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 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*
- $\Sigma$ *est un ensemble fini de symboles*
- $B, \$ \in \Sigma$
- $\delta$ *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,* $\delta(q, (\ldots \$ \ldots)) = (q', (\ldots (\$, \rightarrow) \ldots))$. *(Aucune des têtes de lecture ne se déplace à gauche du marqueur de début correspondant).*

Représentation graphique ...

**Exemple 6.4.1**

| | $\$,\$$ | $0,\$$ | $1,\$$ | $\#,\$$ | $0,B$ | $1,B$ | $B,B$ | $0,0$ | $0,1$ | $1,0$ | $1,1$ | $B,0$ | $B,1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_0$ | $q_0$<br>$\$,\rightarrow$<br>$\$,\downarrow$ | $q_0$<br>$0,\rightarrow$<br>$\$,\downarrow$ | $q_0$<br>$1,\rightarrow$<br>$\$,\downarrow$ | $q_1$<br>$B,\rightarrow$<br>$\$,\rightarrow$ | $q_0$<br>$B,\rightarrow$<br>$0,\rightarrow$ | $q_0$<br>$B,\rightarrow$<br>$1,\rightarrow$ | $q_1$<br>$B,\leftarrow$<br>$B,\leftarrow$ | | | | | | |
| $q_1$ | $q_2$<br>$\$,\rightarrow$<br>$\$,\rightarrow$ | $q_1$<br>$\$,\leftarrow$<br>$\$,\downarrow$ | $q_1$<br>$\$,\leftarrow$<br>$\$,\downarrow$ | | | | | $q_1$<br>$0,\leftarrow$<br>$0,\leftarrow$ | $q_1$<br>$0,\leftarrow$<br>$1,\leftarrow$ | $q_1$<br>$1,\leftarrow$<br>$0,\leftarrow$ | $q_1$<br>$1,\leftarrow$<br>$1,\leftarrow$ | $q_1$<br>$B,\leftarrow$<br>$0,\leftarrow$ | $q_1$<br>$B,\leftarrow$<br>$1,\leftarrow$ |
| $q_2$ | | | | | $q_2$<br>$0,\rightarrow$<br>$B,\downarrow$ | $q_2$<br>$1,\rightarrow$<br>$B,\downarrow$ | | $q_2$<br>$0,\rightarrow$<br>$0,\rightarrow$ | $q_2$<br>$0,\rightarrow$<br>$1,\rightarrow$ | $q_2$<br>$1,\rightarrow$<br>$0,\rightarrow$ | $q_3$<br>$0,\rightarrow$<br>$0,\rightarrow$ | $q_2$<br>$0,\rightarrow$<br>$B,\rightarrow$ | $q_2$<br>$1,\rightarrow$<br>$B,\rightarrow$ |
| $q_3$ | | | | | $q_2$<br>$1,\rightarrow$<br>$B,\downarrow$ | $q_3$<br>$0,\rightarrow$<br>$B,\downarrow$ | | $q_2$<br>$1,\rightarrow$<br>$0,\rightarrow$ | $q_3$<br>$0,\rightarrow$<br>$1,\rightarrow$ | $q_3$<br>$0,\rightarrow$<br>$0,\rightarrow$ | $q_3$<br>$1,\rightarrow$<br>$0,\rightarrow$ | $q_2$<br>$1,\rightarrow$<br>$B,\rightarrow$ | $q_3$<br>$0,\rightarrow$<br>$B,\rightarrow$ |

La machine à deux rubans dont la table est donnée ci-dessus calcula 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 $\Sigma$ 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 $\Sigma$ 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 $\Sigma$, 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 $q_\epsilon$. 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 $\gamma_0 \vdash^{m+1} \gamma_m = (q_x, \$w', w'')$ 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 $|x|$, $(q_x^c, w', \epsilon) \vdash^{|x|} (q_\epsilon^c, w'x, \epsilon)$ 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 $q,\ w_1, w_1',\ \ldots\ w_k, w_k'$ où $w_1', \ldots, w_k' \neq \epsilon$ de $M$ corresponde à la configuration $[q_0, \epsilon, 0],\ \epsilon,\ w_1 \# w_1' D w_2 \# w_2' D \cdots D w_k \# w_k' DF$.

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.

— $\delta'$ est défini dans les diverses étapes ci-dessous.

On suppose que le ruban de $M'$ contient initialement $\#\$w(D\#)^{k-1}F$ 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 $M_{D,B,F}$ jusqu'à atteindre sa configuration finale (sans tenir compte de la première composante de l'état, qui reste inchangée)
3. On effectue une transition (en restant sur place) de $(q, q_f)$ vers $[q, \epsilon, 1]$.
4. On collecte les symboles qui suivent le marqueur # (il y en a toujours un, grâce au décalage effectué aux étapes précédentes) :

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

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

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

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

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

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

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

Simulation du mouvement droit :

$$\begin{array}{rcl} \delta'([q, ax_1, d_1, 5], \alpha) & = & [q, x_1, d_1, 7], a, \rightarrow \\ \delta'([q, x, d, 7], \alpha) & = & [q, x, d, 4], \alpha, \rightarrow \end{array}$$

Simulation du mouvement gauche :

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

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 $\theta \vdash^N_{M'} \theta'$ 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 alternativement 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 symétriquement 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 155 (3)**
Montrer que la classe des langages récursifs est close par les opérations Booléennes : intersection, union, complémentaire.

**Exercice 156 (4)**
Soit $f$ une fonction calculable. Montrer que l'image de $f$ est un langage récursivement énumérable.

**Exercice 157 (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 158 (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 159 (5)**
Montrer que, si $f$ est calculable et $L$ est récursivement énumérable, alors $f(L)$ est récursivement énumérable.

## 6.5 Machine de Turing universelle

On peut construire une M.T. *universelle* $U$ qui,étant donné $< M, w >$ exécute $M$ sur $w : U(< M, w >) = M(w)$. Le codage $< M, w >$ est défini par :

— $< M, w >=< M >; < w >$

— $< (Q, q_0, \Sigma, \delta, \{B, \$\}) >=< Q > d_\Sigma < \Sigma > d_\delta < \delta > f_\delta$

— $< Q >= \overbrace{0 \cdots 0}^{1+\lfloor \log_2 |Q| \rfloor}$ ; il y a autant de zéros que dans l'écriture en binaire du nombre d'éléments de $Q$. On suppose les états codés par des entiers $1, ..., |Q|$ et $q_0 = 1$, avec des 0 devant de façon à avoir tous même longueur

— $< \Sigma >= \overbrace{0 \cdots 0}^{\lfloor \log_2 |\Sigma| \rfloor}$ On suppose que $B$ et \$ correspondent aux entiers 1 et 10 en binaire. Tous les symboles de $\Sigma$ sont représentés par des entiers en binaire, complétés le cas échéant par des 0 à gauche, de manière à ce qu'ils aient tous même longueur.

— $< \delta >$ est codé comme suit : c'est une séquence de règles écrites sous la forme $c(q), c(a) \mapsto c(q'), c(b), m$; où le codage des états et symboles est par leur représentation binaire

— $< aw >= c(a) < w >$.

Le codage $< M, w >$ peut aussi n'utiliser qu'un alphabet de deux symboles : il suffit de recoder l'ensemble des symboles utilisés en binaire, ce que nous ne faisons pas pour des raisons de lisibilité.