Proposition 2.3 *Si $f : (\Sigma \setminus \{B, \$\})^* \to (\Sigma\{B, \$\})^*$ 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 initiale de M_g sur $f(w)$.

2.4 Variantes, réductions

Il y en a beaucoup. Par exemple :

Théorème 2.4 *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)...

2.5 Machines de Turing à k rubans

Definition 2.7 *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,* $\delta(q, (\ldots \$ \ldots)) = (q', (\ldots (\$, \rightarrow) \ldots))$. *(Aucune des têtes de lecture ne se déplace à gauche du marqueur de début correspondant).*

Représentation graphique ...

Exemple 2

	$\$, \$$	$0, \$$	$1, \$$	$\#, \$$	$0, B$	$1, B$	B, B	$0, 0$	$0, 1$	$1, 0$	$1, 1$	$B, 0$	$B, 1$
q_0	q_0 $\$, \rightarrow$ $\$, \downarrow$	q_0 $0, \rightarrow$ $\$, \downarrow$	q_0 $1, \rightarrow$ $\$, \downarrow$	q_1 $B, \rightarrow$ $\$, \rightarrow$	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 $\$, \leftarrow$ $\$, \downarrow$	q_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_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 $0, \rightarrow$ $B, \rightarrow$

Exemple : addition de deux nombres écrits en binaire de droite à gauche.

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 2.8 *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 2.9 *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 2.10 *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. $f \in \textbf{\textit{Space}}(g(n))$).

Théorème 2.5 *Si M est une machine de Turing à k rubans calculant en temps $f(n) \geq n$, alors il existe une machine de Turing M' à 1 ruban, calculant en temps $O(f(n)^2)$ et telle que $M(x) = M'(x)$ pour tout x.*

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 $[q, (), (), 0],\ \epsilon,\ w_1 \# w'_1 D w_2 \# w'_2 D \cdots D w_k \# w'_k F$.

Pour simuler une transition de M, dans une première étape on parcourt tout le ruban et on collecte les premiers symboles de $w'_1, \ldots, w'_k$, pour atteindre l'état $[q, (a_1, \ldots, a_k), (), 0]$. On revient ensuite en début de ruban pour atteindre l'état $[q, (a_1, \ldots, a_k), (), 1]$. Si $q, (a_1, \ldots, a_k) \mapsto_M q', (b_1, \ldots, b_k), (d_1, \ldots, d_k)$, on passe dans un état $[q', (b_1, \ldots, b_k), (d_1, \ldots, d_k), 2]$. On parcourt ensuite à nouveau l'intégralité du ruban en mettant à jour les positions des # et les écritures sur les codages de ruban, jusqu'à atteindre $[q', (), (), 2]$. On passe alors dans l'état $[q', (), (), 3]$ et on se place en début de ruban jusqu'à atteindre $[q', (), (), 4]$. On gère alors les décalages afin de concaténer B à droite des nouveaux w'_i (Ceci peut nécessiter jusqu'à k symboles dans la mémoire locale). Enfin, on retourne en début de ruban, dans l'état $[q', (), (), 0]$.

En début de calcul, il faut construire la configuration initiale et en fin de calcul nettoyer le ruban en ne gardant que le mot de sortie (au lieu d'accepter directement).

En fin de compte, si la donnée est de longueur n, M' simule M en effectuant $O(n)$ opérations d'initialisation, puis simule chacune des transitions de M en un temps $O(l)$ où l est la longueur de la configuration. Celle-ci est majorée par $k \times (n + g(n))$. Donc chaque mouvement est simulé par $O(g(n))$ mouvements. D'où le résultat.

Corollaire 2.6 *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 2.11 *Un langage est co-récursivement énumérable si son complémentaire est récursivement énumérable.*

Théorème 2.7 *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.

2.6 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 Σ 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
- $< aw >= c(a) < w >$.