

Chapitre 6

Les classes PSPACE et IP

Complexité de temps

Nous allons considérer le temps comme mesure de complexité.

Nous avons déjà vu les classes de complexité P et EXPTIME.

Rappel :

$$\text{EXPTIME} = \bigcup_{k \geq 0} \text{TIME}(2^{n^k}).$$

Définition 6.1. Une fonction

$$t : \mathbb{N} \rightarrow \mathbb{N}$$

est **totale-temps-constructible** s'il existe une MT M telle que

$$L(M) \in \text{TIME}(t(n))$$

et si $\forall n \in \mathbb{N}$ et $\forall w \in \Sigma^*$ tels que $|w| = n$, on a que M s'arrête après $t(n)$ étapes sur entrée w . 

Remarques 6.2.

- Les fonctions $\lfloor \log(n+1) \rfloor$, n , n^k , 2^n et $n!$ sont totalement-temps-constructibles.
- Si t_1 et t_2 sont totalement-temps-constructibles, alors $t_1(n)t_2(n)$, $2^{t_1(n)}$ et $t_1(n)^{t_2(n)}$ sont totalement-temps-constructibles.



Théorème 6.3. Si t_2 est totalement-temps-constructible et si

$$\liminf_{n \rightarrow \infty} \frac{t_1(n) \log(t_1(n))}{t_2(n)} = 0$$

alors $\text{TIME}(t_1(n)) \subset \text{TIME}(t_2(n))$. ■

Cela signifie, par exemple, que plus de langages peuvent être reconnus en temps $O(n^3)$ qu'en temps $O(n^2)$.

Théorème 6.4.

$$P \subset EXPTIME$$



Complexité d'espace

Considérons maintenant l'espace comme mesure de complexité :

- l'espace logarithmique ;
- l'espace polynômial ;
- l'espace exponentiel.

Définition 6.5. Soit

$$s : \mathbb{N} \rightarrow \mathbb{N}.$$

Une MT utilise un **espace** $s(n)$ si, pour tout mot de longueur n , la tête de lecture-écriture ne se déplace jamais vers la droite de plus de $s(n)$ positions.

Pour définir un espace de calcul inférieur à la taille de l'input, nous considérons que l'input réside sur un ruban en lecture seule, et qu'un autre ruban est utilisé pour faire le calcul. ▲

Définition 6.6.

$$\text{SPACE}(s(n)) = \{L \mid L \text{ est décidé par une MT en temps } O(s(n))\}.$$



Définition 6.7.

$$\textcolor{red}{L} = \text{SPACE}(\log n)$$



Définition 6.8.

$$\textcolor{red}{PSPACE} = \bigcup_{k \geq 0} \text{SPACE}(n^k)$$



Définition 6.9.

$$\textcolor{red}{EXPSPACE} = \bigcup_{k \geq 0} \text{SPACE}(2^{n^k})$$



Théorème 6.10.

$$\text{TIME}(f(n)) \subseteq \text{SPACE}(f(n))$$



Théorème 6.11. Si $L \in \text{SPACE}(f(n))$ et $f(n) \geq \log n$, alors il existe une constante c qui dépend de L telle que $L \in \text{TIME}(c^{f(n)})$. ■

Définition 6.12. Une fonction

$$s : \mathbb{N} \rightarrow \mathbb{N}$$

est **totalement-espace-constructible** s'il existe une MT M telle que

$$L(M) \in \text{SPACE}(s(n))$$

et si $\forall n \in \mathbb{N}$ et $\forall w \in \Sigma^*$ tels que $|w| = n$, on a que M utilise exactement $s(n)$ positions du ruban sur entrée w . 

Remarques 6.13.

- Les fonctions $\lfloor \log(n+1) \rfloor$, n , n^k , 2^n et $n!$ sont totalement-espace-constructibles.
- Si s_1 et s_2 sont totalement-espace-constructibles, alors $s_1(n)s_2(n)$, $2^{s_1(n)}$ et $s_1(n)^{s_2(n)}$ sont totalement-espace-constructibles.



Théorème 6.14. Si s_2 est totalement-espace-constructible et si

$$\liminf_{n \rightarrow \infty} \frac{s_1(n)}{s_2(n)} = 0$$

alors $\text{SPACE}(s_1(n)) \subset \text{SPACE}(s_2(n))$. ■

Cela signifie, par exemple, que plus de langages peuvent être reconnus en espace $O(n^3)$ qu'en espace $O(n^2)$.

Théorème 6.15.

$$L \subset PSPACE \subset EXPSPACE$$



Théorème 6.16.

$$L \subseteq P$$

Preuve. Si $A \in L$ alors il existe une MT qui décide A en espace borné par $a \log n$ et donc qui est bornée en temps par

$$\begin{aligned} c^{a \log n} &= 2^{b \log n} \\ &= (2^{\log n})^b \\ &= n^b. \end{aligned}$$



Théorème 6.17.

$$P \subseteq PSPACE.$$



Corollaire 6.18.

$$L \subset P \text{ ou } P \subset PSPACE.$$



Exemples 6.19.

- $\{w \mid |w|_{\mathbf{a}} = |w|_{\mathbf{b}}\} \in \mathbf{L}$
- $\{w \mid w = w^{\mathbf{R}}\} \in \mathbf{L}$
- $\{\langle x, y, z \rangle \mid x, y, z \in \mathbb{N} \text{ et } z = x + y\} \in \mathbf{L}$
- $\{\langle x, y, z \rangle \mid x, y, z \in \mathbb{N} \text{ et } z = xy\} \in \mathbf{L}$



Définition 6.20. Une **expression booléenne quantifiée (EBQ)** est une expression de la forme $Q_1x_1 Q_2x_2 \dots Q_nx_n : E$ où :

- les x_i sont des variables booléennes ;
- les Q_i sont des quantificateurs (*pour tout* $\forall$ ou *il existe* $\exists$) ;
- E est une expression booléenne ;
- toutes les variables apparaissant dans E doivent être quantifiées.



Définition 6.21.

$$\textcolor{red}{QBF} = \{ \langle E \rangle \mid E \text{ est une EBQ dont la valeur est } 1 \}$$



Exemple 6.22. L'expression $\exists x \forall y \exists z : (x \vee y) \wedge (y \vee z)$ est une EBQ qui appartient à QBF car :

- on choisit $x = 1$;
- l'expression devient : $\forall y \exists z : y \vee z$;
- si $y = 0$, on choisit $z = 1$;
- si $y = 1$, on choisit $z = 1$;
- dans les deux cas l'expression donne 1.



Exemple 6.23. L'expression $\forall x \forall y \exists z : (x \vee y) \wedge \neg z$ est une EBQ qui n'appartient pas à QBF car :

- si $x = 0$ et $y = 0$, l'expression devient $\exists z : 0$;
- cette expression vaut 0 quelle que soit la valeur choisie pour z .



Théorème 6.24.

$$QBF \in PSPACE$$

Aperçu de la preuve. On montre qu'on peut décider de l'appartenance d'une EBQ à QBF en espace polynômial à l'aide d'un algorithme récursif.

- Si l'EBQ ne possède qu'un seul quantificateur, alors l'expression a au plus une variable et on peut facilement calculer sa valeur en espace polynômial.

- Si l'EBQ contient deux quantificateurs ou plus, alors on considère deux cas :
 - Si l'EBQ commence par $\forall$, alors on donne successivement les valeurs 0 et 1 à la variable quantifiée par ce $\forall$, et on appelle récursivement la procédure de vérification deux fois. On retourne la conjonction des deux valeurs retournées par ces appels.
 - Si l'EBQ commence par $\exists$, alors on donne successivement les valeurs 0 et 1 à la variable quantifiée par ce $\exists$, et on appelle récursivement la procédure de vérification deux fois. On retourne la disjonction des deux valeurs retournées par ces appels.

La profondeur de la récursion est bornée par le nombre de variables, donc la taille de la pile et l'espace mémoire utilisé seront de tailles polynômiales. ■

Définition 6.25. Le langage L est **PSPACE-complet** si

- $L \in \text{PSPACE}$;
- pour tout $L' \in \text{PSPACE}$ on a $L' \leq_P L$.



Théorème 6.26.

QBF est PSPACE-complet.



Définition 6.27. Une **expression régulière avec exposants (ERE)** sur un alphabet Σ est récursivement définie comme suit :

- $\emptyset$ est une ERE (l'ensemble vide) ;
- ε est une ERE (le mot vide) ;
- si $x \in \Sigma$, alors x est une ERE (un symbole) ;
- si E est une ERE, alors (E) est une ERE (parenthésage) ;
- si E est une ERE, alors E^* est une ERE (fermeture de Kleene) ;
- si E est une ERE et si $n \in \mathbb{N}$, alors E^n est une ERE (exponentiation) ;
- si E_1 et E_2 sont des ERE, alors $E_1 E_2$ est une ERE (concaténation) ;
- si E_1 et E_2 sont des ERE, alors $E_1 | E_2$ est une ERE (union).



Définition 6.28. Soit E une ERE sur l'alphabet Σ . Le **langage engendré** par E , $L(E)$, est l'ensemble des mots de Σ^* qu'on peut former en appliquant les opérations de E sur les symboles de Σ . La priorité des opérations est définie par l'ordre dans lequel elles apparaissent à la définition 6.27. ▲

Exemples 6.29. Si $\Sigma = \{a, b\}$:

- $L(a^*b) = \{b, ab, aab, \dots\}$
- $L((aa)^3) = \{aaaaaa\}$
- $L((aa|bb)^3) =$
 $\{aaaaaa, aaaabb, aabbaa, aabbbb, bbaaaa, bbaabb, bbbbaa, bbbbbb\}$
- $L(a(a|bb)^2) = \{aaa, aabb, abba, abbbb\}$



Définition 6.30.

$$\textcolor{red}{TOUT}_{\text{ERE}} = \{ \langle E \rangle \mid E \text{ est une ERE et } L(E) = \Sigma^* \}$$



Exemple 6.31.

$$\langle (a|b|\varepsilon)^* ((aa)^3|\varepsilon|a|b|aa|ab|ba|bb) \rangle \in TOUT_{ERE}$$



Théorème 6.32.

$$TOUT_{\text{ERE}} \in \text{EXPSPACE}.$$



Définition 6.33. Le langage L est **EXPSPACE-complet** si

- $L \in \text{EXPSPACE}$;
- pour tout $L' \in \text{EXPSPACE}$ on a $L' \leq_P L$.



Théorème 6.34.

$\overline{TOU_{\text{ERE}}}$ est EXPSPACE-complet.



Théorème 6.35.

$$\text{PSPACE} \subset \text{EXPSPACE}.$$



Corollaire 6.36.

$$TOUT_{\text{ERE}} \notin \text{P}.$$

