

Complexité en espace.

Remarque 1 (Rappels). On a que

- ▷ $\text{NSPACE}(s(n)) \subseteq \text{DTIME}(2^{O(s(n))})$
- ▷ et $\text{NSPACE}(s(n)) \subseteq \text{DSPACE}(s(n)^2)$ (Savitch)

lorsque $s(n) \geq \log n$. On a

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{PSPACE},$$

et avec Savitch et la hiérarchie en espace, on a

$$\text{NL} \subsetneq \text{PSPACE}.$$

Définition 1. On définit

$$\text{PSPACE} := \bigcup_{k \geq 1} \text{DSPACE}(n^k).$$

Remarque 2. On peut aussi définir la classe NPSPACE mais cette classe est égale à PSPACE par Savitch. On a donc

$$\text{P} \subseteq \text{NP} \stackrel{(*)}{\subseteq} \text{PSPACE} \stackrel{(**)}{\subseteq} \text{EXPTIME} = \bigcup_{k \geq 1} \text{DTIME}(2^{n^k}),$$

où

- ▷ $(*)$ est vraie car $\text{NTIME}(t(n)) \subseteq \text{DSPACE}(t(n))$;
- ▷ $(**)$ est vraie car $\text{DSPACE}(s(n)) \subseteq \text{DTIME}(2^{O(s(n))})$.

On sait, de plus, que $\text{P} \subsetneq \text{EXPTIME}$ par la hiérarchie en temps (variante (vue en TD) de la hiérarchie en espace vue avant).

1 Formules booléennes quantifiées.

On autorise des quantificateurs universels et existentiels aux formules vues précédemment.

Exemple 1. Les formules

- ▷ $\forall x \exists y \left(\overbrace{(x \vee y) \wedge (\bar{x} \vee \bar{y})}^{\text{c'est } x \text{ xor } y} \right)$
- ▷ $\exists y \forall x \left((x \vee y) \wedge (\bar{x} \vee \bar{y}) \right)$

sont des formules booléennes quantifiées. La première est vraie (avec $y = \bar{x}$) mais pas la seconde (avec $x = y$). On suppose que les quantificateurs sont tous en début de formule (forme prénexe).

Définition 2. On définit le problème

QBF | **Entrée.** Une formule booléenne quantifiée F (close)
Sortie. Est-ce que F est vraie ?

Proposition 1. On a $\text{QBF} \in \text{PSPACE}$.

Preuve. On utilise l'algorithme suivant.

1. Si F ne contient pas de quantificateurs, accepter si F s'évalue à vrai et rejeter si F s'évalue à faux.
2. Si $F = \exists x G$, on décide récursivement avec $G[x := 0]$ et $G[x := 1]$ Si une de ces formules s'évalue à vrai, accepter sinon rejeter.
3. Si $F = \forall x G$, on décide récursivement avec $G[x := 0]$ et $G[x := 1]$ Si une de ces formules s'évalue à faux, rejeter sinon accepter.

Cet algorithme utilise un espace linéaire. □

Théorème 1. Le problème **QBF** est **PSPACE**-complet.

Preuve. Supposons que A est résolu en espace n^k par une machine M à un ruban. On va montrer que $A \leq_P \text{QBF}$.

On considère le diagramme espace-temps de taille n^k en espace et $2^{c \cdot n^k}$ en temps. On ne peut pas simplement utiliser la même preuve que pour SAT car le diagramme est exponentiel.

On construit une formule $\phi_t(c_1, c_2)$ qui exprime qu'on peut aller de la configuration c_1 en la configuration c_2 en au plus 2^t étapes de calcul.

On a donc que M accepte si $\phi_{c \cdot n^k}(c_{\text{initiale}}, c_{\text{finale}})$.¹

La formule $\phi_0(c_1, c_2)$ est de taille polynomiale (*c.f.* preuve du théorème de Cook).

Pour aller de ϕ_t à ϕ_{t+1} , on cherche la configuration du milieu. On pourrait utiliser $(*) := \exists c \phi_t(c_1, c) \wedge \phi_t(c, c_2)$ qui est correcte mais trop couteuse (taille exponentielle). On choisit plutôt :

$$\exists c \forall c_3 \forall c_4 [(c_3 = c_1 \wedge c_4 = c) \vee (c_3 = c \wedge c_4 = c_2)] \Rightarrow \phi_t(c_3, c_4).$$

Cette formule est logiquement équivalente à $(*)$ (en regardant les cas où $(c_3, c_4) = (c_1, c)$ et $(c_3, c_4) = (c, c_2)$). Il est important de se rappeler que c, c_3, c_4 ne sont pas des variables mais des n -uplets de taille $O(n^k)$ variables booléennes. La taille de ϕ_{t+1} augmente de $O(n^k)$ par rapport à la taille de ϕ_t . Au final, on est de taille $O(n^{2k})$.² \square

Le problème **QBF** est « le » problème **PSPACE**-complet. En TD, on fera des réductions de certains problèmes à **QBF**.

1. On peut considérer qu'il y a une unique configuration acceptante, quitte à transformer tout état acceptant en un état qui efface tout le ruban et remet la tête en position initiale.

2. Le passage quadratique est similaire au théorème de Savitch.

Théorème 2 (Savitch). On a $\text{NSPACE}(s(n)) \subseteq \text{DSPACE}(s(n)^2)$ si $s(n) \geq \log n$.³ □

Corollaire 1. On a $\text{NL} \subseteq \text{DSPACE}(\log^2 n)$.

Proposition 2. On a que $\text{PATH} \in \text{DSPACE}(\log^2 n)$, où

PATH | Entrée. Un graphe G orienté, et $s, t \in V(G)$

| Sortie. Existe-t-il un chemin de s à t dans G ?

Remarque 3. En TD, on a vu que PATH est NL -complet, et donc la proposition implique le corollaire. En effet, soit $A \in \text{NL}$ et $x \in \{0, 1\}^n$, on a

$$x \in A \iff f(x) \in \text{PATH},$$

où $f : A \leq_L \text{PATH}$ est la réduction en espace logarithmique (car le problème PATH est NL -complet). La construction de $f(x)$ se fait en espace $O(\log n)$ et décider si $f(x) \in \text{PATH}$ ou non peut se faire en espace $O(\log^2 |f(x)|)$, or $|f(x)|$ est polynomial en $|x| = n$, d'où la borne annoncée.

Preuve (de la proposition). On donne un algorithme $\text{path}(G, u, v, i)$ en espace $O(\log^2 n)$ qui décide s'il existe, dans G , un chemin de u à v de longueur au plus 2^i . On pourra donc résoudre PATH en appelant $\text{path}(G, s, t, \lceil \log n \rceil)$.

```

1 : Procédure path( $G, u, v, i$ )
2 :   si  $i = 0$  alors
3 :     si  $uv \in E(G)$  ou  $u = v$  alors Accepter
4 :     sinon Rejeter

```

3. Depuis sa preuve, on n'a jamais réussi à améliorer ce résultat, ni montrer qu'un facteur carré est nécessaire.

```

5 :   pour tout sommet  $w \in V(G)$  faire
6 :     si  $\text{path}(G, u, w, i - 1)$  et  $\text{path}(G, w, v, i - 1)$  alors
7 :       Accepter
8 :     Rejeter

```

Pour la correction, on montre si $\text{path}(G, u, v, i)$ accepte alors il existe un chemin de longueur au plus 2^i par récurrence sur i .

- ▷ Pour le cas $i = 0$, c'est bon par le premier « **si** ».
- ▷ Pour l'héritage, si on a un chemin de longueur au plus 2^{i-1} de u à w et un chemin de longueur au plus 2^{i-1} de w à v , on concatène ces chemins pour obtenir un chemin de u à v de longueur au plus 2^i .

Réciproquement, s'il existe un chemin de u à v de longueur au plus 2^i , alors $\text{path}(G, u, v, i)$ accepte, car il suffit de choisir w comme sommet milieu du chemin.

On a $O(\log n)$ appels récursifs ; et à chaque appel, on doit mémoriser w ce qui demande $O(\log n)$ bits. On en déduit une complexité en espace en $O(\log^2 n)$. \square

Preuve (du théorème de Savitch). Supposons que A peut être résolu par une machine non-déterministe M en espace $O(s(n))$ où est $s(n)$ constructible en espace. Soit $x \in \{0, 1\}^n$ une instance de A .

On considère le graphe G_x des **configurations potentielles** de la machine M sur l'entrée x , c'est-à-dire l'ensemble des configurations avec x en entrée et au plus $c \cdot s(n)$ cases utilisées sur chaque ruban de travail.

Lemme 1. Le graphe G_x a $2^{O(s(n))}$ sommets et peut être construit en espace $O(s(n))$. \square

On a que

$$x \in A \iff (G_x, s, t) \in \text{PATH},$$

où s est la configuration initiale de M sur l'entrée x et t la configuration acceptante (qu'on supposera unique, *c.f.* preuve de la PSPACE-complétude de QBF). Par le lemme, on a que (G_x, s, t) se fait en espace $O(s(n))$. Décider si $(G_x, s, t) \in \text{PATH}$ se fait en espace $O(\log^2 |G_x|)$ donc $O(s(n)^2)$. \square

Remarque 4. La preuve précédente utilise deux résultats

- ▷ le théorème de composition *amélioré* :

Théorème 3 (Composition). Soient f et g deux fonctions calculables en espace $s_f(n)$ (*resp.* $s_g(n)$). On peut calculer la composée $(f \circ g)(x)$ en espace $O(s_g(|x|)) + s_f(|g(x)|))$. \square

- ▷ et le fait que l'on pourra supprimer l'hypothèse de constructibilité de $s(n)$:

Pour cela, on essaye $s(n) = 1, 2, \dots$ et on s'arrête à $s(n) = i$ si aucune configuration de taille $i + 1$ n'est accessible à partir de la configuration initiale sur l'entrée x (appel à l'algorithme pour PATH).

Remarque 5. Le problème PATH dans les graphes non-orientés est dans L! C'est un résultat récent (2005).

La prochaine fois, on s'intéresse aux oracles.