Développement : Théorème de Savitch

Pierron Théo – Huguet Lauriane

13 avril 2014

Lemme 1
Soit M une machine de Turing de complexité temporelle $t(n)$ et de complexité spatiale $s(n)$. Alors il existe une constante $K > 0$ telle que $t(n) \leqslant 2^{Ks(n)}$.

Théorème 1 *Si* $s(n) \geqslant n$ *alors* $\text{NSpace}(s(n)) \subset \text{Space}(s(n)^2)$

Corollaire 1 PSpace=NPSpace

Démonstration. Soit M une machine de Turing non déterministe de complexité en espace $s(n)$. On peut supposer qu'il existe un unique étant final q_f. De plus, on peut supposer que la machine efface son ruban avant d'accepter. Ainsi, la seule configuration acceptantes est (q_f, ε).

Soit w un mot de longueur n.

On va donner un algorithme déterministe de complexité spatiale $O(s(n)^2)$ pour déterminer si un sommet du graphe des configurations de M est accessible depuis la configuration (w, q_0).

Algorithme 1: Access(C, C', t, r)

Entrées : C et C' deux configurations, t et r deux entiers
Sorties : oui ssi il existe un calcul $C \to^* C'$ de longueur au plus t utilisant des configurations de taille au plus r

```
1  si t = 0 alors
2  |  retourner C = C'
3  sinon
4  |  si t = 1 alors
5  |  |  retourner C = C' ou C → C'
6  |  sinon
7  |  |  pour C'' configuration de taille au plus r faire
8  |  |  |  si Access(C, C'', ⌊t/2⌋, r) et Access(C'', C', ⌈t/2⌉, r) alors
9  |  |  |  |  retourner oui
10 retourner non
```

D'après nos hypothèses sur M et d'après le lemme, $w \in L(M)$ ssi Access((q_0, w), (q_f, ε), $2^{Ks(n)}$, $s(n)$).

La complexité spatiale de la fonction Access est $O(\log r + (2r + \log t)\log t)$ car :

- La profondeur de pile maximale est $\log(t)$ car à chaque étape, t est divisé par 2
- À chaque appel récursif, on doit stocker C, C' et t d'où une mémoire de $2r + \log(t)$
- r ne varie pas dans la fonction donc il peut être stocké à part dans $\log(r)$ cases mémoire

Ainsi, l'appel Access((q_0, w), (q_f, ε), $2^{Ks(n)}$, $s(n)$) a une complexité de $O(\log(s(n)) + K(2 + K)s(n)^2) = O(s(n)^2)$.

Supposons dans un premier temps que $s(n)$ est calculable en espace $O(s(n))$. Alors la machine M' qui :

- Prend en entrée un mot w de taille n

- Calcule $s(n)$
- Exécute l'algorithme Access((q_0, w), (q_f, ε), $2^{Ks(n)}$, $s(n)$).

est déterministe, a une complexité spatiale $O(s(n)^2)$ et $L(M') = L(M)$.

On ne suppose plus que $s(n)$ est calculable en espace $O(s(n))$. Soit w un mot de taille n. On définit m comme étant la taille maximale d'une configuration accessible à partir de (q_0, w). On va donner un algorithme qui calcule m en espace $O(s(n)^2)$.

Pour $k > n$, on définit N_k comme le cardinal de l'ensemble E_k des configurations de taille au plus k accessibles depuis (q_0, w) en passant que par des configurations de taille au plus k.

La suite N_k est croissante et majorée par le nombre de configurations de taille au plus $s(n)$. Ainsi il existe k tel que $N_k = N_{k+1}$. On pose $k = \min\{i, N_i = N_{i+1}\}$. Par définition de m, on a $k \leqslant m$.

Supposons $k < m$. Alors il existe un calcul à partir de (q_0, w) utilisant des configurations de taille au moins $k + 1$. Soit C la première configuration de ce calcul de taille $k + 1$. On a alors

$$(q_0, w) \to C_1 \to \cdots \to C_p \to C$$

avec $C_1, \ldots, C_p$ de taille au plus k. Alors $C \in E_{k+1} \setminus E_k$ donc $N_{k+1} > N_k$, ce qui est absurde. Alors $k = m$. On déduit de ceci l'algorithme suivant, qui calcule m à partir de w :

Algorithme 2: Calcul_m(w)

```
1  k ← |w|
2  i ← 0
3  répéter
4  |   k ← k + 1
5  |   N ← i
6  |   pour C de taille au plus k faire
7  |   |   si Access((q0, w),C,2^Kk,k) alors
8  |   |   |   i ← i + 1
9  jusqu'à N = i;
10 retourner k
```

$k \leqslant m$ donc l'espace utilisé par chaque appel à Access est en $O(m^2)$. L'espace utilisé par i et N est inférieur à m donc on a un algorithme en $O(m^2)$. Comme $m \leqslant s(n)$, on obtient un algorithme en $O(s(n)^2)$.

Ainsi, la machine de Turing qui :

- Prend en entrée w
- Calcule m
- Exécute Access((q_0, w),(q_f, ε),2^{Km},m)

est déterministe, a une complexité spatiale en $O(s(n)^2)$ et vérifie $L(M) = L(M')$, d'où le résultat.

■