**UNIVERSITE GRENOBLE ALPES** **M1-info**

**Complexité** **Année 2016-17**

**TD-1 — Machines de Turing simples** — corrigé (partiel)

---

*Machines de Turing simples*

---

On s'intéresse d'abord au modèle le plus simple de machine de Turing (MT), à quadruplets. Dans un état $q_i$ et lisant un symbole $S_j$, la MT passe dans un état $q_l$ après avoir effectué l'une des 3 actions suivantes : écrire un symbole $S_k$, aller à droite ($R$), ou aller à gauche ($L$). Un quadruplet "simple" est donc de forme ($q_i \quad S_j \quad S_k \mid L \mid R \quad q_l$).

On code un entier $x$ sur la bande par $\overline{x} = \overbrace{||\cdots|}^{x+1 \text{ fois}} B$,
et un vecteur $\vec{x} = (x_1 \dots x_n)$ par $\overline{\vec{x}} = \overline{x_1} \dots \overline{x_n} = |^{x_1+1}B|^{x_2+1}B \dots |^{x_n+1}B$.
Le calcul de la machine $Z$ sur l'entrée $\vec{x}$ commence avec la configuration initiale
$Init(\vec{x}) = q_0\overline{\vec{x}} = \dots BBBq_0|^{x_1+1}B|^{x_2+1}B \dots |^{x_n+1}BBBB \dots$
Si le calcul s'arrête dans une configuration $u = vqw$, le résultat est le nombre de | sur la bande, soit$\Psi_Z^{(n)}(\vec{x}) = \#(u, |)$ (noté souvent $\langle u \rangle$).

1. Quelles fonctions calcule la machine $[q_0BBq_0]$ ?
   ▶ $\Psi_Z^{(0)} = \uparrow$ (indéfinie, car $Z$ boucle).
   Par contre, si $n > 0$, $\Psi_Z^{(n)} = \lambda x_1 \dots x_n.[\sum_{i=1}^{n}(x_i + 1)]$ est totale. ◀
2. Quelles fonctions calcule la machine $[q_0||q_0]$ ?
   ▶ $\Psi_Z^{(0)} = 0$ (car $Z$ est bloquée dès le départ en $q_0B$).
   Par contre, $\Psi_Z^{(n+1)} = \uparrow$ pour tout $n$. ◀
3. Quelles fonctions calcule la machine

$$Z = \begin{bmatrix} q_0 & | & B & q_0 \\ q_0 & B & R & q_1 \\ q_1 & | & R & q_1 \\ q_1 & B & R & q_2 \\ q_2 & | & | & q_0 \end{bmatrix}$$

   pour toutes les arités $n$ ? Quelle est sa complexité (le nombre des pas de calculs en fonction de la taille de l'entrée) ?
   ▶ $\Psi_Z^{(0)} = 0$
   Sur une entrée $|^{x_1+1}B|^{x_2+1}B \dots |^{x_n+1}B$, le fonctionnement de $Z$ consiste en 2 boucles imbriquées :
   — tant qu'on lit | :
   — remplacer le premier | par B et avancer d'une case
   — tant qu'on lit | : avancer d'une case.
   — avancer d'une case supplémentaire
   — si on lit |, passer dans l'état $q_0$
   Dans tous les cas, la machine s'arrête et la fonction calculée est totale :
   - $\Psi_Z^{(0)} = 0$

- si $n > 0$, $\Psi_Z^{(n)} = \lambda x_1 \dots x_n.[\sum_{i=1}^{n}(x_i)]$.

Pour n = 1, on passe une seule fois dans la boucle principale, et le nombre de pas de calcul est : 2 (pour effacer le premier B et avancer vers la droite) + $x_1$ déplacements vers la droite + 1 (dernier déplacement vers la droite)= $x_1$ + 3, soit $t+2$ où $t$ est la taille de l'entrée (i.e., $x_1+1$).
Pour n > 1, le nombre de pas de calculs est : $\Sigma_{i=1}^{n-1}(2 + x_i + 2)$ + ( 2 + $x_n$ + 1)
(n passages dans la boucle principale, pour l'itération $i$ sauf la dernière : 2 pas de calculs + $x_i$ pas de calculs pour les déplacements vers la droite + 2, et pour la dernière itération on ne passe pas dans l'état $q_0$ ) .
La taille de l'entrée est $t = \Sigma_{i=1}^{n}(1 + x_i)$ + (n-1) , car il faut comter les (n-1) B séparareurs des blocs.
Donc la complexité est donnée par la fonction $C(t)$= $t$ + $2n$ (où $n$ est le nombre de blocs qui constituent l'entrée).
Comme $0 \leq n \leq t$, on peut en conclure que $t \leq C(t) \leq 3 \times t$.
◄

4. Construire une MT qui calcule $Diff = \lambda xy[x \dot{-} y]$, c'est-à-dire :
$Diff(x, y) = x - y$ si $x \geq y$ et $Diff(x, y) = 0$ sinon. Quelle est sa complexité dans le cas le pire (le nombre maximum des pas de calculs en fonction de la taille de l'entrée) ?
► Sur une entrée $|^{x+1}B|^{x+1}$, le principe est le suivant :
— Tant qu'on lit un | dans le blox codant x et un | dans le bloc codant y (remarque : on passe au moins une fois dans cette boucle)
  — on efface le premier | du bloc x (en le remplacant par B) , on avance jusqu'au dernier | du bloc y (pour le répérer on va jusqu'au B suivant et on recule d'une case), on l'efface (en le remplacant par B), et on recule d'une case.
  — Si on lit B (y est égal à, ou est devenu, 0) passer dans l'état $q_{y=0}$.
  — Sinon,
    — Tant qu'on lit | dans le bloc y on recule d'une case (jusqu'à tomber sur B, qui est le séparateur entre ce qui reste des 2 blocs x et y)
    — on recule d'une case (pour se positionner sur ce qui reste du bloc x)
    — Tant qu'on lit | dans le bloc x on recule d'une case (jusqu'à tomber sur B)
    — On avance d'une case (ce qui nous positionne sur B si x -y est devenu 0, ou sur le premier | de ce qui reste du bloc x-y)
    — Si on lit B, passer dans l'état $q_{x=0}$.
— Si on est dans l'état $q_{x=0}$ (il faut effacer tous les | restants dans le le bloc y, et il en reste car sinon on serait dans l'état $q_{y=0}$) : tant qu'on lit des | on les efface.
D'où le programme suivant :

$Z =$

| | | | | |
|---|---|---|---|---|
| $q_0$ | \| | $B$ | $q_0$ | on efface le premier \| du blox codant x |
| $q_0$ | $B$ | $R$ | $q_1$ | on va jusqu'au début du bloc codant y |
| $q_1$ | \| | $R$ | $q_1$ | |
| $q_1$ | $B$ | $R$ | $q_2$ | |
| $q_2$ | \| | $R$ | $q_2$ | on va jusqu'à la fin du bloc y et on efface le dernier \| |
| $q_2$ | $B$ | $L$ | $q_3$ | |
| $q_3$ | \| | $B$ | $q_3$ | |
| $q_3$ | $B$ | $L$ | $q_4$ | |
| $q_4$ | $B$ | $B$ | $q_{y=0}$ | on s'arrete : aucun quadruplet commencant par $q_{y=0}$ |
| $q_4$ | \| | $L$ | $q_5$ | on recule jusqu'à la fin du bloc x |
| $q_5$ | \| | $L$ | $q_5$ | |
| $q_5$ | $B$ | $L$ | $q_6$ | |
| $q_6$ | $B$ | $B$ | $q_{x=0}$ | si le bloc x-y contient encore des \| |
| $q_6$ | \| | $L$ | $q_7$ | on se positionne au début du bloc et on itère la boucle principale |
| $q_7$ | \| | $L$ | $q_7$ | |
| $q_7$ | $B$ | $R$ | $q_0$ | |
| $q_{x=0}$ | $B$ | $R$ | $q_9$ | on efface tous les \|restants dans le bloc y |
| $q_9$ | \| | $B$ | $q_{10}$ | |
| $q_{10}$ | $B$ | $R$ | $q_9$ | |

Le cas où le nombre de pas de calculs est minimal est quand y vaut 0 : on passe une seule fois dans la boucle principale et on s'arrête juste après avoir effacé le seul | du bloc codant y=0. Le nombre de pas de calculs effectué est alors : 1 (effacement du premier | du bloc x) + (x+1) déplacements pour se positionner au début du bloc codant y + 4 (on se déplace d'une case pour s'apercevoir qu'on n'a qu'un seul | dans y, on recule d'une case et on efface le | et on recule d'une case pour passer à l'état $q_{y=0}$ parce qu'on lit un B). Soit : x+6 pas de calcul.

La taille de l'entrée dans ce cas le meilleur est : n= (x+1) + 1 (taille du codage de y=0) + 1 (le B séparateur).

D'où la complexité dans le cas le meilleur en fonction de la taille n de l'entrée est : n+3.

La complexité dans le cas le pire correspond au cas où le nombre de passages dans la boucle principale est maximal, ce qui correspond au cas où x=y. Le premier passage dans la boucle principale correspond à un nombre de pas de calcul qui est égal à : 1 (effacement du premier | du bloc x) + (x+1) déplacements pour se positionner au début du bloc codant y, puis (y+1) déplacements pour aller au B juste après la fin de Y, puis 3 pas de calculs pour reculer, effacer le dernier | de y, et reculer d'une case, puis (y+1 +x +1) déplacements (pas de calcul de la partie "Sinon") : en tout, 1+x+1+y+1+3 + y+1 +x +1 = 2 × (x+1 + y +1 +1) + 2 = 2 × n + 2, où n=(x+1) +( y +1) +1 est la taille du codage de l'entrée. A chaque nouveau passage de la boucle, on diminue de 2 la taille de la partie codante de la bande à explorer. Le dernier passage dans la boucle (il reste un seul | dans chaque bloc, avec un B entre ces 2 | : 8 pas de calcul, ce qui correspond à : 2 × (n- (n-2)) + 2.

Soit n= 2k+1 (k est la taille du codage de x, et du codage de y). On passe k fois dans la boucle.

D'où, le nombre total de pas de calculs est en fonction de n :

(2 × n + 2 ) + (2 × (n-2) + 2 )+ ...+ (2 × (n–(n-2)) + 2 )).

La complexité dans le cas le pire en fonction de k est donnée par :

$C_{pire}(k)$= 2 × T(k) + 2 × k, où T(k) = $\sum_{i=1}^{k}(2i+1)$

Soit T'(k)= $\sum_{i=1}^{k}(2i) = 2 \times \sum_{i=1}^{k} i = k \times (k+1)$

T(k) + T'(k) = ($\sum_{j=1}^{2k+1} j$) -1 = $(k+1) \times (2k+1)$ -1

IMPORTANT : il faut savoir par coeur que $\sum_{i=1}^{n} i = \frac{n \times (n+1)}{2}$

QUESTION : le démontrer (par récurrence sur n).

Donc :

T(k) = $(k+1) \times (2k+1)$ -1 - $k \times (k+1) = (k+1)^2 - 1$

et

$C_{pire}(k)$= 2 × $((k+1)^2 - 1)$+ 2 × k

En remplacant k par $\frac{(n-1)}{2}$, on obtient la complexité dans le cas le pire en fonction de n par la fonction :

$C_{pire}(n)$= $\frac{(n-1)^2}{2} + 3(n-1)$

Conclusion : si C est la fonction qui exprime la complexité de $Z$ en fonction de la taille $n$ du codage de l'entrée, on a l'encadrement suivant :

$$n + 4 \leq C(n) \leq \frac{(n-1)^2}{2} + 3(n-1)$$

◄

5. Construire une MT qui calcule le maximum de 3 arguments. Donner sa complexité.
   ► Principe :

   (a) on installe une marque "d"dans la case juste avant la première case de la partie codante et une marque "*" juste après la dernière case de la partie codante, en supprimant les premiers | de chacun des trois blocs codant x, y et z, et on revient au début (repéré par "d"),

   (b) si et tant qu'on trouve un | entre "d" et "*", on efface chaque | qu'on trouve immédiatement après un B, et on ajoute un | après "*" (à la fin du bloc de | déjà ajoutés).

   Quand on sort de la boucle, le nombre de | qui ont été ajoutés après "*" est bien égal au nombre de | dans le plus grand bloc x, y ou z, et il n'y a pas d'autre | restants entre les marques "d" et "*" : CQFD.

   **Spécification plus précise de l'étape (a) :**

   - on recule d'une case, on écrit "d" pour indiquer le début de la partie codante
   - tant que la case courante ne contient pas un B :
     - on avance d'une case (elle contient forcément |)
     - on efface | et on avance d'une case
     - si et tant qu'on lit |, on avance d'une case
     - on avance d'une case
   - on recule d'une case, on on écrit "*" pour indiquer la fin de la partie codante
   - tant qu'on ne lit pas "d " :
     on recule d'une case

   **Spécification plus précise de l'étape (b) (que l'on commence avec la tete de lecture positionnée sur la case "d") :**

   - tant qu'on ne lit pas "*" :
     - on avance d'une case
     - tant qu'on ne lit pas |, on avance d'une case
     - si on lit "*", passer dans l'état $q_{stop}$ (pour indiquer qu'il ne reste aucune | dans la partie codante)
     - sinon :
       - ecrire B (on supprime le premier | restant dans les blocs encore non vides), et avancer d'une case
       - si et tant qu'on lit |, on avance d'une case
   - si on n'est pas dans l'état $q_{stop}$ (il faut aller écrire | à la fin du bloc de | après "*", puis revenir à la case $d$) : - on avance d'une case
     - si et tant qu'on lit | on avance d'une case
     - on écrit |
     - tant qu'on ne lit pas "d " on recule d'une case

**Complexité $C(n)$ :** on veut estimer (dans le cas le pire) le nombre de pas de calculs effectués en fonction de la taille $n$ du codage de l'entrée (i.e., le nombre de cases entre "d" et "*").

- Cout de Etape (a) : 2 +[ (n+2) + k ] (1ere boucle tant que avec k= le nombre de blocs de |, 3 pour la question) + 2 +[ (n+1)] (la boucle tant que pour revenir à "d"

Soit : 2n + 7 + k (2n + 10 pour k=3)

- Cout de Etape (b) :

On passe $x_i + 1$ fois dans la boucle, où $x_i$ est le maximum des nombres $x_1, \ldots, x_k$.

Au $i^e$ passage dans la boucle (pour $i < x_i+1$) : on fait (n+1) déplacements à droite et $k$ écritures de B, autant de déplacements que de | déjà écrits après le "*" (soit $i-1$), plus l'écriture d'un |, et un nombre de déplacements à gauche égal à i + n + 1 (pour revenir à la case "d")

Au dernier passage, on fait juste (n+1) déplacements vers la droite.

Le cout de l'étape (b) est donc :

$(\Sigma_{i=1}^{x_i}(n+1+k+i+i+n+1)) + (n+1)$

$= x_i \times (2n+2+k) + x_i \times (x_i+1)$

Comme $x_i < n$, si $k$ est une constante ($k = 3$ dans cet exercice, on obtient que le cout de cette étape est inférieur à : $n \times (2n+2+k) + n \times (n+1)$

Donc : $C(n) < (2n+7+k) + n \times (2n+2+k) + n \times (n+1)$ (on dira en ordre de grandeur inférieur à $n^2$).

◄

6. Généraliser pour construire une MT qui calcule, pour tout $n$, le maximum de $n$ arguments.
   ► Exactement le même principe de la question précédente : la même machine s'applique. ◄
7. Construire une MT $Z$ qui calcule $3^{(0)}$.
   ► On part en $q_0B$, et il faut écrire 3 | sur la bande et s'arrêter. D'où :

$$Z = \begin{bmatrix} q_0 & B & | & q_0 \\ q_0 & | & R & q_1 \\ q_1 & B & | & q_1 \\ q_1 & | & R & q_2 \\ q_2 & B & | & q_2 \end{bmatrix}$$

   ◄
8. Construire une MT $Z$ qui calcule $3^{(n)}$, pour toutes les arités $n$.
   ► Il suffit de tout effacer, puis d'écrire 3 | sur la bande. On n'aura qu'à composer la MT précédente (calculant $3^{(0)}$) avec celle qui efface tout, en décalant les numéros des états. D'où :

$Z =$

| | | | | |
|---|---|---|---|---|
| $q_0$ | \| | $R$ | $q_1$ | $q_0\vec{x} \vdash^* Bq_0B$ |
| $q_1$ | $B$ | $R$ | $q_0$ | $\vdash^* Bq_2B$ |
| $q_0$ | $B$ | $B$ | $q_2$ | Fin de l'effacement de la bande initiale |
| $q_2$ | $B$ | \| | $q_2$ | |
| $q_2$ | \| | $R$ | $q_3$ | Écriture de \|\|\| |
| $q_3$ | $B$ | \| | $q_3$ | |
| $q_3$ | \| | $R$ | $q_4$ | |
| $q_4$ | $B$ | 1 | $q_4$ | $\vdash^* \|\|q_4\|$ |

   ◄
9. Construire une MT $Z$ qui calcule $n^{(n)}$, pour toutes les arités $n$. [Attention, ce n'est pas $\lambda n[n^n]$.]
   ► $Z$ parcourt sa bande en allant vers la droite et en effaçant tous les | de chaque bloc de | sauf le premier. ◄
10. Construire une MT $Z$ qui calcule la fonction prédécesseur $Pred^{(1)}$ définie par :
    $Pred(x) = \begin{cases} x-1 & \texttt{si } x > 0 \\ 0 & \texttt{sinon} \end{cases}$
    ► $Z$ efface le premier | s'il existe. $Z = \begin{bmatrix} q_0 & | & B & q_0 \end{bmatrix}$ suffit. ◄

11. Construire une MT $Z$ qui calcule la division *entière* par 3, c'est-à-dire que :
$\Psi_Z^{(1)} = \lambda x.\lfloor \frac{x}{3} \rfloor$. Justifier la correction et la terminaison.
► Il suffit d'effacer le premier |, puis d'aller vers la droite en effaçant les | de numéro non multiple de 3, i.e. le premier et, s'il existe, le second de chaque groupe |||. D'où :

$$Z = \begin{array}{|cccc|l|} \hline q_0 & | & B & q_0 & q_0\overline{\vec{x}} \vdash^* Bq_0B \\ q_0 & B & R & q_1 & \vdash^* Bq_1|^xB \\ q_1 & | & B & q_1 & \\ q_1 & B & R & q_2 & \\ q_2 & | & B & q_2 & \\ q_2 & B & R & q_3 & \\ q_3 & | & R & q_1 & \vdash^* (BB|)^{x \texttt{ div } 3}B^{x \texttt{ mod } 3}q_3B \\ \hline \end{array}$$

$Z$ termine nécessairement en $q_3$. ◄

12. En modifiant le plus simplement possible la machine précédente, construire une machine $Z'$ qui calcule la division *exacte* par 3 de la somme de deux arguments, c'est-à-dire que : $\Psi_{Z'}^{(2)} = \lambda xy.\frac{x+y}{3}$, ou encore : $\Psi_{Z'}^{(2)}(x,y) = \begin{cases} p & \texttt{si } x+y = 3p \\ \uparrow & \texttt{sinon} \end{cases}$
► === à faire === ◄

13. Soit la MT Z définie par ses quadruplets :

$$Z = \left\{ \begin{bmatrix} q_0 & B & R & q_0 \\ q_0 & | & B & q_1 \\ q_1 & B & R & q_2 \\ q_2 & | & B & q_1 \end{bmatrix} \begin{bmatrix} q_2 & B & R & q_4 \\ q_3 & B & R & q_4 \\ q_4 & | & B & q_3 \\ q_4 & B & R & q_6 \end{bmatrix} \begin{bmatrix} q_5 & B & R & q_6 \\ q_6 & | & R & q_6 \\ q_6 & B & R & q_7 \\ q_7 & | & B & q_6 \end{bmatrix} \begin{bmatrix} q_7 & B & R & q_8 \\ q_8 & | & | & q_7 \\ & & & \end{bmatrix} \right\}$$

Quelles sont les fonctions $\Psi_Z^{(n)}$ calculées par $Z$, pour toutes les arités $n$ ?

► === à faire === ◄

---

*Machines de Turing transformant des chaînes de caractères*

---

Il ne faut pas oublier que les MT les plus "naturelles" transforment simplement des chaînes de caractères. Voici un petit problème illustrant cela.

1. Construire une MT $Z$ à deux bandes qui transforme une chaîne $w$ sur $V = \{a, b, c\}$ en son miroir $\widetilde{w}$. Bien sûr, on suppose que le blanc $S_0$, usuellement noté $B$, n'est pas dans $V$. La configuration initiale sera $c_0 = (B, B)q_0(wB, q_0B)$, ou, si on préfère une notation "verticale", $c_0 = \begin{pmatrix} B \\ B \end{pmatrix} q_0 \begin{pmatrix} wB \\ B \end{pmatrix}$, et la configuration finale contiendra $\widetilde{w}$ sur la deuxième bande, et rien sur la première (seulement des blancs). On ne demande rien de particulier sur la position finale des deux têtes de lecture-écriture.
[Ne pas oublier qu'on peut utiliser autant de symboles qu'on veut, et autant d'états qu'on veut.]
► === à faire === ◄

2. Même question, mais avec une MT $Z'$ à une seule bande. On essaiera de trouver deux méthodes radicalement différentes.
► === à faire === ◄

---

*Compléments de cours sous forme d'exercices libres*

---

**Encadrement de la partie active de la bande d'une MT**

Soit $Z$ une MT quelconque, $r(Z)$ le plus grand indice d'un de ses états et $s(Z)$ le plus grand indice d'un de ses caractères. Donc $E(Z) \subseteq \{q_0, \dots q_{r(Z)}\}$ et $A(Z) \subseteq \{S_0 = B, S_1 = |, \dots S_{s(Z)}\}$.
On appelle $\lambda(Z)$ et $\delta(Z)$ les deux prochains caractères non utlisés par $Z$, i.e. $\lambda(Z) = S_{s(Z)+1}$ et $\delta(Z) = S_{s(Z)+2}$. Dans la suite, on abrège en notant simplement $\lambda$ et $\delta$.
Détailler une méthode générique pour transformer $Z$ en une MT équivalente $Z'$ qui

— commence par encadrer la partie active de la bande initiale par $\lambda$ et $\delta$.
— simule le calcul de $Z$ pas à pas, en poussant $\lambda$ d'une case vers la gauche et $\delta$ d'une case vers la droite si nécessaire.
— si et quand le calcul de $Z$ s'arrête, tasse le | juste à droite de $\lambda$, en restant à gauche de $\delta$, et en "effaçant" (transformant en $B$) tous les caractères différents de |.
— efface le $\delta$, revient à gauche, efface le $\lambda$, et s'arrête dans la configuration finale $q_{r(Z')}\ |^{<\alpha>}B$, si $Z$ s'arrête en $\alpha$ sur cette même entrée. [$< \alpha >$ est le nombre de | de la description instantanée $\alpha$.]

**Variantes de machines de Turing**

On obtient des variantes des MT de multiples façons.

1. Si on permet d'écrire puis de mouvoir la tête de lecture-écriture en un pas de calcul, on obtient des *MT à quintuplets*. Un quintuplet "simple" est de forme
   $(q_i \quad S_j \quad S_k \quad q_l \quad L \mid R \mid N)$, et un quintuplet "à oracle" est de forme
   $(q_i \quad S_j \quad q_k \quad q_l \quad L \mid R \mid N)$. ($N$ pour "no move".)

   Montrer qu'on peut simuler toute MT à quintuplets (simple ou à oracle) par une MT à quadruplets.

2. On peut permettre plusieurs bandes, avec une tête de lecture-écriture par bande.

   — Définir ce modèle pour $k+1$ bandes, $k$ étant un entier quelconque.
   — Montrer qu'on peut simuler une MT à 2 bandes par une MT à 1 bande.
   — Essayer d'évaluer le coût de cette simulation sans regarder le cours (le livre de Hopcroft & Ullman "Formal Languages and their Relation to Automata", Addison-Wesley, 1969, donne un certain nombre d'évaluations de ce genre).

3. On peut permettre plusieurs têtes de lecture-écriture par bande, et aussi remplacer les bandes par des structures plus compliquées, comme des matrices (non bornées) de cellules à 2 dimensions ou plus ("plan" ou "(hyper)volume" de travail), avec des mouvements élémentaires adaptés (N, NE, E, SE, S, SO, O, NO par exemple, en suivant la rose des vents dans un "plan de travail" à 2 dimensions).

   Réfléchir à la façon de simuler de telles MT augmentées par les MT précédentes.

4. Enfin, on peut essayer de réduire la puissance en définissant des MT à 1/2 bandes. Définir ce modèle et montrer qu'il est aussi puissant que celui des MT à bandes doublement infinies.

5. Définir un automate à pile déterministe [1] comme une MT à deux demi-bandes, la première correspondant à un flot d'entrée (en lecture seulement, tête allant dans un seul sens, arrêt forcé en fin d'entrée), et la seconde soumise aux restrictions usuelles des piles. [2]

6. Montrer par contre qu'une MT à 2 piles est équivalente à une MT à une bande.

1. Pas de transition avec $\varepsilon$ en entrée, pas deux arcs quittant un état avec le même symbole d'entrée et la même chaîne lue en haut de pile (il est toujours possible d'éliminer les $\varepsilon$ d'un AP, mais pas toujours possible de le déterminiser).

2. Ce modèle est nécessairement moins puissant que celui des MT générales, puisque les AP ne peuvent calculer que des langages hors-contexte, tous décidables.