Preuve du théorème de Savitch

Julie Parreaux

2018-2019

Référence du développement : Sipser [2, p.335]

Leçons où on présente le développement : 913 (Machines de Turing) ; 915 (Classes de complexité).
Leçons dans lesquelles on peut l'évoquer : 926 (Analyse complexité) ; 931 (Schéma algorithmique) (application du diviser pour régner).

1 Introduction

Le théorème de Savitch permet de montrer l'égalité entre deux classe de complexité NPSPACE et PSPACE, ce qui est un des rares résultats de ce type. Généralement, lorsque nous nous donnons deux classes de complexité nous ne pouvons pas dire si elles sont distinctes ou non (P égal ou non NP).

Dans le cas très particulier de la complexité en espace polynomiale, le théorème de Savitch, nous assure que l'équivalence entre une machine déterministe et non déterministe est solvable en espace polynomial (ce qui induit PSPACE = NPSAPCE).

Remarques sur le développement

Le développement présente une preuve du théorème de Savitch : il en existe des théorèmes qui sont plus généraux (voir le Carton [1] ?).

1. Hypothèse sur M.
2. Construction de M'.
3. Complexité spatiale de M'.
   (a) Majoration du nombre de configurations accessibles depuis la configuration initiale.
   (b) Caractérisation de la présence d'un chemin.

2 Théorème de Savitch

Théorème. *Soit $f : \mathbb{N} \to \mathbb{R}^+$ telle que $f(n) \geq n$, pour n assez grand. Toute machine de Turing non-déterministe qui décide en espace $f(n)$ est équivalente à une machine de Turing déterministe en espace $O(f^2(n))$.*

2.1 Idée de la preuve

On souhaite modéliser le comportement d'une machine non-déterministe à l'aide d'une machine déterministe en bornant l'espace que cette dernière peut utiliser (en fonction de l'espace occupé par la première).

L'idée est alors d'utiliser le graphe des configurations de la machine non-déterministe (qui est borné par hypothèse), et de lui appliqué le problème d'accessibilité dans un graphe orienté. Comme celui-ci est dans NL, on va pouvoir exploiter sa complexité spatiale afin d'obtenir le résultat que l'on recherche. On va donc "ramener" le problème au d'accessibilité dans un graphe orienté.

2.2 Preuve du théorème

Nous allons montrer une version plus faible du théorème de Savitch que nous énonçons ci-dessous.

Théorème. *Soit $f : \mathbb{N} \to \mathbb{R}^+$ telle que $f(n) \geq n$ et $f(n)$ est calculable en espace $O(f)$. Alors, $NSPACE(f) \subseteq SPACE(f^2)$.*

Démonstration. Soit M une machine non-déterministe qui décide un langage L en espace $f(n)$ ($L \in NSPACE(f)$). Montrons qu'il existe une machine M' qui simule M pour décider L en espace $O(f^2(n))$.

Hypothèses sur M Nous supposons sans perte de généralité que M possède un unique état initial q_{init} et un unique état final q_{final} qui est atteint une fois que M a effacé son ruban et que la tête de lecture est revenue se placer à gauche. On notera c_{final} la configuration finale (liée à l'état final). Soit w l'entrée sur M, on note c_{init} la configuration initial correspondant à w.

On s'intéresse à la mémoire utilisée par la machine de Turing durant son exécution (pas à son temps de calcule ou à sa taille), c'est à dire la quantité de ruban que la machine utilise durant son exécution. Les changements que nous effectuons ici, ne touche en aucun cas à la quantité de ruban utilisée puisque la seule opération effectuée sur le ruban par cette nouvelle machine consiste à l'effacer ce qui ne change pas la quantité de ruban utilisé.

Définition de M' On définit M' par l'algorithme (1). Elle prend en entrée w l'entrée de M et elle accepte l'exécution si $w \in L(M)$ (elle la rejette dans le cas contraire). On remarque que cet algorithme est bien déterministe.

Algorithm 1 Définition de la machine de Turing déterministe M'

```
1: procedure M'(w)
2:     if (c_init) ⊢*_M c_accept dans le graphe des configuration de M then
3:         Accepter
4:     else
5:         Rejeter
6:     end if
7: end procedure
```

Complexité de M' Nous allons évaluer la complexité de M' en étudiant la complexité pour le calcul de ce prédicat : $(c_init) \vdash^*_M c_{accept}$. On remarque dans un premier temps que si on a $(c_init) \vdash^*_M c_{accept}$ alors il existe un chemin de c_{init} à c_{final} dans le graphe des configurations de M (Le graphe des configuration est un graphe dont les sommets sont les configurations de la machine de Turing et les arrêtes sont définies telles que la fonction de transition permet de passer d'une configuration à une autre en une étape.) que l'on note $(c_init) \to^*_M c_{accept}$.

Pour ce calcul de complexité, nous allons majorer le nombre de configuration accessible à partir $c_{initial}$ et définir une fonction qui calcul le chemin de $c_{initial}$ à c_{final} en majorant le nombre d'étapes. Pour cela, nous allons utiliser deux lemmes.

Lemme. *Il existe un $d \in \mathbb{N}$ tel que le nombre de configuration accessible depuis $c_{initial}$ soit majoré par $2^{df(|w|)}$.*

Démonstration. Le nombre de configuration accessible depuis $c_{initial}$ est majoré par $|\mathcal{Q}| \times |\Gamma|^{f(|w|)} \times f(|w|)$ car le nombre d'état que l'on peut atteindre, le nombre de mot que l'on peut écrire sur le ruban sans violer la propriété sur M et le nombre de position possibles pour la tête de lecture. □

Lemme. *$c_{initial} \to^* c_{accept}$ si et seulement si $c_{initial} \to^{2df(|w|)} c_{accept}$.*

Nous allons alors nous appuyer sur la fonction CHEMIN ? dont voici la spécification. Pour deux configurations c_1 et c_2 de M, ainsi qu'un entier t, CHEMIN ?(c_1, c_2, t) retourne *vrai* s'il existe un chemin (une suite de configuration) de c_1 vers c_2 dans le graphe des configurations de M en au plus t étapes ; et *faux* sinon. Dans la suite, on suppose que t est une puissance de 2 (dans le cas contraire, comme on a toujours l'existence d'une puissance de 2 qui est plus grande que t, on prend la plus petit de ces puissance de 2 comme nouvelle valeur de t (ce n'est pas géant car on cherche à majorer la complexité)).

Démonstration. Le sens indirect est immédiat. Nous allons alors montrer la réciproque.

Nous implémentons le test $c_{init} \to^* c_{final}$ à l'aide de la fonction CHEMIN ?$(c_{initial}, c_{accept}, 2^{df(|w|)})$ où CHEMIN ? est définit comme précédemment et par l'algorithme 2.

Algorithm 2 Fonction CHEMIN ?

```
1:  function CHEMIN ?(c_1, c_2, t)
2:      if t = 1 then
3:          Renvoie c_1 →^{≤1} c_2
4:      else
5:          for toute configuration c de taille de ruban d'au plus f(|w|) do
6:              if CHEMIN ?(c_1, c, t/2) et CHEMIN ?(c, c_2, t/2) then
7:                  Renvoie vraie
8:              end if
9:          end for
10:         Renvoie faux
11:     end if
12: end function
```

La fonction CHEMIN ? est suit le principe diviser pour régner. Sa complexité spatiale est alors de $Cout(t) = O(f) + Cout(\frac{t}{2})$ où t est le troisième argument de CHEMIN ? (Hypothèse 2 : on peut calculer $f(|w|)$ en $O(f)$). On obtient donc $Cout(t) = O(f \log_2 t)$ (par le master théorème). La complexité spatiale de M', par l'hypothèse 1 est de $O(f) + Cout(2^{df(|w|)}) = O(f) + O(f \log_2(2^{df(|w|)}) = O(f) + O(fdf) = O(f) + O(f^2) = O(f^2)$ car on calcul une fois $f(|w|)$ pour la fonction CHEMIN ? et par l'hypothèse 2 se fait en $O(f)$ et on exécute la fonction CHEMIN ? avec $t = 2^{df(|w|)}$.

□

D'où le résultat.

□

2.3 Remarques sur la preuve

— On peut le faire dans le cadre du plan mais il faut faire attention à la complexité du calcul de $f(|w|)$ Pour contrer cela, on appel la fonction CHEMIN ? pour chaque valeur de $f(i)$ et on incrémente i pas à pas tant qu'on a pas trouvé de chemin acceptante. (Cela nous permet de calculer expérimentalement la borne que l'on a ajouté dans les hypothèses).

— L'hypothèse $f(n) \geq n$ nous permet de lire l'entrée sur le ruban ; cependant avec une machine à plusieurs rubans, $f(n) \geq \log n$ suffit.

— Cette preuve est basée sur la NL-complétude du problème d'accessibilité dans un graphe. Avec cet argument (non trivial), la preuve devient très facile.

Pendant le calcul de la complexité, nous utilisons de manière centrale ce théorème A savoir énoncer et montrer !

Théorème (Master theorem). *Soient $t : \mathbb{N} \mapsto \mathbb{R}_+$ une fonction croissante à partir d'un certain rang n_0, $b \geq 2$ entiers, $k \geq 0$ entier et $a, b, c, d > 0$ réels positifs tels que pour tout n tel que $\frac{n}{n_0}$ puissance de b :*

$$\begin{cases} t(n_0) = b \\ t(n) = at(\frac{n}{b}) + cn^b \end{cases}$$

Alors, on a :

$$t(n) = \begin{cases} \Theta(n^k) & si\ a < b^k \\ \Theta(n^k \log_b n) & si\ a = b^k \\ \Theta(n^{\log_b a}) & si\ a > b^k \end{cases}$$

Références

[1] O. Carton. *Langages formels, calculabilité et complexité*. Vuibert, 2008.

[2] M. Sipser. *Introduction to the Theory of Computation*. Cengage learning, 1133187811.