Simulation d'une machine de Turing à k bandes de temps $T(n)$ avec une machine de Turing à 2 bandes en temps $O(T(n) \log_2 T(n))$

Cadé David

Table des matières

Introduction

On commence par voir comment simuler une machine à k bandes de temps $T(n)$ avec une machine de Turing à 2 bandes en temps $T(n)\log_2 T(n)$. Ceci permettra de démontrer le théorème d'hiérarchie temporelle qui permettra de montrer que $\mathbf{EXP} \neq \mathbf{P}$.

1 Simulation d'une machine de Turing à k bandes

Nous allons nous intéresser à une machine de Turing à k bandes bidirectionnelles M_1, que nous voulons simuler dans une machine de Turing M_2 ayant 2 bandes.

1.1 Architecture

Soit A l'alphabet de bande de la machine de Turing M_1. L'alphabet de bande de la machine de Turing M_2 est $\hat{A}^{2k}$, avec $\hat{A}$ l'alphabet A dans lequel on a rajouté plusieurs symboles, dont un symbole de vide, un symbole séparant les blocs, et un marqueur pour le bloc central.

La deuxième bande sert uniquement à faire des transferts de données.

Définition 1. *La* piste i *est la projection des données de la première bande sur la i-ème composante.*

Définition 2.
- *On appelle* piste haute *de la i-ème bande la piste $2i$.*
- *On appelle* piste basse *de la i-ème bande la piste $2i+1$.*

Au début, toutes les pistes hautes sont vides, et les pistes basses contiennent les données initiales $(a_j^i)_{0\leq i\leq k-1}$.

Ces deux pistes sont découpées en blocs $(B_j^i)_{j\in\mathbf{Z}}$ de taille $2^{|i|-1}$ pour $i\neq 0$. Le bloc B_0^i joue un rôle spécial et est de taille 1. Les blocs sont arrangés comme dans la figure 1.

Remarque 1. *Les blocs sont séparés par un symbole de $\hat{A}$, l'alphabet de bande de la machine. Ils ne sont pas représentés dans la figure 1.*

1.2 B_i-opération

1.2.1 Description

On va se concentrer sur une bande de M_1, et ses pistes associées dans M_2.

Sur les pistes hautes et basses de la bande i, on va faire en sorte que les éléments à gauche de B_0 sont les éléments précédant le marqueur de position de la bande i, les éléments à droite ceux suivant le marqueur, et l'élément de la bande B_0 l'élément qui est sur le marqueur.

Définition 3. *Une piste d'un bloc est* plein *si tous les éléments de la piste du bloc sont dans A.*

Une piste d'un bloc est vide *si tous les éléments de la piste du bloc sont vides.*

Définition 4. *L'*état *d'un bloc est soit :*

	B_{-2}		B_{-1}	B_0	B_1	B_2			
...								...	Piste haute de la bande 0
...	a^0_{-3}	a^0_{-2}	a^0_{-1}	a^0_0	a^0_1	a^0_2	a^0_3	...	Piste basse de la bande 0
...								...	Piste haute de la bande 1
...	a^1_{-3}	a^1_{-2}	a^1_{-1}	a^1_0	a^1_1	a^1_2	a^1_3	...	Piste basse de la bande 1

$\vdots$

FIG. 1 – Description de la structure utilisée dans la première bande

– *Plein, si les deux pistes du bloc sont pleines.*
– *Vide, si les deux pistes du bloc sont vides.*
– *À moitié plein, si la piste du bas est pleine, et la piste du haut est vide.*

Voici les invariants que l'on garde à la fin de chaque simulation de déplacement de tête de lecture :

– Pour tout $i > 0$, Soit B_i et B_{-i} sont à moitié pleins, soit B_i est vide et B_{-i} est plein, soit B_i est plein et B_{-i} est vide.
– Le contenu d'un bloc est un ensemble d'éléments de a_i consécutifs. Si $i > 0$, les éléments de la piste haute de B_i précèdent ceux de la piste basse de B_i, et inversement si $i < 0$. Si $i < j$, les éléments de B_i précèdent ceux de B_j.
– Le bloc B_0 est à moitié plein, et sa piste haute est marquée pour la retrouver facilement.

Supposons que la tête de lecture aille à gauche dans la bande de M_1 que l'on regarde. Nous allons déplacer des données à droite dans M_2 pour qu'on aie en B_0 la donnée sur la tête de lecture.

Pour cela, on va trouver le premier $i > 0$ tel que B_i ne soit pas plein, en stockant par la même occasion les données de $B_0, \dots, B_{i-1}$ dans la deuxième bande, et vidant $B_0, \dots, B_{i-1}$.

On a là $1 + 2\sum_{j=1}^{i-1} 2^{j-1} = 2^i - 1$ données dans la deuxième bande. On met les $2^{i-1} - 1$ premières données dans les pistes basses de $B_1, \dots, B_{i-1}$, puis, si B_i est à moitié plein,on met les 2^{i-1} données restantes dans la piste haute de B_i, sinon on les met dans la piste basse.

Vu que les blocs $B_1, \dots, B_{i-1}$ étaient pleins, les blocs $B_{-(i-1)}, \dots, B_{-1}$ sont vides.

Ensuite on trouve B_{-i}, en mesurant la distance de B_i à B_0 avec la deuxième bande. Si B_{-i} est plein, on vide la piste du haut de B_{-i} en la stockant dans la deuxième bande de M_2. Si B_{-i} est à moitié plein, on vide la piste du bas de B_{-i} en la stockant dans la deuxième bande de M_2.

Ensuite, on transfère les données sauvegardées dans la deuxième bande dans les pistes du bas de $B_{-(i-1)}, \dots, B_0$.

La figure 2 présente le contenu de la première bande après un et deux déplacements de la tête de lecture vers la gauche.

On fait la même chose dans l'autre sens si la tête de lecture va vers la droite.

Cette opération est appelée B_i-opération.

On remarque que cette opération garde bien les invariants définis plus haut, et simule donc correctement la machine M_1.

	B_{-2}		B_{-1}	B_0	B_1	B_2		
...								...
...	a_{-3}	a_{-2}	a_{-1}	a_0	a_1	a_2	a_3	...
...					a_0			...
...	a_{-3}	a_{-2}		a_{-1}	a_1	a_2	a_3	...
...						a_0	a_1	...
...			a_{-3}	a_{-2}	a_{-1}	a_2	a_3	...

FIG. 2 – Contenu de la première bande après un et deux déplacements de la tête de lecture vers la gauche

1.2.2 Complexité

La complexité en temps de chacune des opérations décrites pour la réalisation d'une $B - i$-opération sont proportionnelles à la taille de B_i, donc il existe une constante c telle qu'une B_i-opération se fasse en moins de $c2^i$.

Théorème 1. *La simulation prend $O(T(n) \log T(n))$ opérations.*

Démonstration. Après une B_i-opération, les $B_{-(i-1)}, \ldots, B_{i-1}$ sont à moitié pleins, donc il faudra au moins 2^{i-1} opérations avant de pouvoir refaire une B_i-opération.

Pour la même raison, il faut au moins 2^{i-1} opérations avant de pouvoir faire une B_i-opération ; donc on a au maximum $\frac{T(n)}{2^{i-1}}$ B_i-opérations.

On ne peut pas faire de B_i-opération avec $2^i \geq T(n)$, donc $i \leq \log_2 T(n)$. Donc la complexité temporelle de M_2 est :

$$T_2(n) \geq k \sum_{i=1}^{\log_2 T(n)+1} m2^i \frac{T(n)}{2^{i-1}}$$

On en déduit :

$$T_2(n) \geq 2kmT(n)(\log_2 T(n) + 1)$$

Et donc que $T_2(n) = O(T(n) \log T(n))$

□

2 Théorème de hiérarchie temporelle

Définition 5. *Une* fonction $f(n)$ *est* complètement constructible en temps *si il existe une machine de Turing s'arrêtant en $f(n)$ étapes quelle que soit l'entrée de taille n.*

Exemple 1.
- *$n + 1$ est complètement constructible : la machine qui va toujours à droite jusqu'à rencontre du symbole de bande vide effectue $n+1$ opérations quelque soit le mot de longueur n.*
- *La plupart des fonctions usuelles supérieures à $n + 1$ sont complètement constructibles.*

Proposition 1. *Quelle que soit la machine de Turing M encodée par un mot de taille n dans un alphabet A, il existe une machine M' encodée par un mot de taille strictement supérieure à n dans A, et effectuant exactement les mêmes opérations.*

Démonstration. On peut, par exemple, rajouter des transitions inaccessibles, ce qui fera grossir la description de la machine. □

Corollaire 1. *Quel que soit $k > 0$, et quel que soit la machine de Turing M, et quel que soit A, il existe un encodage de M dans A de taille supérieure à k.*

Théorème 2 (Hiérarchie temporelle)**.** *Soient $T_1(n)$ et $T_2(n)$ deux fonctions.*
Si :
– *$T_2(n)$ est complètement constructible en temps ;*
– $\liminf_{n\to\infty} \frac{T_1(n)\log T_1(n)}{T_2(n)} = 0$ *;*
Alors il existe un langage dans ***TIME***$(T_2(n))$ *mais pas dans* ***TIME***$(T_1(n))$.

Démonstration. Soit la machine de Turing M qui prend en entrée une machine de Turing. On simule celle-ci avec 2 bandes avec l'algorithme vu précédemment, et on fait tourner en parallèle une ¡¡horloge¿¿ qui arrête la simulation si elle dure plus que $T_2(n)$ opérations ; d'où le besoin que $T_2(n)$ soit complètement constructible.

Si la simulation finit avant l'horloge, on répond l'opposé de la simulation, sinon, on accepte.

Cette machine est donc en temps $T_2(n)$.

Soit M' une machine de Turing en $T_1(n)$. Il existe c tel que M' soit simulé avec au plus $cT_1(n)\log T_1(n)$.

Il existe $k > 0$ tel que pour tout $n \geq k$, $cT_1(n)\log T_1(n) \geq T_2(n)$. Il existe un encodage de M' de taille supérieure à k, donc la simulation finira.

Quelle que soit la machine M' en temps $T_1(n)$, $M' \in L(M') \Leftrightarrow M' \notin L(M)$, d'où M vérifie un langage différent de tous les langages en $T_1(n)$.

Donc, $L(M)$ est dans **TIME**$(T_2(n))$ et pas dans **TIME**$(T_1(n))$.

□

Exemple 2 (**EXP** $\neq$ **P**)**.** *Pour tout $k, m > 1$, on a :*

$$\lim_{n\to\infty} \frac{n^k \log(n^k)}{2^{mn}} = 0$$

Et on a 2^{mn} qui est complètement constructible en temps.
D'après le théorème précédent, on a ***TIME***$(n^k) \subsetneq$ ***TIME***(2^{mn}).
On a donc ***EXP*** $\neq$ ***P***.