TD 12 – Circuits

1 Classe AC

Un circuit booléen avec n bits d'entrée est un graphe orienté acyclique (DAG) où les feuilles sont soit une des variables d'entrée x_i ($i \in [\![1, n]\!]$) soit une constante $\top$ ou $\bot$, la sortie est l'unique nœud dont le degré sortant est nul et chaque nœud est soit un nœud $\wedge$ ou $\vee$ (qui peuvent être d'arité quelconque) soit un nœud $\neg$ (unaire). Étant donné un mot $w \in \{0,1\}^n$ avec $n \in \mathbb{N}$ et un circuit booléen dont la taille d'entrée est n, on peut évaluer le circuit sur w en remplaçant la i-ème feuille d'entrée par $\top$ quand $w_i = 1$ et $\bot$ quand $w_i = 0$. Un tel circuit décide donc le langage des mots de $\{0,1\}^n$ sur lesquels il s'évalue à $\top$.

On dit qu'une *famille de circuits* $(C_n)_{n\in\mathbb{N}}$ décide un langage $L \subseteq \{0,1\}^*$ lorsque pour chaque $n \in \mathbb{N}$, le circuit C_n a n bits d'entrée et décide $L \cap \{0,1\}^n$.

On définit la classe AC^i pour $i \in \mathbb{N}$ comme la classe des langages sur $\{0,1\}$ décidés par une famille de circuits booléens $(C_n)_{n\in\mathbb{N}}$ dont la taille est polynomiale et dont la profondeur est en $\mathrm{O}(\log(n)^i)$, c'est à dire qu'il existe un polynôme p et un entier k tels que pour tout $n \in \mathbb{N}$, la taille de C_n est majorée par $p(n)$ et la profondeur de C_n est majorée par $k \cdot \log(n)^i$.

Exercice 1. Justifier que l'on ne s'intéresse qu'aux circuits de taille polynomiale (et non pas exponentielle ou illimitée).

Solution 1. Soit un langage $L \subseteq \{0,1\}^n$ pour $n \in \mathbb{N}$.

En posant $\varphi_u(x_1, \dots, x_n) = \bigwedge_{i\in[\![1,n]\!], u_i=1} x_i \wedge \bigwedge_{i\in[\![1,n]\!], u_i=0} \neg x_i$ pour tout $u \in \{0,1\}^n$, il suit que $\varphi_u(x_1, \dots, x_n)$ s'évalue à $\top$ si et seulement si le mot en entrée est u, et donc on a que le circuit

$$\bigvee_{u\in L} \varphi_u(x_1, \dots, x_n)$$

décide L.

Il est ainsi possible de décider n'importe quel langage $L \subseteq \{0,1\}^n$ avec $n \in \mathbb{N}$ par un circuit de profondeur au plus 3.

Exercice 2. Montrer que le langage $\mathrm{PARITY} = \{w \in \{0,1\}^* \mid |w|_1 = 1 \mod 2\}$ est dans AC^1.

Solution 2. On construit des circuits $\oplus(x_1, \dots, x_n)$ décidant $\mathrm{PARITY} \cap \{0,1\}^n$ pour tout $n \in \mathbb{N}$ par induction sur n. On pose $\oplus(\varepsilon) = \bot$, $\oplus(x_1) = x_1$ et pour tout $n \in \mathbb{N}, n \geq 2$,

$$\begin{aligned}\oplus(x_1, \dots, x_n) =& \big(\oplus(x_1, \dots, x_{\lceil n/2\rceil}) \wedge \neg \oplus(x_{\lceil n/2\rceil+1}, \dots, x_n)\big) \vee \\ & \big(\neg \oplus(x_1, \dots, x_{\lceil n/2\rceil}) \wedge \oplus(x_{\lceil n/2\rceil+1}, \dots, x_n)\big)\ .\end{aligned}$$

Exercice 3. Montrer que le langage

$$\mathrm{ADD} = \{abc \in \{0,1\}^* \mid |a| + 1 = |b| + 1 = |c|,\ a + b = c \text{ avec } a, b, c \text{ en binaire petit-boutiste}\}$$

est dans AC^0.

Solution 3. Dans la suite, pour alléger la notation, pour deux cicuits C et D, on écrira $C \oplus D$ pour $(C \wedge \neg D) \vee (\neg C \wedge D)$ et $C \Leftrightarrow D$ pour $(C \wedge D) \vee (\neg C \wedge \neg D)$. De même, lorsque $n \in \mathbb{N}_{>0}$ suit clairement du contexte, nous écrirons $\overline{x}$ plutôt que $x_1, \dots, x_n$.

Étant donné $n \in \mathbb{N}_{>0}$, nous allons construire les circuits $A_i(\overline{x}, \overline{y})$ et $R_i(\overline{x}, \overline{y})$ pour tout $i \in [\![1, n]\!]$ qui donnent, respectivement, le bit en position i et l'éventuelle retenue obtenue suite à l'addition en

position i dans la somme de $\overline{x}$ et $\overline{y}$ en binaire petit-boutiste. Pour tout $i \in [\![1, n]\!]$, on pose

$$A_i(\overline{x}, \overline{y}) = \begin{cases} R_{i-1}(\overline{x}, \overline{y}) \oplus x_i \oplus y_i & \text{si } i \geq 2 \\ x_1 \oplus y_1 & \text{sinon} \end{cases}$$

et

$$R_i(\overline{x}, \overline{y}) = \begin{cases} \bigvee_{j=1}^{i} \big(x_j \wedge y_j \wedge \bigwedge_{k=j+1}^{i} (x_k \vee y_k)\big) & \text{si } i \geq 2 \\ x_1 \wedge y_1 \end{cases}.$$

Pour décider ADD, on utilise ensuite le circuit

$$\mathrm{ADD}(\overline{x}, \overline{y}, \overline{z}) = \bigwedge_{i=1}^{n} \big(A_i(\overline{x}, \overline{y}) \Leftrightarrow z_i\big) \wedge \big(R_n(\overline{x}, \overline{y}) \Leftrightarrow z_{n+1}\big)$$

pour tout $n \in \mathbb{N}_{>0}$.

Exercice 4. Montrer que le langage

$$\mathrm{MULT} = \{abc \in \{0,1\}^* \mid 2\,|a| = 2\,|b| = |c|,\ a \cdot b = c \text{ avec } a, b, c \text{ en binaire petit-boutiste}\}$$

est dans AC^1.

Solution 4. Faire le produit de deux nombres de n bits revient à faire la somme de n nombres de $2n$ bits, ce que nous allons faire ici.

Étant donné $n \in \mathbb{N}_{>0}$, pour faire la somme de n nombres de $2n$ bits donnés par les variables $\overline{x^{(1)}}, \ldots, \overline{x^{(n)}}$, nous allons réutiliser les circuits $A_i(\overline{x}, \overline{y})$ pour tout $i \in [\![1, 2n]\!]$ de la solution à l'exercice précédent et construire par récurrence sur n, pour tout $i \in [\![1, 2n]\!]$, le circuit $C_i(\overline{x^{(1)}}, \ldots, \overline{x^{(n)}})$ donnant le bit en position i dans la somme de $\overline{x^{(1)}}, \ldots, \overline{x^{(n)}}$ en binaire petit-boutiste. Pour tous $n \in \mathbb{N}_{>0}$ et $i \in [\![1, 2n]\!]$, on pose

$$C_i(\overline{x^{(1)}}, \ldots, \overline{x^{(n)}}) = \begin{cases} A_i\big(\overline{C(\overline{x^{(1)}}, \ldots, \overline{x^{(\lceil n/2 \rceil)}})}, \overline{C(\overline{x^{(\lceil n/2 \rceil + 1)}}, \ldots, \overline{x^{(n)}})}\big) & \text{si } n \geq 2 \\ x_i^{(1)} & \text{sinon} \end{cases}.$$

Maintenant, étant donné $n \in \mathbb{N}_{>0}$, pour calculer le produit de deux nombres de n bits donnés par les variables $\overline{x}, \overline{y}$, on effectue la somme des n nombres de $2n$ bits donnés par $\overline{B^{(1)}(\overline{x}, \overline{y})}, \ldots, \overline{B^{(n)}(\overline{x}, \overline{y})}$, tels que

$$B_j^{(i)}(\overline{x}, \overline{y}) = \begin{cases} x_j \wedge y_i & \text{si } i \leq j \leq n + i - 1 \\ \bot & \text{sinon} \end{cases}$$

pour tous $i \in [\![1, n]\!]$ et $j \in [\![1, 2n]\!]$. Pour décider MULT, on utilise ensuite le circuit

$$\mathrm{MULT}(x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_{2n}) = \bigwedge_{i=1}^{2n} \Big(C_i\big(\overline{B^{(1)}(\overline{x}, \overline{y})}, \ldots, \overline{B^{(n)}(\overline{x}, \overline{y})}\big) \Leftrightarrow z_i\Big)$$

pour tout $n \in \mathbb{N}_{>0}$.

Exercice 5. Montrer que tout langage rationnel sur $\{0,1\}$ est dans AC^1.

Solution 5. Soit $\mathcal{A} = (Q, \{0,1\}, \delta, q_0, F)$ un AFD reconnaissant le langage considéré $L \subseteq \{0,1\}^*$.

On construit des circuits $R_{q,p}(x_1, \ldots, x_n)$ pour tous $n \in \mathbb{N}$ et $q, p \in \mathbb{N}$ par récurrence sur n, indiquant si, oui ou non, on va de l'état q à l'état p dans $\mathcal{A}$ avec le mot correspondant à $x_1, \ldots, x_n$. Pour tous $q, p \in Q$, on pose

$$R_{q,p}(\varepsilon) = \begin{cases} \top & \text{si } q = p \\ \bot & \text{sinon} \end{cases}$$

ainsi que

$$R_{q,p}(x_1) = \begin{cases} \top & \text{si } \delta(q,0) = \delta(q,1) = p \\ \neg x_1 & \text{si } \delta(q,0) = p \text{ mais } \delta(q,1) \neq p \\ x_1 & \text{si } \delta(q,1) = p \text{ mais } \delta(q,0) \neq p \\ \bot & \text{sinon} \end{cases} .$$

Ensuite, pour tout $n \in \mathbb{N}, n \geq 2$ et tous $q, p \in Q$, on pose

$$R_{q,p}(x_1, \ldots, x_n) = \bigvee_{r \in Q} \left(R_{q,r}(x_1, \ldots, x_{\lceil n/2 \rceil}) \wedge R_{q,r}(x_{\lceil n/2 \rceil + 1}, \ldots, x_n) \right) .$$

Pour décider L, on utilise ensuite le circuit

$$L(x_1, \ldots, x_n) = \bigvee_{q \in F} R_{q_0,q}(x_1, \ldots, x_n)$$

pour tout $n \in \mathbb{N}$.

2 Classe NC

On note NC^i la classe des langages sur $\{0,1\}$ décidés par une famille de circuits booléens $(C_n)_{n \in \mathbb{N}}$ dont la taille est polynomiale et dont la profondeur est en $O(\log(n)^i)$, mais où l'arité des portes $\vee$ et $\wedge$ est limitée à 2.

Exercice 6. Montrer que pour tout $i \in \mathbb{N}$, on a l'inclusion $\mathsf{NC}^i \subseteq \mathsf{AC}^i \subseteq \mathsf{NC}^{i+1}$.

Solution 6. Clairement, par définition, on a $\mathsf{NC}^i \subseteq \mathsf{AC}^i$ pour tout $i \in \mathbb{N}$.

Maintenant, étant donné un circuit booléen correspondant à une famille pour AC^i avec $i \in \mathbb{N}$, on remarque que chaque porte $\wedge$ ou $\vee$ a une arité bornée par la taille du circuit qui est polynomiale en la taille d'entrée. En remplaçant chacune de ces portes par un arbre binaire équilibré des mêmes portes on obtient le résultat.

Exercice 7. Montrer que $\mathsf{NC}^0 \neq \mathsf{AC}^0$.

Solution 7. Un circuit booléen dont la profondeur est d mais où l'arité des portes $\vee$ et $\wedge$ est limitée à 2 utilise au plus 2^d variables parmi toutes celles possibles.

Il s'ensuit que tout langage L de NC^0 a la propriété qu'il existe $c \in \mathbb{N}$ vérifiant que pour tout $n \in \mathbb{N}$, il existe au plus c positions de 1 à n telles que l'appartenance d'un mot à $L \cap \{0,1\}^n$ dépende uniquement des bits à ces positions. Il vient donc en particulier que $\mathrm{ADD} \notin \mathsf{NC}^0$.

3 Classe TC^0

La classe TC^0 correspond à la classe des langages sur $\{0,1\}$ décidés par une famille de circuits booléens dont la taille est polynomiale et dont la profondeur est en $O(1)$, où les portes sont $\neg$ ainsi que des portes de seuil $\mathrm{T_k}$ d'arité non bornée pour $k \in \mathbb{N}$ quelconque, telles que pour tout $n \in \mathbb{N}$, on ait que $\mathrm{T_k}(x_1, \ldots, x_n)$ s'évalue à $\top$ quand k de ses entrées au moins s'évaluent à $\top$.

Exercice 8. Montrer que tout circuit avec des portes $\neg$ ainsi que $\wedge$, $\vee$ et MAJ d'arité non bornée, telles que pour tout $n \in \mathbb{N}$, on ait que $\mathrm{MAJ}(x_1, \ldots, x_n)$ s'évalue à $\top$ quand la moitié de ses entrées au moins (c'est-à-dire au moins $\lceil n/2 \rceil$ de celles-ci) s'évaluent à $\top$, se réécrit comme un circuit de même taille et profondeur avec des portes $\neg$ et $\mathrm{T_k}$ pour $k \in \mathbb{N}$ quelconque décidant le même langage.

Solution 8. Pour tout $n \in \mathbb{N}$, on a

$$\begin{aligned} \wedge(x_1, \ldots, x_n) &\equiv \mathrm{T_n}(x_1, \ldots, x_n) \\ \vee(x_1, \ldots, x_n) &\equiv \mathrm{T_1}(x_1, \ldots, x_n) \\ \mathrm{MAJ}(x_1, \ldots, x_n) &\equiv \mathrm{T}_{\lceil \mathrm{n}/2 \rceil}(x_1, \ldots, x_n) \ . \end{aligned}$$

Exercice 9. Montrer qu'à l'inverse, TC^0 se définit aussi comme la classe des langages sur $\{0,1\}$ décidés par une famille de circuits booléens dont la taille est polynomiale et dont la profondeur est en $\mathrm{O}(1)$, où les portes sont $\neg$ ainsi que $\wedge$, $\vee$ et MAJ d'arité non bornée.

Solution 9. Pour tous $n, k \in \mathbb{N}$, on a

$$\mathrm{T_k}(x_1, \ldots, x_n) \equiv \begin{cases} \mathrm{MAJ}(x_1, \ldots, x_n, \underbrace{\top, \ldots, \top}_{n-2k}) & \text{si } k \leq \frac{n}{2} \\ \mathrm{MAJ}(x_1, \ldots, x_n, \underbrace{\bot, \ldots, \bot}_{2k-n}) & \text{sinon} \end{cases} .$$

Notons que les portes $\vee$ et $\wedge$ sont donc inutiles.

Exercice 10. Montrer que $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

Solution 10. Pour tous $n, i \in \mathbb{N}_{>0}$, on construit des circuits $B_i(x_1, \ldots, x_n)$ et $R_i(x_1, \ldots, x_n)$ qui donnent, respectivement, le bit de poids i dans la représentation binaire du nombre de variables parmi $x_1, \ldots, x_n$ à $\top$ et l'éventuelle retenue obtenue suite à l'addition au poids i dans la somme du nombre de variables parmi $x_1, \ldots, x_{\lceil n/2 \rceil}$ à $\top$ et du nombre de variables parmi $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$ à $\top$. D'abord, pour tout $i \in \mathbb{N}$, on pose $B_i(\varepsilon) = R_i(\varepsilon) = \bot$ ainsi que

$$B_i(x_1) = \begin{cases} x_1 & \text{si } i = 0 \\ \bot & \text{sinon} \end{cases}$$

et $R_i(x_1) = \bot$. Ensuite, pour tous $i \in \mathbb{N}$ et $n \in \mathbb{N}, n \geq 2$, on pose

$$B_i(x_1, \ldots, x_n) = \begin{cases} R_{i-1}(x_1, \ldots, x_n) \oplus B_i(x_1, \ldots, x_{\lceil n/2 \rceil}) \oplus B_i(x_{\lceil n/2 \rceil + 1}, \ldots, x_n) & \text{si } i \geq 1 \\ B_0(x_1, \ldots, x_{\lceil n/2 \rceil}) \oplus B_0(x_{\lceil n/2 \rceil + 1}, \ldots, x_n) & \text{sinon} \end{cases}$$

et

$$R_i(x_1, \ldots, x_n) = \begin{cases} \big(B_i(x_1, \ldots, x_{\lceil n/2 \rceil}) \wedge B_i(x_{\lceil n/2 \rceil + 1}, \ldots, x_n)\big) \vee \\ \Big(R_{i-1}(x_1, \ldots, x_n) \wedge \big(B_i(x_1, \ldots, x_{\lceil n/2 \rceil}) \vee B_i(x_{\lceil n/2 \rceil + 1}, \ldots, x_n)\big)\Big) & \text{si } i \geq 1 \\ B_0(x_1, \ldots, x_{\lceil n/2 \rceil}) \wedge B_0(x_{\lceil n/2 \rceil + 1}, \ldots, x_n) & \text{sinon} \end{cases} .$$

On peut montrer qu'il existe un entier $\alpha \in \mathbb{N}$ tel que pour tous $n \in \mathbb{N}_{>0}$ et $i \in \mathbb{N}$, la profondeur de $B_i(x_1, \ldots, x_n)$ et $R_i(x_1, \ldots, x_n)$ soit majorée par $\alpha \cdot (\lceil \log_2(n) \rceil + i)$. À noter que la même construction peut-être généralisée pour additionner m nombres de n bits avec des circuits de taille polynomiale, profondeur logarithmique et portes $\vee$ et $\wedge$ d'arité au plus 2 (sachant qu'ici on additionne n nombres de 1 bit).

Par conséquent, étant donné $n, k \in \mathbb{N}$, on simule $\mathrm{T_k}(x_1, \ldots, x_n)$ par $\top$ si $k = 0$, par $\bot$ si $k > n$ et sinon, si $b_{l-1} \cdots b_0 \in \{0,1\}^l$ avec $l = \lceil \log_2(n+1) \rceil$ est la représentation binaire gros-boutiste de k, par

$$\bigvee_{i \in [\![0, l-1]\!], b_i = 0} \Big(\bigwedge_{j \in [\![i+1, l-1]\!], b_j = 1} B_j(x_1, \ldots, x_n) \wedge \bigwedge_{j \in [\![i+1, l-1]\!], b_j = 0} \neg B_j(x_1, \ldots, x_n) \wedge B_i(x_1, \ldots, x_n) \Big) \vee \Big(\bigwedge_{j \in [\![0, l-1]\!], b_j = 1} B_j(x_1, \ldots, x_n) \wedge \bigwedge_{j \in [\![i+1, l]\!], b_j = 0} \neg B_j(x_1, \ldots, x_n) \Big) .$$

Exercice 11. Montrer que PARITY $\in \mathsf{TC}^0$.

Solution 11. On construit des circuits $\oplus(x_1, \ldots, x_n)$ décidant PARITY $\cap \{0,1\}^n$ pour tout $n \in \mathbb{N}$, tels que

$$\oplus(x_1, \ldots, x_n) = \bigvee_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \left(\mathrm{T}_{2i+1}(x_1, \ldots, x_n) \wedge \neg \mathrm{T}_{2i+2}(x_1, \ldots, x_n) \right) .$$

Exercice 12. Montrer que MULT $\in \mathsf{TC}^0$.

Solution 12. Pour simplifier la présentation de cette solution, nous allons procéder de façon modulaire.

Passage de la multiplication de deux nombres à l'addition de n nombres On ramène facilement le problème de la multiplication de deux nombres de taille n à l'addition de n nombres de taille $2n$ car $(a \times b) = \sum_{1 \leq j \leq n} \sum_{1 \leq k \leq n} 2^{k+j}(a_j \wedge b_k)$.

Passage de l'addition de m nombres de tailles n à l'addition en parrallèle de $ln(m)$ nombres de taille $2ln(m)$. Notons $a^1, \ldots, a^m$ les nombres à additionner et $r^i = \sum_j a_i^j$. Comme r^i est la somme de m nombres valant 0 ou 1, $r^i \leq m$ et donc r^i tient sur $ln(m)$ bits. On veut calculer le bit r_j^i en profondeur constante. Notons pour cela $bit(k, j)$ qui vaut 1 si la représentation binaire de k contient un 1 en j-ème position on a :

$$r_j^i = \bigvee_{\substack{1 \leq k \leq n \\ bit(k,j)=1}} T_k(a_i^1, \ldots, a_i^m)$$

On a par ailleurs que $\sum_k a^k = \sum_i 2^i r^i$ avec :

$$\begin{array}{ccc} a_1^1 & \cdots & a_n^1 \\ \vdots & \ddots & \vdots \\ a_1^m & \cdots & a_n^m \\ \hline r_1 = \sum_k a_1^k & \cdots & r_n = \sum_k a_n^k \end{array}$$

Comme chacun des r^i tient sur $ln(m)$ bits, on va regrouper les r^i en bloc de taille $B = ln(m)$ bits en calculant

$$v^\ell = \sum_{1 \leq k \leq ln(m)} 2^k r^{k+\ell}$$

et donc on a toujours :

$$\sum_k a^k = \sum_i 2^i r^i = \sum_\ell v^\ell 2^{B\ell}$$

Nous allons montrer ensuite comment faire le calcul de v^ℓ en profondeur constante et taille polynomiale. Montrons d'abord comment s'en sortir si l'on sait calculer les v^ℓ.

Une fois que les v^ℓ sont calculés on calcule $x = \sum_\ell 2^{3B\ell} v^{3\ell}$ et $y = \sum_\ell 2^{B(3\ell+1)} v^{3\ell+1}$ et $z = \sum_\ell 2^{B(3\ell+2)} v^{2\ell+2}$. Chacun des v^ℓ tient sur $2ln(m)$ bits on a que les sommes plus hauts peuvent être transformées en OU bit à bit car les représentations binaires de ces nombres sont disjointes (entre $2^{B\ell} v^\ell$ et $2^{B(\ell+3)} v^{\ell+3}$ il n'y a pas de bits en commun car $v^\ell \leq 2^{2B}$).

Enfin la dernière étape est de faire la somme $x + y + z$ mais cela peut être fait en profondeur constante puisque la somme est dans $\mathsf{AC}^0 \subseteq \mathsf{TC}^0$.

Calcul des v^ℓ en profondeur constante Ce que l'on a envie de faire pour calculer v^ℓ c'est d'utiliser sa table de vérité. Le problème c'est que chaque bit de v^ℓ dépend de $ln(m) \times ln(m)$ bits des r^i. Donc la table de vérité peut être exponentielle en $ln(m) \times ln(m)$ ce qui est plus que polynomial en m.

Le calcul d'un v^ℓ correspond donc à faire la somme de $ln(m)$ nombres de taille $2ln(m)$. En itérant *une seule fois* le processus qui nous a permis de passer d'une somme de a^k à une somme de v^ℓ, on se ramène à écrire $v^\ell = \sum 2^{B'j} w^k$. Où les w^k sont de taille logarithmique en la taille v^ℓ et donc $|w^k| \leq 2ln(ln(m)^2) \leq 4ln(ln(m))$

Maintenant les w^k sont de taille très petite par rapport à n et donc le calcul de chaque w^k peut se faire en profondeur constante en utilisant sa table de vérité (comme à la question 2). Passer par la table de vérité produit bien un circuit de profondeur constante mais de taille exponentielle. Cependant cette taille est exponentielle en le nombre de bits dont dépend w^k qui ici est $4ln(ln(m)) \times ln(ln(m))$ et donc sub-linéaire en m. Notez qu'il y a un nombre polynomial de w^k car il y a $ln(m)/ln(ln(m))$ par v^ℓ et qu'il y a $m/ln(m)$ v^ℓ par a^k.

Une fois que les w^k sont calculés on passe des w^k aux v^ℓ comme nous sommes passés des v^ℓ aux a^k en en sommant un tous les trois, ce qui nous donne des nombres x', y' et z' et l'on fait la somme $x' + y' + z'$.