Simulation d'une machine de Turing à k bandes en temps $t(n)$ par une machine de Turing à deux bandes en temps $t(n)\log(t(n))$

Mehdi Bouaziz

Jeudi 20 Décembre 2007

Théorème 1. *Soit M une machine de Turing déterministe à k bandes calculant en temps $t(n)$. Il existe une machine de Turing M' déterministe à deux bandes équivalente à M calculant en temps $O(t(n)\log(t(n)))$.*

Démonstration. La bande principale de la machine M' correspond à un codage des k bandes de la machine M. Pour chaque bande j de la machine M, la bande principale de la machine M' comporte deux pistes (une *piste basse* et une *piste haute*) qui contiennent toutes les informations nécessaires pour simuler les calculs effectués sur la bande j. L'alphabet de M' est donc $\Sigma' = \Sigma^{2k}$ où Σ désigne l'alphabet de M. De plus, une cellule de la bande j figure de manière exclusive soit dans piste haute, soit dans la piste basse. Une cellule d'une piste est dite « vide »si elle ne correspond pas à une cellule de la bande j. La seconde bande est une bande de travail, qui servira à mettre à jour les données de la bande principale.

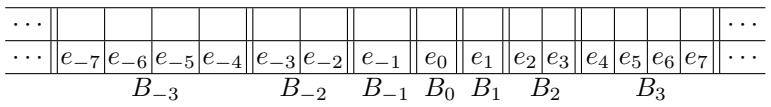
Pour simplifier, nous nous intéresserons à une certaine bande j de la machine M, les autres étant simulées de la même façon. La paire de pistes conrrespondant à cette bande est virtuellement découpée en blocs $(B_i)_{i\in\mathbb{Z}}$ dans l'ordre de $\mathbb{Z}$. Le bloc B_0 est de taille 1 et la taille des blocs augmente exponentiellement de part et d'autre du bloc B_0. Ainsi pour $i \neq 0$, le bloc B_i est de taille $2^{|i|-1}$. Pour simplifier, on appelle *piste basse* d'un bloc la partie de la piste basse située dans ce bloc ; on définit de même la piste haute d'un bloc.

Au lancement de la simulation, le mot d'entrée est placé sur la piste basse correspondant à la bande d'entrée. Pour simuler une transition de la machine M, au lieu de déplacer la tête de la bande, M' va transporter les données dans la direction opposée à la transition, de manière à ce que la piste basse du bloc B_0 contienne toujours, avant la simulation d'une transition de la machine M, le symbole lu par la tête j de la machine M. On dit qu'un bloc est :

- *vide* si ses deux pistes sont vides ;
- *à moitié plein* si sa piste haute est vide et si sa piste basse ne contient que des cellules de la bande j ;

– *plein* si les deux pistes ne contiennent que des cellules de la bande j.

Le dessin suivant montre les blocs correspondant à une bande de la machine M.



Après la simulation de chaque transition de la machine M, les invariants suivants sont conservés :

- un bloc est toujours vide, à moitié plein ou plein ;
- le bloc B_0 (position de la tête de lecture) est toujours à moitié plein ;
- pour $i \neq 0$, B_i est plein si et seulement si B_{-i} est vide (donc B_i est à moitié plein si et seulement si B_{-i} est à moitié plein) ;
- le contenu des blocs représente les cellules consécutives de la bande k de la machine M. Si B_i est un bloc plein, et si $i > 0$, alors les cellules de la piste haute sont à gauche des cellules de la piste basse ; si $i < 0$ alors les cellules de la piste haute sont à droite des cellules de la piste basse ;
- si $i < i'$, B_i représente des cellules à gauche de celles de $B_{i'}$;
- la piste haute de B_0 est marquée de manière à retrouver ce bloc lors des mises à jour.

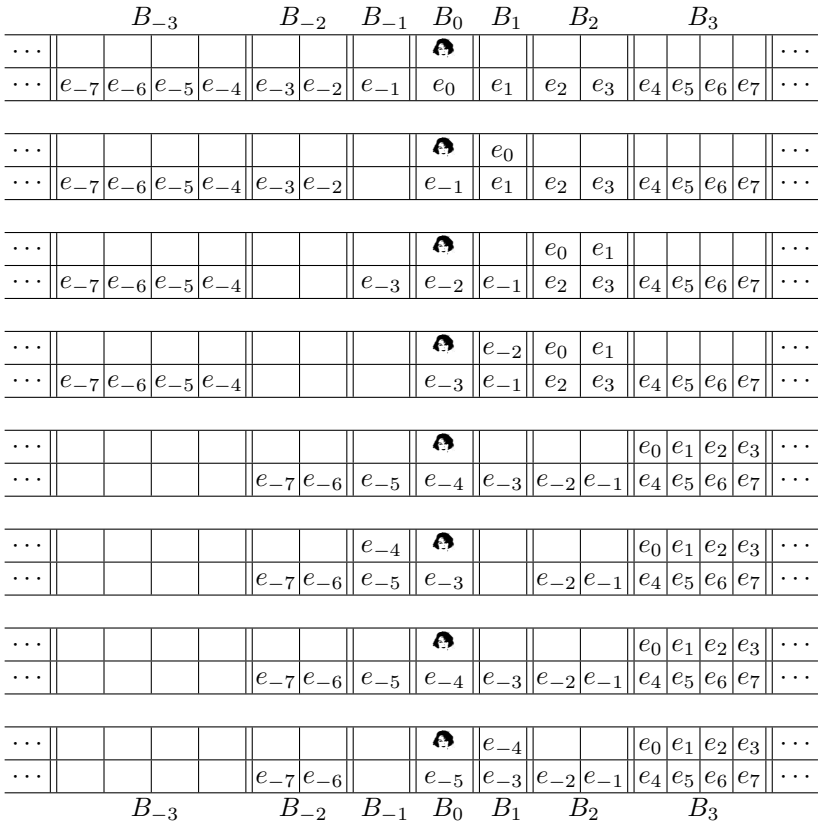
Comme la tête de lecture de la machine M' lit le bloc B_0 au début de la simulation d'une transition de la machine M, la machine M' peut écrire simultanément tous les symboles. Pour gérer les déplacements, elle procède de la manière suivante. Supposons par exemple que la machine M effectue un déplacement vers la gauche sur la bande j. Il faut donc décaler vers la droite une partie des données de la bande principale. La machine M' utilise l'algorithme suivant :

repeat {Tasse la bande à droite}
  Chercher le plus petit $i > 0$ tel que B_i n'est pas plein
  if B_i est à moitié plein **then**
    Déplacer le contenu de B_{i-1} dans la piste haute de B_i
  else {B_i est vide}
    Déplacer le contenu de B_{i-1} dans la piste basse de B_i
until $i = 1$

Chercher le plus grand $i < 0$ tel que B_i n'est pas vide
if B_i est à moitié plein **then**
  Déplacer le contenu de la piste basse de B_i dans les pistes basses des blocs $B_{i+1}, B_{i+2}, \cdots, B_{-1}, B_0$
else {B_i est plein}
  Déplacer le contenu de la piste haute de B_i dans les pistes basses des blocs $B_{i+1}, B_{i+2}, \cdots, B_{-1}, B_0$

On vérifie que cet algorithme respecte les invariants donnés précédemment.

La simulation d'un déplacement vers la droite se fait de façon symétrique. La figure suivante montre la simulation de quatre déplacements vers la gauche, un déplacement vers la droite, puis deux déplacements vers la gauche :



Pour effectuer les déplacements des cellules au sein d'un bloc, on utilise une pile sur la bande de travail de M'. Un tel déplacement nécessite au plus $c \times 2^{|i|}$ transitions pour le bloc B_i (c étant une constante). Donc, par somme, une mise à jour de la bande principale nécessite au plus $2c \times 2^{|i|}$ transitions, où i est l'indice du bloc le plus éloigné de B_0 qu'il faut modifier. Avec $i > 0$, après la modification du bloc B_i, tous les blocs $B_1, B_2, \cdots, B_{i-1}$ sont tous à moitié plein. B_i ne sera modifié à nouveau que lorsque ces blocs seront une nouvelle fois pleins. Or seule la simulation complète d'au moins $2^{i-1} - 1$ transitions de la machine M oblige à nouveau la modification du bloc B_i. Ainsi B_i n'est modifié au plus que toutes les $2^{|i|-1}$ simulations d'une transition de la machine M.

Comme il n'y a, pour une entrée de longueur n, au plus, que $t(n)$ telles transitions, seuls les blocs B_i avec $|i| \leq \lfloor \log_2(t(n)) \rfloor + 1$ sont éventuellement

utilisés dans la simulation. La simulation d'une bande de M demande donc

$$\sum_{i=1}^{\lfloor \log_2(t(n)) \rfloor + 1} \frac{t(n)}{2^{|i|-1}} \times 2c \times 2^{|i|}$$

transitions, au plus. La simulation de toutes les bandes étant identique et indépendante, le nombre de transitions nécessaires à la simulation de la machine M est donc majoré par

$$k \times \sum_{i=1}^{\lfloor \log_2(t(n)) \rfloor + 1} \frac{t(n)}{2^{|i|-1}} \times 2c \times 2^{|i|} = O(t(n) \log(t(n))).$$

□

Le théorème d'accélération fournit immédiatement le résultat suivant.

Corollaire 2. *Soit M une machine de Turing déterministe à k bandes calculant en temps $t(n)$. Il existe une machine de Turing M' déterministe à deux bandes équivalente à M calculant en temps $t(n)\log(t(n))$.*

Théorème 3 (Théorème de la hiérarchie temporelle)**.** *Soient $f : \mathbb{N} \to \mathbb{N}$ et $g : \mathbb{N} \to \mathbb{N}$ deux fonctions telles que $g\log(g) = o(f)$ et $f(n) \geq g(n) \geq n$. Si f est constructible en temps, l'inclusion $\textsc{Time}(g(n)) \subsetneq \textsc{Time}(f(n))$ est stricte.*

Ce résultat est moins fort que le théorème de hiérarchie du cours puisque le facteur $\log(g)$ n'apparaît pas dans ce dernier. Il reste important de noter que cette inclusion ne concerne que les classes de complexités des machines déterministes.

Démonstration. D'après le théorème 1, $\textsc{Time}(g(n)) \subseteq \textsc{Time}_2(g(n)\log(g(n)))$. Il suffit donc de montrer que $\textsc{Time}_2(g(n)\log(g(n))) \subsetneq \textsc{Time}(f(n))$.

On suppose fixé un codage des machines de Turing sur un alphabet $\Sigma \cup \{0,1\}$. À chaque codage c d'une machine de Turing est associée une fonction récursive $\phi_c : \mathbb{N} \to \{0,1\}$ ($\phi_c(e) = 1$ si la machine de code c accepte l'entrée e, et 0 sinon). La fonction $\phi_c(c)$ est récursive et il existe une machine de Turing M_0 à deux bandes qui calcule $\phi_c(c)$.

On a $g(n) \geq n$ et $\lim_{n\to+\infty} \frac{g(n)\log(g(n))}{f(n)} = 0$, donc $\lim_{n\to+\infty} \frac{f(n)}{n} = +\infty$. Quitte à ajouter à M_0 des bandes supplémentaire, on la modifie de sorte que sur l'entrée c :

- la machine accepte si le temps $f(|c|)$ est atteint ou si, lors de la simulation, la machine de code c s'arrête et rejette ;
- la machine rejette si, lors de la simulation, la machine de code c s'arrête et accepte.

On appelle M_1 cette machine de Turing qui s'arrête pour l'entrée c en temps $f(|c|)$.

Soit L le langage accepté par M_1 (un entier étant identifié par sa représentation binaire), on a donc $L \in \textsc{Time}(f(n))$.

Par un raisonnement par l'absurde, montrons que $L \notin \textsc{Time}(g(n))$. D'après le théorème 1, si $L \in \textsc{Time}(g(n))$, il existe une machine de Turing M à deux bandes qui accepte L en temps $g(n)\log(g(n))$. Il en existe en fait une infinité car on peut toujours lui ajouter des transitions inaccessibles. Il y a donc une infinité de codes pour une machine de Turing qui accepte L *avec le même alphabet* Σ que M. Soit c un code d'une telle machine. Lors du calcul par la machine M_1 sur l'entrée c, le nombre de transitions nécessaires pour simuler la machine M est donc majoré par $mg(|c|)\log(g(|c|))$ où m est une constante dépendant de la constante $|\Sigma|$. Pour ne pas rencontrer de problème de place lors du calcul, il faut donc avoir :

$$mg(|c|)\log(g(|c|)) \leq f(|c|).$$

Comme $\lim_{n\to+\infty} \frac{g(n)\log(g(n))}{f(n)} = 0$, on peut choisir c suffisamment grand pour que cette condition soit remplie. Ainsi, les cas d'acceptation de M sont tous pris en compte. Maintenant $c \in L$ si et seulement si la simulation de la machine de code c conduit cette dernière à un calcul acceptant en temps $mg(|c|)\log(g(|c|)) \leq f(|c|)$ donc si et seulement si la machine M rejette l'entrée et donc si et seulement si $c \notin L$. Ainsi L ne peut appartenir à $\textsc{Time}(g(n))$.

□

Exemple 4. Comme pour tous $k, k' > 1$, on a $\lim_{x\to\infty} \frac{n^k \log(n)}{2^{k'n}} = 0$, le théorème de hiérarchie temporelle montre que $\textsc{Time}(n^k) \subsetneq \textsc{Time}(2^{k'n})$. Ainsi, $\textsc{P} \subsetneq \textsc{Exp}$.