux rubans dont la table est donnée ci-dessus calcule la somme de deux nombres en binaire : la donnée est $u\#v$ (écrite sur le premier ruban), où u, v sont deux nombres entiers écrits en binaire de droite à gauche. Dans un premier temps, (en utilisant les états q_0, q_1), la machine écrit v sur le second ruban (et l'efface du premier ruban). Ensuite (en utilisant q_2, q_3), elle effectue la somme en écrivant le résultat sur le premier ruban.

La définition de configuration s'étend à k rubans : il suffit de considérer cette fois les k contenus des rubans et les k positions des têtes de lecture. La configuration initiale ne contient de symboles non blanc ou \$ que sur le premier ruban. la définition de mots acceptés, rejetés, etc.. s'étend. Pour le calcul des fonctions, on distingue un *ruban d'entrée* contenant la donnée et un *ruban de sortie* contenant le résultat.

Definition 6.4.2 *Une machine de Turing I/O est une machine à trois rubans telle que :*

- *On n'écrit jamais sur le premier ruban (ruban de lecture)*
- *Les transitions ne dépendent jamais du contenu du troisième ruban (ruban d'écriture).*

Definition 6.4.3 *Le* temps de calcul *d'une machine de Turing à k rubans sur la donnée w est le nombre de mouvements de la machine de Turing depuis la*

configuration initiale $(q_0, \$w)$ *jusqu'à l'arrêt de la machine. M calcule en temps* $f(n)$ *si, pour une donnée* w *de taille inférieure ou égale à* n*, le temps de calcul de* M *sur* w *est inférieur ou égal à* $f(n)$*.*

*L'*espace de calcul *d'une machine de Turing I/O sur la donnée* w *est la longueur maximale d'un mot (privé de ses blancs finaux) inscrit sur le deuxième ruban, lors du calcul de la machine sur* w*. M calcule en espace* $f(n)$ *si, pour une donnée* w *de taille inférieure ou égale à* n *l'espace de calcul de* M *sur* w *est inférieur ou égal à* $f(n)$*.*

On peut alors définir les classes de complexité :

Definition 6.4.4 *Si le langage* L *(resp. la fonction* f*) est décidé (resp. calculé) par une machine de Turing à* k *rubans en temps* $g(n) \geq n$*, on dit que* $L \in \textbf{\textit{Time}}(g(n))$ *(resp.* $f \in \textbf{\textit{Time}}(g(n))$*).*

Si le langage L *(resp. la fonction* f*) est décidé (resp. calculé) par une machine de Turing I/O en espace* $g(n)$ *on dit que* $L \in \textbf{\textit{Space}}(g(n))$ *(resp.* $f \in \textbf{\textit{Space}}(g(n))$*).*

Si Σ est un ensemble de symboles et $k \in \mathbb{N}$, $\Sigma^{\leq k} = \{w \in \Sigma^* \mid |w| \leq k\}$ et $\Sigma_a^{\leq k} = \{w \in \Sigma^* \mid |w|_a \leq k\}$. La fonction de décalage $d_{a,b,c}$ est l'application qui à un mot wc tel que c n'apparait pas dans w associe $f(w)c$ où f est le morphisme $a \mapsto ba$.

Lemme 6.4.1 *Soit* Σ *un alphabet fini contenant* $\$, B$*. Soient* $a, b, c \in \Sigma \setminus \{\$\}$ *trois symboles distincts. Soit* $k \in \mathbb{N}$*. Il existe une machine de Turing* $M^k_{a,b,c}$ *qui, sur la donnée* wc*,* $w \in \Sigma_a^{\leq k}$*, calcule en temps au plus* $2(|w| + k + 1)$ *le mot* $d_{a,b,c}(w)$ *et s'arrête avec la tête de lecture en début de ruban.*

Preuve:

$M^k_{a,b,c}$ a pour alphabet Σ, ensemble d'états $Q = \{q_x \mid x \in \Sigma^{\leq k}\} \cup \{q_x^c \mid x \in \Sigma^{\leq k}\} \cup \{q_f\}$. L'état initial est q_ϵ. Les transitions de la machine sont données par :

$$\delta(q_x, \alpha) = \begin{cases} q_{y\alpha}, \beta, \rightarrow & \text{si } x = \beta y \text{ et } \alpha \notin \{a, c\} \\ q_\epsilon, \alpha, \rightarrow & \text{si } x = \epsilon \text{ et } \alpha \notin \{a, c\} \\ q_{yba}, \beta, \rightarrow & \text{si } x = \beta y,\ \alpha = a,\ |y| \leq k-2 \\ q_a, b, \rightarrow & \text{si } x = \epsilon,\ \alpha = a \\ q_y^c, \beta \rightarrow & \text{si } x = \beta y,\ \alpha = c \\ q_f, c \leftarrow & \text{si } x = \epsilon, \alpha = c \end{cases}$$

$$\delta(q_x^c, \alpha) = \begin{cases} q_y^c, \beta, \rightarrow & \text{Si } x = \beta y \\ q_f, c, \leftarrow & \text{Si } x = \epsilon \end{cases}$$

$$\delta(q_f, \alpha) = q_f, \alpha, \leftarrow \quad \text{Si } \alpha \neq \$$$

On montre, par récurrence sur $m \leq |w|$ que $\gamma_0 \vdash^{m+1} \gamma_m = (\$w' q_x, , w'')$ avec
— $w_1 w'' = w$ et $|w_1| = m$,
— $w'x = w'_1$ et $|x| = |w_1|_a$,

— w'_1 est l'image de w_1 par le morphisme $\{a \mapsto ba\}$

Par récurrence sur $|x|$, $(q^c_x, w', \epsilon) \vdash^{|x|} (q^c_\epsilon, w'x, \epsilon)$ et enfin, $(q_f, w', \epsilon) \vdash^{|w'|} (q_f, \epsilon, w')$.

Bout à bout, on obtient :

$$\gamma_0 \vdash^{|w|+1} (q_{x_1}, \$w_1, c) \vdash^{|w|_a+1} (q_f, v, \alpha c) \vdash^{|w|+|w|_a} (q_f, \epsilon, v\alpha c)$$

où x_1 est le suffixe de w de longueur $|w|_a$, $v\alpha$ est l'image de $\$w$ par le morphisme $a \mapsto ba$.

On vérifie que le nombre détapes de calcul est inférieur à $|w|+1+k+|w|+k+1 = 2(|w|+k+1)$.

Théorème 6.4.2 *Si M est une machine de Turing à k rubans calculant f en temps $g(n) \geq n$, alors il existe une machine de Turing M' à 1 ruban, calculant f en temps $O(g(n)^2)$.*

Preuve:

Soit $w \in \Sigma^*$, $n = |w|$ et $g(n)$ le nombre d'étapes de calcul de M sur w.

Pour prouver ce théorème, on va construire M' de manière à ce que la configuration q, w_1, w'_1, ... w_k, w'_k où $w'_1, \ldots, w'_k \neq \epsilon$ de M corresponde à la configuration $[q, \epsilon, 0]$, ϵ, $\$'w_1\#w'_1Dw_2\#w'_2D\cdots Dw_k\#w'_kDF$.

Si $M = (q_0, Q, \Sigma, \delta, \{B, \$\}$, on construit $M' = ([q_0, \epsilon, 0], Q', \Sigma', \delta', \{B, \$\})$ comme suit :

— $\Sigma' = \Sigma \uplus \{D, \#, F\}$

— $Q' = Q \times (\Sigma'^{\leq k} \uplus \Sigma'^{\leq k} \times \{\leftarrow, \rightarrow, \downarrow\}^{\leq k}) \times \{0,1,2,3,4,5,6,7,8,9,10\} \uplus Q \times Q_{D,B,F}$ où $Q_{D,B,F}$ est l'ensemble des états de la machine $M_{D,B,F}$ du lemme 6.4.1.

— δ' est défini dans les diverses étapes ci-dessous.

On suppose que le ruban de M' contient initialement $\$'\#\$w(D\#)^{k-1}DF$ où w est la donnée de f.

Chaque étape de calcul de M est simulée par les 6 étapes suivantes :

1. On effectue une transition (en restant sur place) de l'état $[q, \epsilon, 0]$ vers l'état (q, q_i) où q_i est l'état initial de $M_{D,B,F}$
2. On applique $M^k_{D,B,F}$ jusqu'à atteindre sa configuration finale (sans tenir compte de la première composante de l'état, qui reste inchangée)
3. On effectue la transition $\delta'((q, q_f), \S') = ([q, \epsilon, 1], \$', \rightarrow)$, .
4. On collecte les symboles qui suivent le marqueur # (il y en a toujours un, grâce au décalage effectué aux étapes précédentes) :

$$\delta'([q,x,1],\alpha) = \begin{cases} [q,x,1],\alpha,\rightarrow & \text{Si } \alpha \notin \{\#, F\} \\ [q,x,2],\#\rightarrow & \text{Si } \alpha = \# \\ [q,x,3],F,\leftarrow & \text{Si } \alpha = F \end{cases}$$

$$\delta'([q,x,2],\alpha) = [q,x\alpha,1],\alpha,\rightarrow$$

et $\delta'([q,x,3],\alpha) = [q,x,3],\alpha,\leftarrow$ si $\alpha \neq \$'$.

5. On effectue la transition de M : si $\delta(q,x) = (q', y, d)$ alors

$$\delta'([q,x,3],\$') = [q', y, d, 4], \$', \rightarrow$$

où x, y, d sont ici des mots de longueur k.

6. On parcourt le ruban pour la mise à jour des configurations locales :

$$\delta'([q, ax_1, dd_1, 4], \alpha) = \begin{cases} [q, ax_1, dd_1, 4], \alpha, \rightarrow & \text{Si } \alpha \notin \{\#, F\} \\ [q, x_1, d_1, 5], a, \rightarrow & \text{Si } \alpha = \# \text{ et } d = \rightarrow \\ [q, ax_1, d_1, 6], \#, \rightarrow & \text{Si } \alpha = \# \text{ et } d = \leftarrow \\ [q, x, d, 10], F, \leftarrow & \text{Si } \alpha = F \end{cases}$$

Simulation du mouvement droit :

$$\delta'([q, x_1, d_1, 5], \alpha) = [q, x_1, d_1, 4], \#, \rightarrow$$

Simulation du mouvement gauche :

$$\begin{array}{rcl} \delta'([q, ax_1, d_1, 6], \alpha) & = & [q, x_1, d_1, 7], a, \leftarrow \\ \delta'([q, x_1, d_1, 7], \#) & = & [q, x_1, d_1, 8], \#, \leftarrow \\ \delta'([q, x_1, d_1, 8], c) & = & [q, cx_1, d_1, 9], \#, \rightarrow \\ \delta'([q, cx_1, d_1, 9], \#) & = & [q, x_1, d_1, 4], c, \rightarrow \end{array}$$

Retour au départ :

$$\begin{array}{l} \delta([q, \epsilon, \epsilon, 10], \alpha) = [q, \epsilon, \epsilon, 10], \alpha \leftarrow \quad \text{Si } \alpha \neq \$ \\ \delta([q, \epsilon, \epsilon, 10], \$) = [q, \epsilon, 0], \$, \downarrow \end{array}$$

Il faudrait (entre autres) prouver l'invariant suivant :

> Si $\gamma = q, (\$w_1, w_1'), \ldots, (\$w_k, w_k')$ est une configuration de M et $\gamma \vdash_M \gamma'$ avec $\gamma' = q', (\$x_1, x_1'), \ldots, (\$x_k, x_k')$, alors
>
> $$\theta = [q, \epsilon, 10], \epsilon, \$w_1 \# w_1'' D \cdots D w_k \# w_k'' D F$$
>
> est une configuration de M' dans laquelle w_i' et w_i'' ne diffèrent que par des blancs ajoutés ou retirés en fin de mot et $\theta \vdash_{M'}^{N} \theta'$ où $\theta' = [q', \epsilon, 10], \epsilon, \$x_1 \# x_1'' D \cdots D x_k \# x_k'' F$ où x_i'' et x_i' ne diffèrent que par des blancs ajoutés ou retirés en fin de mot.

De plus, si $h = |w_1 w_1'| + \cdots + |w_k w_k'| + 2k$, $N \leq N_1 + \cdots + N_6$ où chacun des N_i correspond au nombre de transitions de chacune des étapes décrites ci-dessus :

— $N_1 = N_3 = N_5 1$
— $N_2 \leq 2(h + k + 1)$ d'après le lemme 6.4.1
— $N_4 = 2(h + k + 1)$
— $N_6 \leq$

Cette phase comporte au plus $2 \times g(n) + 5k$ étapes. De plus $N \leq 6 \times g(n) + 7k$.

Corollaire 6.4.1 *Les fonctions calculables (resp. les langages r.é.) pour une machine à k rubans sont les mêmes que pour une machine à un ruban.*

Il suffit de remarquer que M' ne s'arrête pas sur x ssi M ne s'arrête pas sur x.

Definition 6.4.5 *Un langage est co-récursivement énumérable si son complémentaire est récursivement énumérable.*

Théorème 6.4.3 *Un langage est récursif ssi il est récursivement énumérable et co-récursivement énumérable.*

Si un langage est récursif, alors son complémentaire aussi. Un sens de l'implication est donc immédiat. Si maintenant L est récursivement énumérable et co-récursivement énumérable, soient M_L et $\overline{M_L}$ les machines qui acceptent respectivement L et $\overline{L}$. On construit alors une machine M à deux rubans, qui commence par recopier sa donnée sur le second ruban, puis effectue simultanément un mouvement de M_L sur le premier ruban et un mouvement de $\overline{M_L}$ sur le second ruban. Dès que l'une des machines va entrer dans **accept**, la machine M s'arrête : en **accept**si c'est M_L qui a accepté, et en **reject**si c'est $\overline{M_L}$ qui a accepté. On définit dualement les transitions lorsque l'une des machines M_L ou $\overline{M_L}$ est sur le point de rejeter.

Pour toute entrée w, ou bien $w \in L$ et M_L s'arrête en **accept**(et $\overline{M_L}$ ou bien ne s'arrête pas ou bien s'arrête en **reject**). Dans ce cas, M s'arrête, ou bien quand M va accepter ou bien quand $\overline{M_L}$ va rejeter. Dans les deux cas M accepte w.

Ou bien $w \notin L$ et dans ce cas $\overline{M_L}$ s'arrête en **accept**sur w (et M_L pu bien ne s'arrête pas, ou bien s'arrête en **reject**). Dans ce cas, comme ci-dessus, M s'arrête en **reject**.

Donc, sur toute entrée w, M s'arrête. De plus $w \in L$ ssi M accepte w.

Exercice 157 (3)
Montrer que la classe des langages récursifs est close par les opérations Booléennes : intersection, union, complémentaire.

Exercice 158 (4)
Soit f une fonction calculable. Montrer que l'image de f est un langage récursivement énumérable.

Exercice 159 (4)
Montrer que la classe des langages récursivement énumérables est close par union et intersection. (Mais pas par complémentaire, comme le montre l'exemple du langage universel).

Exercice 160 (5)
Montrer que L est un langage récursivement énumérable, si et seulement s'il existe une machine de Turing M à deux rubans et un état q_e de M tels que,

$(\epsilon, q_0, \$, \$) \vdash_M^* (\epsilon, q_e, \$w', \$w)$ si et seulement si $w \in L$.

Exercice 161 (5)
Montrer que, si f est calculable et L est récursivement énumérable, alors $f(L)$ est récursivement énumérable.