Simulation d'une Machine de Turing a k bandes en $T(n)$ par une Machine de Turing a deux bandes en $O(T(n)\log(T(n)))$

Quentin de Mourgues

5 février 2011

Résumé

Nous avons vu en cours que l'on pouvait simuler une machine de Turing à k bandes par une machine à une seule bande. Cette transformation fait cependant apparaître un facteur carré dans le temps d'exécution de la machine et celui-ci ne peut être réduit : il suffit de considérer le langage des palindromes $L = \{wa\overline{w} \mid w \in A^*,\ a \in A \cup \{\varepsilon\}\}$, une machine a une bande le reconnait nécessairement en temps $O(n^2)$ tandis qu'il suffit d'une machine a deux bandes pour que cela se fasse en temps linéraire. Le présent article démontre que l'utilisation d'une machine de Turing à deux bandes permet de s'affranchir de ce facteur carré en le remplaçant par un facteur $n \log n$. Nous étudierons aussi une application de ce résultat aux théorèmes de hiérarchie en temps.

1 Simulation d'une MT à k bandes par une MT à deux bandes

Théorème 1. *Soit M_1 une machine de Turing à k bandes reconnaissant le langage L en temps $T(n)$, Alors il existe une machine de Turing à deux bandes M_2 qui reconnait L en $O(T(n)\log T(n))$.*

Démonstration. On suppose tout d'abord que les k bandes de M_1 ont toutes le même alphabet de bande Γ (cela permet d'éviter des notations trop lourdes). On construit alors M_2 telle que l'alphabet de la bande 1 soit $(\Gamma^2)^k$ et que l'alphabet de la bande 2 soit Γ. La bande 2 ne nous servira donc qu'à stocker des données se trouvant sur la bande 1 et à les recopier sur celle-ci, tandis que la bande 1 simulera les k bandes de M_1. L'idée principale pour obtenir un facteur $\log T(n)$ est de pouvoir simuler une étape de calcul de M_1 par M_2 en temps amorti $O(\log T(n))$, pour cela on distingue deux étapes :

- le choix de la transition (qui s'effectuera en temps constant) ;
- la mise à jour de la bande 1 dans le nouvel état atteint par M_1 (en temps amorti $O(\log T(n))$).

Pour pouvoir choisir en temps constant quelle transition effectuer dans M_2, il faut que toutes les têtes de lecture dans M_1 se retrouvent au même endroit sur la bande 1 de M_2 que l'on nommera B_0 (ceci est possible car l'alphabet de la bande 1 est $(\Gamma^2)^k$: la bande 1 a k étages qui représentent les k bandes de M_1 et chacun de ces k étages a deux niveaux).

Exemple 2. On considère M_1 à 4 bandes telle que les têtes de lectures se trouvent sur les lettres a_0, b_0, c_0 et d_0, les lettres à gauche de chaque tête sont notées négativement et celles à droite positivement. On crée M_2 en alignant a_0, b_0, c_0 et d_0 dans le même compartiment B_0 où se trouve la tête de lecture :

									↓	
Bande 4	d_{-8}	d_{-7}	d_{-6}	d_{-5}	d_{-4}	d_{-3}	d_{-2}	d_{-1}	d_0	d_1
				↓						
Bande 3	c_{-3}	c_{-2}	c_{-1}	c_0	c_1	c_2	c_3	c_4	c_5	c_6
		↓								
Bande 2	b_{-1}	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
						↓				
Bande 1	a_{-5}	a_{-4}	a_{-3}	a_{-2}	a_{-1}	a_0	a_1	a_2	a_3	a_4

FIG. 1 – M_1 Machine de Turing à 4 bandes.

										↓								
Bande 2																		
										↓								
Bande 1	E4																	
		d_{-8}	d_{-7}	d_{-6}	d_{-5}	d_{-4}	d_{-3}	d_{-2}	d_{-1}	d_0	d_1							
	E3																	
							c_{-3}	c_{-2}	c_{-1}	c_0	c_1	c_2	c_3	c_4	c_5	c_6		
	E2																	
									b_{-1}	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
	E1																	
					a_{-5}	a_{-4}	a_{-3}	a_{-2}	a_{-1}	a_0	a_1	a_2	a_3	a_4				
										$\overbrace{\quad}^{B_0}$								

FIG. 2 – M_2 Machine de Turing à 2 bandes simulant M_1, il suffit que la tête de bande soit placée en B_0 pour décider de la transition à effectuer.

On doit maintenant réussir à préserver cette propriété : "les têtes de lecture sont alignés en B_0 dans la bande 1 de M_2" lorsqu'on effectue une transition et ce en temps amorti $O(\log(T(n))$. Pour cela, nous allons nous concentrer sur un

seul étage de la bande 1 (il suffira alors de répéter k fois la procédure, une fois pour chaque étage, pour obtenir le contenu de bande suivant de M_1 dans M_2).

On divise la bande 1 en une infinité de blocs ..., B_{-k}, ..., B_{-1}, B_0, B_1, ..., B_k, ... de taille croissante exponentielle : B_0 a pour taille 1 et $\forall k \in \mathbb{Z}^*$ B_k a pour taille $2^{|k|-1}$.

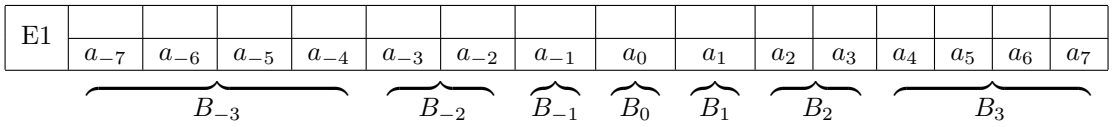

FIG. 3 – Les blocs sur la bande 1 de M_2

On va considérer l'invariant suivant :

1. $\forall i > 0$
   (a) B_i a ses deux niveaux remplis et B_{-i} est vide.
   (b) B_{-i} a ses deux niveaux remplis et B_i est vide.
   (c) B_{-i} et B_i ont leur premier niveau rempli et leur second niveau vide.
2. (a) $\forall i > 0$ le second niveau de B_i contient les symboles à gauche de ceux du premier niveau.
   (b) $\forall i < 0$ le second niveau de B_i contient les symboles à droite de ceux du premier niveau.
3. $\forall i < j$ B_i contient des symboles à gauche de ceux contenus par B_j.
4. B_0 a un symbole spécial sur son second niveau .

Au départ, la bande 1 vérifie bien les propriétés 1.(a), 2, 3 et 4.
Supposons qu'à l'étape j, l'invariant soit vérifié alors et que la tête de lecture de la bande 1 de M_1 avance vers la gauche (le cas vers la droite est symétrique). Alors :

- On déplace la tête de lecture de la bande 1 (qui se trouve en B_0) vers la droite jusqu'à trouver le premier bloc non plein B_i. On recopie alors grace à la bande 2 de M_2, les données contenues dans $B_0 \dots B_{i-1}$ dans les premiers niveaux de $B_1 \dots B_{i-1}$ ainsi que le premier niveau niveau de B_i si celui-ci est vide sinon dans son second niveau. Ce transfert de données se fait en respectant les propriétés 2 et 3 et prend un temps proportionel à la taille de B_i.
- On retourne en B_0 dont le premier niveau est maintenant vide et on déplace la tête de la bande 1 vers la gauche jusqu'à trouver B_{-i} (c'est le premier bloc non vide d'après la propriété 1).
  - Si B_i est plein alors B_{-i} a son premier niveau seulement plein et on doit le vider. Pour cela on utilise la bande 2 pour le recopier sur les premier niveaux de $B_{-i+1} \dots B_0$ en respectant les propriétés 2 3. La propriété 1 est alors vérifiée : $\forall\ 0 < j < i$ B_j et B_{-j} sont à moitié plein B_i est plein et B_{-i} est vide

– Si B_i est a moitié plein alors B_{-i} est plein, on recopie son second niveau sur son premier niveau et son premier niveau sur $B_{-i+1} \dots B_0$ respectant les propriétés 2 3. La propriété 1 est alors vérifiée : $\forall\ 0 < j \leq i$ B_j et B_{-j} sont à moitié plein

Comme auparavant ces opérations s'effectuent en temps proportionel à la taille de B_i.

Une telle opération sera appellée une B_i-opération.

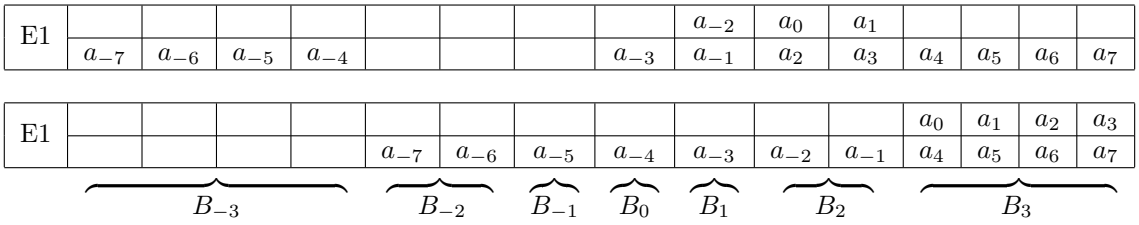

FIG. 4 – Une B_3-opération sur le premier étage de la bande 1 lorsque la tête de lecture de la bande 1 de M_1 se déplace vers la gauche.

Nous allons maintenant démontrer que le temps amorti de la mise à jour de la bande est en $O(\log(T(n)))$.

Remarquons tout d'abord qu'après une B_i-opération, $\forall j < i$ B_j est a moitié plein et qu'un bloc B_j quelconque ne peut être plein qu'après une B_j-opération. Comme on effectue une B_i-opération si et seulement si $\forall j < i$ B_j est plein, on en déduit qu'une fois une B_i-opération effectuée et avant d'en effectuer une autre, on doit effectuer une B_{i-1}-opération pour remplir le bloc B_{i-1} puis une B_{i-2}-opération pour remplir le bloc B_{i-2} etc ... jusqu'à une B_1-opération pour remplir le bloc B_1

Posons T_i le nombre de déplacement de déplacement de la tête de lecture de la bande 1 de M_1 entre deux B_i-opérations, on a d'après la remarque précédente :

$$T_1 \geq 2 \quad \text{et} \quad \forall i > 1 \quad T_i \geq \sum_{i=1}^{k-1} T_i,$$

d'où :

$$T_k \geq 2\sum_{i=1}^{k-2} T_i \geq 4\sum_{i=1}^{k-3} T_i \geq \dots \geq 2^{k-2} T_1 \geq 2^{k-1}.$$

On en déduit que M_2 fera des B_i-opération pour les i tels que $i \leq \log_2 T(n) + 1$

D'où si M_1 fait $T(n)$ déplacements, M_2 fait au plus :

$$T_2(n) = \sum_{i=1}^{\log_2 T(n)+1} \frac{m 2^i T(n)}{2^{i-1}} \leq 4m\, T(n)\, log_2\, T(n).$$

Il reste finalement à effectuer ces opérations pour chacun des k étages de la bande 1 ce qui introduit un facteur k dans le resultat :

$$T_2(n) \leq 4m\,k\,T(n)\,log_2\,T(n).$$

□

2 Application au théorème de hiérarchie en temps

Définition 3. On dit qu'une fonction $f : \mathbb{N} \to \mathbb{N}$ est constructible en temps s'il existe une machine de Turing M majorée en temps par f telle que $\forall n \in \mathbb{N}\ \exists w \in A^*$ tel que l'execution de M sur w se fasse en exactement $f(n)$ déplacements.

Théorème 4. *Soit T_2 une fonction constructible en temps et T_1 telle que*

$$inf_{n\to\infty} \frac{T_1(n)\log T_1(n)}{T_2(n)} = 0.$$

Alors il existe un langague L dans $DTIME(T_2(n))$ qui n'est pas dans $DTIME(T_1(n))$ ie $DTIME(T_1(n)) \subsetneq DTIME(T_2(n))$.

Démonstration. On construit une machine M_2 telle que $\forall w \in A^*$ M_2 s'arrête après $T_2(|w|)$ déplacements et telle que pour toute machine M acceptant le langage L en $T_1(n)$ $\exists w \in L$ / M_2 n'accepte pas w. Il résulte alors par argument diagonal que $L(M_2)$ est différent de tous les langages qui sont dans $DTIME(T_1(n))$ ie $L(M_2) \in DTIME(T_2(n)) \setminus DTIME(T_1(n))$.

Soit M_1 une machine à k bandes s'exécutant exactement en $T_2(n)$ (elle existe car T_2 est constructible en temps par hypothèse) et M_w la machine dont le codage est w $\forall w \in A^*$ et telle qu'elle s'execute en $T_1(n)$. Comme M_w a un nombre de bandes quelconque, on va la simuler sur M_2 en utilisant seulement 2 bandes, d'où un temps d'execution de M_w sur M_2 en $cT_1(n)\log T_1(n)$ où c est une constante dépendant de M_w (en effet M_w peut avoir des alphabets de bandes différents de T_2 ce qui augmente la constante).
M_2 possède donc 2 bandes pour simuler M_w et k bandes pour simuler M_1.

M_2 :
Entrée w
Simuler M_w et M_1 sur w
  Tant que M_1 ne s'arrete pas
    Si M_w accepte w **Alors** rejeter w
    Sinon Si M_w rejete w **Alors** accepter w
  fin Tant que
Si w n'a pas déjà été accepté **Alors** rejeter w

Il reste à montrer que pour toute Machine M acceptant un langage $L \in DTIME(T_1(n))$ il existe un mot w tel que M_2 le refuse, *i.e.* il faut trouver un mot w tel que

$$c_{M_w} T_1(|w|)\log T_1(|w|) \leq T_2(|w|).$$

Ceci est possible car on peut créer une machine M' telle que son codage soit arbitrairement grand en rajoutant des états inutiles, par exemple, ce qui ne change pas la constante et reconnaissant L. □

Exemple 5. Comme $T(n) = 2^n$ est constructible en temps et que

$$\forall k \quad inf_{n\to\infty} \frac{kn^k \log n}{2^n} = 0.$$

On a $PTIME \subsetneq EXPTIME \subsetneq EXP2TIME \subsetneq EXP3TIME \ldots$

Références

[1] Jeffrey D. ULLMAN John E. HOPCROFT. *Introduction to automata theory, languages, and computation.* ADDISON-WESLEY, 1979.