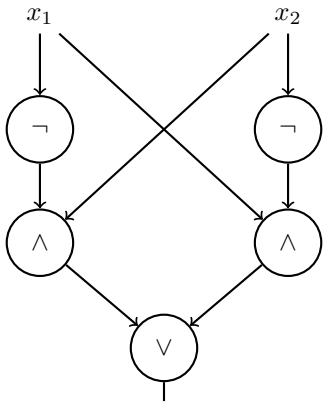
Calcul parallèle et ses limitations

1 Circuits

Un *circuit (logique)* est un graphe dirigé acyclique qui satisfait ces propriétés:

- Les sommets d'entrance zéro sont les sommets dits d'*entrée*;
- Les sommets de sortance zéro sont les sommets dits de *sortie*;
- Les sommets internes d'entrance un sont étiquetés par la porte logique $\neg$;
- Les sommets internes d'entrance deux ou plus sont étiquetés par les portes logiques $\{\wedge, \vee\}$;
- Les n sommets d'entrée sont étiquetés respectivement par des variables $x_1, \ldots, x_n$.

Un circuit avec n entrées et m sorties calcule naturellement une fonction $f\colon \{0,1\}^n \to \{0,1\}^m$. Par exemple, le circuit ci-dessous calcule $f(x_1, x_2) \coloneqq x_1 \oplus x_2$, où $\oplus$ dénote le OU exclusif:



La direction des arêtes est rarement indiquée dans les diagrammes; on suppose que la direction va du haut vers le bas.

La *taille* d'un circuit est son nombre de portes, par ex. 5 pour le circuit ci-dessus. La profondeur d'un circuit est la quantité maximale de portes sur un chemin allant d'une entrée vers une sortie, par ex. 3 pour le circuit

ci-dessus. Plus formellement, la *profondeur* d'un sommet est définie par ces règles récursives:

$$\begin{aligned}
\mathrm{prof}(x_i) &\coloneqq 0,\\
\mathrm{prof}(\vee_{1\le i\le n}\, v_i) &\coloneqq \max(\mathrm{prof}(v_1),\dots,\mathrm{prof}(v_n))+1,\\
\mathrm{prof}(\wedge_{1\le i\le n}\, v_i) &\coloneqq \max(\mathrm{prof}(v_1),\dots,\mathrm{prof}(v_n))+1,\\
\mathrm{prof}(\neg v) &\coloneqq \mathrm{prof}(v)+1.
\end{aligned}$$

La *profondeur* d'un circuit C est définie par $\mathrm{prof}(C) \coloneqq \max\{\mathrm{prof}(u) : u \text{ est un sommet de sortie de } C\}$.

Nous interdirons parfois les portes d'entrance arbitraire, c.-à-d. les sommets avec plus de deux prédecesseurs. Il est possible de remplacer une porte d'entrance arbitraire par une cascade de portes équivalentes; par exemple, $\bigvee_{1\le i\le 8} x_i$ devient

$$\Big(\big(x_1 \vee x_2\big) \vee \big(x_3 \vee x_4\big)\Big) \vee \Big(\big(x_5 \vee x_6\big) \vee \big(x_7 \vee x_8\big)\Big).$$

Lorsque l'entrée est de taille impaire, nous découpons environ en deux. Plus formellement, étant donné une porte $\circ \in \{\vee, \wedge\}$, nous utilisons cette réécriture récursive:

$$\mathrm{casc}(\bigcirc_{1\le i\le n}\, x_i) \coloneqq \begin{cases} x_1 & \text{si } n = 1,\\ x_1 \circ x_2 & \text{si } n = 2,\\ \big(\mathrm{casc}(\bigcirc_{1\le i\le n\div 2}\, x_i)\big) \circ \big(\mathrm{casc}(\bigcirc_{n\div 2<i\le n}\, x_i)\big) & \text{si } n > 2. \end{cases}$$

Cette transformation crée un circuit de taille linéaire et de profondeur logarithmique:

Proposition 1. *Soit* $u \coloneqq \mathrm{casc}(\bigcirc_{1\le i\le n}\, x_i)$ *où* $\circ \in \{\vee, \wedge\}$. *Nous avons* $\mathrm{taille}(u) = n - 1$ *et* $\mathrm{prof}(u) \le \lceil \log n \rceil$.

2 Classes de complexité

Contrairement aux machines de Turing ou aux programmes, un circuit ne peut gérer qu'une seule taille d'entrée. Ainsi, pour accomplir une tâche calculatoire, il faut utiliser une famille de circuits, c'est-à-dire un circuit pour chaque taille d'entrée.

Soit $i \in \mathbb{N}$. Nous disons qu'une fonction $f\colon \{0,1\}^* \to \{0,1\}^*$ appartient à la classe de complexité FAC^i s'il existe une famille de circuits $\{C_0, C_1, \dots\}$ et une constante $c \in \mathbb{N}$ telles que

- — $\mathrm{taille}(C_n) \in \mathcal{O}(n^c)$,
- — $\mathrm{prof}(C_n) \in \mathcal{O}(\log^i n)$ où $\log^i n$ dénote $(\log n)^i$,
- — $C_n(x) = f(x)$ pour toute entrée $x \in \{0,1\}^n$.

La classe FNC^i est définie de la même façon, mais en interdisant les portes d'entrance arbitraire. Les classes AC^i et NC^i sont définies respectivent comme FAC^i et FNC^i mais en ne permettant qu'un seul bit en sortie; autrement dit, la fonction calculée doit être de la forme $f\colon \{0,1\}^* \to \{0,1\}$. Ainsi, AC^i et NC^i sont des classes de problèmes de décision.

Posons

$$\mathsf{AC} \coloneqq \bigcup_{i\in\mathbb{N}} \mathsf{AC}^i \text{ et } \mathsf{NC} \coloneqq \bigcup_{i\in\mathbb{N}} \mathsf{NC}^i.$$

En mots, un problème de décision appartient à AC s'il existe une famille de circuits de taille polynomiale et de profondeur polylogarithmique qui calcule f.

Comme la taille d'un circuit correspond informellement au nombre d'unités de calcul, et que la profondeur d'un circuit correspond au temps d'exécution, la classe NC est considérée comme l'ensemble des problèmes de décision qui peuvent être résolus efficacement en parallèle.

Par la proposition 1, si une porte possède n^c entrées, alors elle peut être remplacée par un sous-circuit équivalent de taille $n^c - 1$ et de profondeur au plus $\lceil \log(n^c) \rceil = \lceil c \log n \rceil \in \mathcal{O}(\log n)$. Ainsi, si un circuit de taille polynomiale et de profondeur $\mathcal{O}(\log^i n)$ possède des portes d'entrance arbitraire, ce circuit peut être converti en un circuit équivalent de taille polynomiale et de profondeur $\mathcal{O}(\log(n) \cdot \log^i(n)) = \mathcal{O}(\log^{i+1} n)$.

Par conséquent, $\mathsf{AC}^i \subseteq \mathsf{NC}^{i+1}$ pour tout $i \in \mathbb{N}$, ce qui implique $\mathsf{AC} = \mathsf{NC}$ car

$$\mathsf{NC}^0 \subseteq \mathsf{AC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{AC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{AC}^2 \subseteq \cdots .$$

Ainsi, pour montrer l'appartenance à NC, il suffit de raisonner sur AC sans se soucier des portes d'entrance arbitraire.

3 Addition

Considérons le problème calculatoire qui consiste à additionner deux entiers naturels:

ADD

ENTRÉE: deux entiers $a, b \geq 0$ décrits en binaire avec n bits,
SORTIE: $a + b$ décrit en binaire avec $n + 1$ bits.

Nous allons montrer que ce problème appartient à FAC^0 et ainsi qu'il est parallélisable.

3.1 Somme de deux bits

Supposons que l'on souhaite additionner les bits a et b. La somme peut être représentée sur deux bits $c_1 c_0$ puisque la somme varie entre 0 et 2 inclusivement. La table de vérité est comme suit:

a	b	c_1	c_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

En inspectant la table, on remarque que $c_1 = a \wedge b$ et $c_0 = a \oplus b$. Rappelons que « $a \oplus b$ » n'est pas directement permis, mais peut être émulé par $(a \wedge \neg b) \vee (\neg a \wedge b)$.

3.2 Somme de trois bits

Supposons que l'on souhaite additionner les bits a, b et r. La somme peut être représentée sur deux bits $c_1 c_0$ puisque la somme varie entre 0 et 3 inclusivement. La table de vérité est comme suit:

a	b	r	c_1	c_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

En inspectant la table, on remarque que $c_1 = 1$ ssi $\{a, b, r\}$ contient au moins deux occurrences de 1, et que c_2 est la parité du nombre d'occurrences de 1. Ainsi, $c_1 = (a \wedge b) \vee ((a \vee b) \wedge r)$ et $c_0 = (a \oplus b) \oplus r$.

Rappelons que « $(a \oplus b) \oplus r$ » n'est pas directement permis, mais peut être émulé par

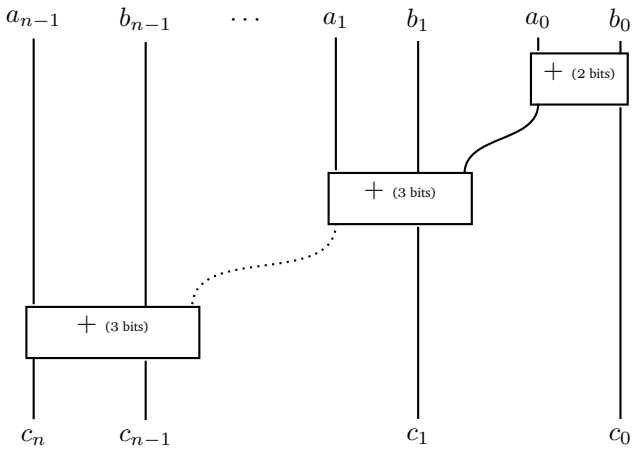
$$(y \wedge \neg r) \vee (\neg y \wedge r) \text{ où } y = (a \wedge \neg b) \vee (\neg a \wedge b).$$

3.3 Somme de deux nombres: approche séquentielle

Nous pouvons additionner deux entiers binaires $a_{n-1} \cdots a_1 a_0$ et $b_{n-1} \cdots b_1 b_0$ en additionnant séquentiellement leurs bits, du poids faible vers le poids fort. Plus formellement, en réutilisant les identités établies, nous avons

$$c_0 \coloneqq a_0 \oplus b_0, \qquad c_i \coloneqq (a_i \oplus b_i) \oplus r_{i-1}, \qquad c_n \coloneqq r_{n-1},$$
$$r_0 \coloneqq a_i \wedge b_i, \qquad r_i \coloneqq (a_i \wedge b_i) \vee ((a_i \vee b_i) \wedge r_{i-1}).$$

Cela mène à ce circuit C_n:



Analysons la taille de C_n. La portion qui calcule $a_0 + b_0$ possède six portes. La portion qui calcule $a_i + b_i$, où $i > 0$, possède quatorze portes. Ainsi, taille$(C_n) = 6 + 14(n-1) = 14n - 8 \in \Theta(n)$. Comme chaque retenue r_i dépend de la retenue précédente r_{i-1}, la profondeur de C_n est linéaire. En effet, la proposition suivante montre que prof$(C_n) \in \Theta(n)$:

Proposition 2. *Nous avons* prof$(r_i) = 2i + 1$ *et* prof$(c_i) = \max(2i + 2, 1)$ *pour tout* $0 \leq i < n$.

3.4 Somme de deux nombres: approche parallèle déraisonnable

Il est possible de calculer la somme de deux nombres par force brute en figeant la table d'addition dans le circuit. Cela mène à un circuit de profondeur constante, mais de taille au moins exponentielle comme il y a $2^{n+n} = 2^{2n} = 4^n$ lignes à la table. Techniquement, cela mène à un temps parallèle constant, mais, comme la taille correspond informellement au nombre d'unités de calcul, cela est déraisonnable. Formellement, un tel circuit ne satisfait pas les constraintes de FAC^0 en raison de la taille non-polynomiale du circuit. Nous verrons donc une façon plus ingénieuse d'obtenir une profondeur constante.

3.5 Somme de deux nombres: approche parallèle raisonnable

Rappelons les identités de la section 3.3:

$$\begin{aligned} c_0 &= a_0 \oplus b_0, & c_i &= (a_i \oplus b_i) \oplus r_{i-1}, & c_n &\coloneqq r_{n-1}, \\ r_0 &= a_i \wedge b_i, & r_i &= (a_i \wedge b_i) \vee ((a_i \vee b_i) \wedge r_{i-1}). \end{aligned}$$

Nous pouvons exprimer chaque retenue de telle sorte à ce que sa valeur ne dépende plus des retenues précédentes:

$$\begin{aligned} c'_0 &\coloneqq a_0 \oplus b_0, & c'_i &\coloneqq (a_i \oplus b_i) \oplus r'_{i-1}, & c'_n &\coloneqq r'_{n-1}, \\ r'_0 &\coloneqq a_i \wedge b_i, & r'_i &\coloneqq \bigvee_{i \geq j \geq 0} \left((a_j \wedge b_j) \wedge \bigwedge_{i \geq k > j} (a_k \vee b_k) \right). \end{aligned}$$

Voyons d'abord informellement pourquoi nous avons choisi ces nouvelles identités pour les retenues. La somme de a_i, b_i et r'_{i-1} mène à une retenue ssi nous avons $a_i = b_i = 1$, ou $(a_i \vee b_i) = 1$ et $r'_{i-1} = 1$. Similairement, $r'_{i-1} = 1$ ssi $a_{i-1} = b_{i-1} = 1$, ou $(a_{i-1} \vee b_{i-1}) = 1$ et $r'_{i-2} = 1$. Comme on ne peut pas dérouler ce raisonnement à l'infini, on doit forcément s'arrêter à un certain indice j où $a_j = b_j = 1$. Graphiquement, $r'_i = 1$ ssi nous avons quelque chose du genre:

$$\begin{array}{cccccccccc} & i & & \longleftarrow & k & \longrightarrow & & j & \\ & 0 & 1 & 0 & 0 & 1 & \cdots & 1 & \cdots \\ + & 1 & 0 & 1 & 1 & 0 & \cdots & 1 & \cdots \\ \hline \end{array}$$

Formellement, la nouvelle identité ne change en effet pas les valeurs calculées:

Proposition 3. *Nous avons $r_i = r'_i$ pour tout $0 \leq i < n$, et $c_i = c'_i$ pour tout $0 \leq i \leq n$.*

Analysons la taille de C'_n, sans se soucier des « $\oplus$ » qui ne changent la taille et la profondeur que d'un facteur constant. Il y a deux portes pour le calcul de c'_0 et r'_0. Le nombre de portes pour le calcul de r'_i est borné par

$$\sum_{i \geq j \geq 0} \left(3 + \sum_{i \geq k > j} 2 \right) \in \mathcal{O}(n^2).$$

Le nombre de portes pour le calcul de c'_i est égal à celui pour r'_i plus deux, donc aussi de $\mathcal{O}(n^2)$. Par conséquent, taille$(C'_n) \in \mathcal{O}(2 + n \cdot n^2) = \mathcal{O}(n^3)$. Remarquons que cette analyse se raffine. En effet, les termes de la forme $\bigwedge_{i \geq k > j} (a_k \vee b_k)$ peuvent être partagés par $r'_1, \ldots, r'_n$ sans être entièrement recalculés par chaque r'_i. Une telle analyse mène à taille$(C'_n) \in \mathcal{O}(n^2)$, bien que la complexité exacte n'importe pas dans notre contexte.

Analysons maintenant la profondeur du circuit. Il découle directement des définitions que $\text{prof}(c'_0) = \text{prof}(r'_0) = 1$, $\text{prof}(r'_i) = 4$ et $\text{prof}(c'_i) = 5$.

Comme la taille est polynomiale et que la profondeur est constante, nous venons de montrer ce résultat:

Théorème 1. $\mathsf{ADD} \in \mathsf{FAC}^0$.

En mots, ce résultat signifie qu'il est possible d'additionner deux nombres en temps parallèle constant.

4 Accessibilité

Considérons le problème d'accessibilité dans les graphes dirigés:

PATH

ENTRÉE:	un graphe dirigé $G = (V, E)$ décrit par une matrice d'adjacence, et deux sommets $s, t \in V$,
QUESTION:	existe-t-il un chemin de s vers t dans G?

Dans cette section, nous allons montrer que ce problème appartient à $\mathsf{AC}^1 \subseteq \mathsf{NC}^2$ et ainsi qu'il est parallélisable.

Étant donné un graphe $G = (V, E)$ et deux sommets $u, v \in V$, nous définissons $\mathbf{A}^\ell[u, v]$ comme suit:

$$\mathbf{A}^0[u, v] \coloneqq 1 \text{ si } u = v, 0 \text{ sinon,}$$

$$\mathbf{A}^1[u, v] \coloneqq 1 \text{ si } u = v \text{ ou } (u, v) \in E, 0 \text{ sinon,}$$

$$\mathbf{A}^\ell[u, v] \coloneqq \bigvee_{x \in V} \left(\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, x] \wedge \mathbf{A}^{\lceil \ell/2 \rceil}[x, v] \right).$$

Proposition 4. *Il existe un chemin de u vers v de longueur au plus ℓ ssi $\mathbf{A}^\ell[u, v] = 1$.*

Démonstration. Nous procédons par induction sur $\ell \in \mathbb{N}$. Les cas où $\ell \in \{0, 1\}$ découlement trivialement de la définition de $\mathbf{A}$. Soit $\ell > 1$.

$\Rightarrow$) Supposons qu'il existe un chemin de u vers v de longueur au plus ℓ. En particulier, il existe un chemin de longueur au plus $\lfloor \ell/2 \rfloor$ de u vers un certain sommet w, et un chemin de longueur au plus $\lceil \ell/2 \rceil$ de w vers v. Par hypothèse d'induction, nous avons $\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, w] = 1$ et $\mathbf{A}^{\lceil \ell/2 \rceil}[w, v] = 1$. Rappelons que

$$\mathbf{A}^\ell[u, v] = \bigvee_{x \in V} \left(\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, x] \wedge \mathbf{A}^{\lceil \ell/2 \rceil}[x, v] \right).$$

Comme $\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, w] \wedge \mathbf{A}^{\lceil \ell/2 \rceil}[w, v]$ est vrai, nous avons forcément $\mathbf{A}^\ell[u, v] = 1$.

$\Leftarrow$) Supposons que $\mathbf{A}^\ell[u, v] = 1$. Par définition, nous avons

$$\mathbf{A}^\ell[u, v] = \bigvee_{x \in V} \left(\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, x] \wedge \mathbf{A}^{\lceil \ell/2 \rceil}[x, v] \right) = 1.$$

Il existe donc $x \in V$ tel que $\mathbf{A}^{\lfloor \ell/2 \rfloor}[u, x] = 1$ et $\mathbf{A}^{\lceil \ell/2 \rceil}[x, v]$. Par hypothèse d'induction, il existe un chemin de longueur au plus $\lfloor \ell/2 \rfloor$ de u vers x, et un chemin de longueur au plus $\lceil \ell/2 \rceil$ de x vers v. Par conséquent, il existe un chemin de longueur au plus $\lfloor \ell/2 \rfloor + \lceil \ell/2 \rceil = \ell$ de u vers v. □

Remarquons que s'il existe un chemin entre deux sommets, alors il en existe forcément un qui est simple (c.-à-d. sans répétitions). En effet, il est toujours possible de retirer les cycles d'un chemin afin d'en obtenir un plus court. Cette observation implique ce lemme:

Lemme 1. *Soit $G = (V, E)$ un graphe dirigé. S'il existe un chemin de u vers v dans G, alors il existe un chemin de u vers v de longueur au plus $2^{\lceil \log |V| \rceil}$.*

Nous sommes maintenant prêt à montrer que le problème d'accessibilité se parallélise.

Théorème 2. PATH $\in$ AC^1.

Démonstration. Soit $G = (V, E)$ le graphe le graphe dirigé en entrée décrit par une matrice d'adjacence **B**, et soient $s, t \in V$ les sommets en entrée. En combinant la proposition 4 et le lemme 1, nous remarquons qu'il existe un chemin de s vers t ssi $\mathbf{A}^k[s,t] = 1$ où $k \coloneqq 2^{\lceil |V| \rceil}$. Il suffit donc de donner un circuit qui calcule $\mathbf{A}^k[s,t]$.

Définissons un sommet $w_{i,u,v}$ du circuit qui calcule $\mathbf{A}^{2^i}[u,v]$. Posons $w_{0,u,v} \coloneqq 1$ si $u = v$, et $w_{0,u,v} \coloneqq \mathbf{B}[u,v]$ sinon. De plus, posons

$$w_{i,u,v} \coloneqq \bigvee_{x \in V} \left(w_{i-1,u,x} \wedge w_{i-1,x,v} \right).$$

La sortie du circuit global est $w_{\lceil \log |V| \rceil, s, t}$. Par définition, le circuit détermine correctement s'il existe un chemin de s vers t dans G.

Analysons la taille du circuit. Remarquons que la taille n de l'entrée appartient à $\Theta(|V|^2 + 2\log|V|) = \Theta(|V|^2)$ puisque l'entrée est formée d'une matrice d'adjacence et de l'identifiant de deux sommets. Chaque sommet $w_{i,u,v}$ crée deux portes: une conjonction d'entrance deux, et une disjonction d'entrance arbitraire. Au total, le nombre de portes est donc d'au plus

$$2 \cdot \lceil \log |V| \rceil \cdot |V| \cdot |V| \in \mathcal{O}(|V|^2 \log |V|) \subseteq \mathcal{O}(n \log n).$$

Analysons maintenant la profondeur du circuit. Il n'y a que deux portes entre deux sommets de la forme $w_{i,*,*}$ et $w_{i-1,*,*}$. Comme i varie de 1 à $\lceil \log |V| \rceil$, la profondeur du circuit appartient à

$$\mathcal{O}(\log |V|) \subseteq \mathcal{O}(\log \sqrt{n}) = \mathcal{O}(\log(n^{1/2})) = \mathcal{O}(1/2 \log n) = \mathcal{O}(\log n).$$

Comme le circuit est de taille polynomiale et de profondeur logarithmique, nous obtenons PATH $\in$ AC^1. □

5 Uniformité

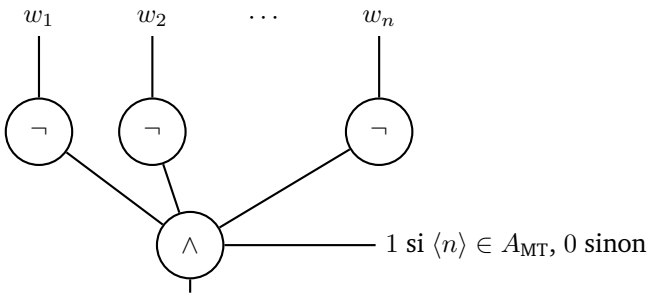
Rappelons qu'afin de montrer l'appartenance à AC ou NC, nous utilisons une *famille* de circuits. Dans nos exemples précédents, tous les circuits d'une même famille étaient similaires et relativement simples à produire. Or, techniquement, la définition permet de « tricher » en choisissant $\{C_0, C_1, C_2, \ldots\}$ de telle sorte à ce que les circuits soient très différents ou difficilement calculables.

En fait, pour des raisons pédagogiques, nous n'avons volontairement pas vu la dernière contrainte de la vraie définition de AC et NC: il faut que les circuits soient uniformes, ce qui informellement signifie qu'il est possible et simple d'obtenir le $n^{\text{ème}}$ circuit de la famille.

Par exemple, posons

$$L \coloneqq \{0^n : n, \text{ écrit en binaire, est le code d'une machine de Turing qui s'arrête sur son propre code}\}.$$

Nous savons que L est indécidable. Or, selon la définition de la section 2, le langage L appartient à AC^0 car cette famille de circuits décide L:



La famille de circuits ci-dessus existe mais elle n'est pas constructive: bien que le « bit magique » existe, il est impossible de le produire algorithmiquement.

Formellement, nous disons qu'une famille de circuits $\{C_0, C_1, \ldots\}$ est *uniforme* s'il existe une machine de Turing qui, étant donné n décrit en unaire, produit C_n. Cette définition n'est toujours pas satisfaisante. En effet, s'il était permis de passer un temps exponentiel à construire C_n, alors la haute complexité serait dissimulée. Nous restreignons donc les ressources permises pour qu'elles soient très limitées. Nous disons qu'une famille de circuits $\{C_0, C_1, \ldots\}$ est *log-uniforme* s'il existe une machine de Turing qui, étant donné n décrit en unaire, produit C_n en espace logarithmique.

Clarifions ce que nous entendons par « espace logarithmique ». En effet, cette contrainte est en apparence trop forte car l'entrée est de taille n, et qu'il faut la lire en entier pour espérer produire quelque chose de pertinent. En fait, dans ce contexte, on suppose que la machine de Turing possède plusieurs rubans:

— Un ruban d'entrée en lecture seule;

— Un nombre constant de rubans en lecture/écriture qui ne peuvent excéder une taille de $\mathcal{O}(\log n)$.

Remarquons qu'une telle machine fonctionne forcément en temps polynomial. En effet, chaque case des rubans auxiliaires contient une lettre de l'alphabet de ruban Γ. Comme la taille des rubans auxiliaires est bornée par $c \log n$ pour une certaine constante c, la machine possède au plus $|\Gamma|^{c \log n}$ configurations différentes. Cette valeur est polynomiale puisque

$$|\Gamma|^{c \log n} = (2^{\log |\Gamma|})^{c \log n} = 2^{\log |\Gamma| \cdot c \log n} = 2^{\log(n^{\log |\Gamma| \cdot c})} = n^{\log |\Gamma| \cdot c}.$$

Ainsi, une famille de circuit est log-uniforme ssi son $n^{\text{ème}}$ circuit peut être produit en temps polynomial et en espace logarithmique.

Les véritables définitions de FAC^i, FNC^i, AC^i, NC^i et $\mathsf{AC} = \mathsf{NC}$ demandent à ce que la famille de circuit soit log-uniforme. Tous les résultats établis jusqu'ici satisfont cette contrainte supplémentaire.

6 NC vs. P

Remarquons que les problèmes qui peuvent être résolus efficacement en parallèle peuvent être résolus efficacement en séquentiel:

Théorème 3. NC $\subseteq$ P.

Démonstration. Soit $L \in$ NC. Montrons que $L \in$ P en donnant une machine de Turing qui décide L en temps polynomial. Sur entrée $w \in \{0,1\}^n$, nous générons le $n^{\text{ème}}$ circuit C_n de la famille. Par définition de NC, il est possible d'y arriver en espace logarithmique, et donc en temps polynomial. On assigne ensuite w aux sommets d'entrées de C_n, puis on évalue C_n en le parcourant en ordre topologique. On accepte w ssi C_n évalue à 1. □

À ce jour, nous ne savons pas si P $\subseteq$ NC et ainsi si NC $=$ P. Il existe donc une théorie de la complétude comme pour P vs. NP. Formellement, nous disons qu'un langage L est P-complet ssi

— $L \in$ P, et

— $L' \in$ P $\implies L' \leq_m^{\log} L$ (tout langage de P se réduit à L en espace logarithmique).

L'analogue de SAT en P-complétude est le problème d'évaluation de circuits:

CVP

ENTRÉE: un circuit C et une entrée x,
SORTIE: C évalue à 1 sur entrée x?

Théorème 4. CVP *est* P*-complet.*

Démonstration. Le problème CVP appartient à P puisqu'il est possible d'évaluer C sur entrée x en l'évaluant progressivement en ordre toplogique. Il nous reste donc à montrer que CVP est P-difficile.

Soit L un langage qui appartient à P. Nous devons montrer que L se réduit à CVP. Puisque $L \in$ P, il existe une machine de Turing $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ qui décide L en temps au plus n^c, où c est une constante. Nous allons montrer comment simuler $\mathcal{M}$ sur une entrée $w \in \{0,1\}^n$ à l'aide d'un circuit C. Nous décrivons C en plusieurs étapes.

Codage. Chaque élément d'une configuration de $\mathcal{M}$ appartient à $\Gamma \cup (Q \times \Gamma)$. Nous représentons un tel élément avec un codage *one-hot*, par ex. si $\Gamma = \{0, 1, \sqcup\}$ et $Q = \{q_0, q_1\}$, alors « $q_0 1$ » est représenté par 0000**1**0000:

0	1	$\sqcup$	$q_0\,0$	$q_0\,1$	$q_0\,\sqcup$	$q_1\,0$	$q_1\,1$	$q_1\,\sqcup$
0	0	0	0	1	0	0	0	0

Les configurations atteintes par $\mathcal{M}$ sur entrée w ne peuvent excéder une taille de n^c. Ainsi, toute configuration se représente avec n^c telles suites de bits.

Organisation. Le circuit C est organisé sous forme de « grille ». Le but de la $i^{\text{ème}}$ ligne du circuit est d'indiquer la configuration obtenue après i étapes de l'exécution de la machine $\mathcal{M}$. Par exemple, si $\Gamma = \{0, 1, \sqcup\}$, $Q = \{q_0, q_1\}$ et $w = 101$, alors le premier étage reçoit en entrée une suite de bits w' qui représente la configuration initiale $q_0 1\,0\,1\,\sqcup \cdots \sqcup$:

$$w' = 000010000\ 100000000\ 010000000\ 001000000 \cdots 001000000.$$

Chaque autre ligne du circuit est calculée par rapport à la précédente. Voyons comment. Soit $u_{i,j,x}$ le sommet à la ligne i et colonne j du circuit qui indique si la $j^{\text{ème}}$ position de la $i^{\text{ème}}$ configuration contient l'élément $x \in \Gamma \cup (Q \times \Gamma)$.

Implémentation. Pour un élément de la forme $qb \in Q \times \Gamma$, nous définissons le sommet comme suit:

$$u_{i,j,qb} \coloneqq u_{i-1,j,b} \wedge \left(\bigvee_{(p,a)\to(q,*,D)\in\delta} (j > 0 \wedge u_{i-1,j-1,pa}) \vee \bigvee_{(p,a)\to(q,*,G)\in\delta} (j < n^c \wedge u_{i-1,j+1,pa}) \right).$$

En mots, l'expression ci-dessus affirme que la tête est en ce moment à la position j dans l'état q avec la lettre b ssi à l'étape précédente b était à la position j sans la tête, et

— la tête était à la position $j-1$ dans l'état p avec la lettre a, et la machine a exécuté une transition de la forme $(p,a) \to (q,*,D)$; ou

— la tête était à la position $j+1$ dans l'état p avec la lettre a, et la machine a exécuté une transition de la forme $(p,a) \to (q,*,G)$.

Pour un élément de la forme $b \in \Gamma$, nous définissons le sommet comme suit:

$$u_{i,j,b} \coloneqq (u_{i-1,j,b} \wedge \bigwedge_{q\in Q} \neg u_{i,j,qb}) \vee \bigvee_{(p,a)\to(*,b,*)\in\delta} u_{i-1,j,pa}.$$

En mots, l'expression ci-dessus affirme qu'en ce moment, la $j^{\text{ème}}$ case du ruban contient la lettre b, avec la tête ailleurs, ssi

— à l'étape précédente, b était déjà là, avec la tête ailleurs, et aucun déplacement n'a amené la tête; ou

— à l'étape précédente, la tête était là dans un état p avec une certaine lettre a, et la machine a exécuté une transition qui a écrasé a par b, puis s'est déplacée (à droite ou gauche).

Acceptation. Afin de vérifier si la machine $\mathcal{M}$ accepte w, le circuit vérifie si l'état d'acceptation a été atteint au dernier étage en inspectant chaque colonne. Le sommet v de sortie du circuit est donc défini comme suit:

$$v \coloneqq \bigvee_{1\le j\le n^c} \bigvee_{a\in\Gamma} u_{n^c,j,q_{\text{acc}}a}.$$

Fin de la preuve. Clairement, le circuit C et son entrée w' sont constructibles en temps polynomial à partir de $\mathcal{M}$ et w. Avec un peu plus de travail, on pourrait argumenter que cela se fait en espace logarithmique. De plus, $w \in L$ ssi $\mathcal{M}$ accepte w ssi $C(w') = 1$. Ainsi, $L \le_m^{\log}$ CVP. □

Notons qu'il est également connu que la programmation linéaire est P-complète. Dans sa variante décisionnelle, ce problème consiste à déterminer s'il existe une solution $\boldsymbol{x} \in \mathbb{R}^k$ à un système d'inéquations $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$, où $\mathbf{A} \in \mathbb{Z}^{m\times k}$ et $\boldsymbol{b} \in \mathbb{Z}^m$. Ainsi, bien qu'il soit possible de résoudre de tels systèmes en temps polynomial, nous ne savons toujours pas s'il est possible de tirer profit du parallélisme pour faire mieux!

Remarquons que, comme en théorie de la NP-complétude, si $A \le_m^{\log} B$ et A est P-difficile, alors B est P-difficile. De plus, si on arrive à montrer qu'un problème P-complet appartient à NC, alors NC = P; et si on arrive à montrer qu'un problème P-complet n'appartient pas à NC, alors NC ≠ P.

Annexe: preuves omises

Proposition 1. *Soit* $u \coloneqq \mathsf{casc}(\bigcirc_{1\le i\le n} x_i)$ *où* $\circ \in \{\vee, \wedge\}$. *Nous avons* $\mathsf{taille}(u) = n-1$ *et* $\mathsf{prof}(u) \le \lceil \log n \rceil$.

Démonstration. Nous procédons par induction sur le nombre de termes. La proposition est trivialement vraie lorsque $n \le 2$. Pour $n > 2$, nous avons:

$$\begin{aligned}
&\mathsf{taille}(\mathsf{casc}(\bigcirc_{1\le i\le n} x_i)) \\
&= \mathsf{taille}(\mathsf{casc}(\bigcirc_{1\le i\le n\div 2} x_i)) + \mathsf{taille}(\mathsf{casc}(\bigcirc_{n\div 2<i\le n} x_i)) + 1 && \text{(par déf.)} \\
&= (\lfloor n/2 \rfloor - 1) + (\lceil n/2 \rceil - 1) + 1 && \text{(par induction)} \\
&= n - 1.
\end{aligned}$$

De plus,

$$\begin{aligned}
&\mathsf{prof}(\mathsf{casc}(\bigcirc_{1\le i\le n} x_i)) \\
&= \max\left(\mathsf{prof}(\mathsf{casc}(\bigcirc_{1\le i\le n\div 2} x_i)), \mathsf{prof}(\mathsf{casc}(\bigcirc_{n\div 2<i\le n} x_i))\right) + 1 && \text{(par déf.)} \\
&\le \max(\lceil \log \lfloor n/2 \rfloor \rceil, \lceil \log \lceil n/2 \rceil \rceil) + 1 && \text{(par induction)} \\
&= \lceil \log \lceil n/2 \rceil \rceil + 1 \\
&= \lceil \log \lceil n/2 \rceil + 1 \rceil \\
&= \lceil \log \lceil n/2 \rceil + \log 2 \rceil \\
&= \lceil \log 2 \lceil n/2 \rceil \rceil \\
&= \lceil \log n \rceil.
\end{aligned}$$

□

Proposition 2. *Nous avons* $\mathsf{prof}(r_i) = 2i+1$ *et* $\mathsf{prof}(c_i) = \max(2i+2, 1)$ *pour tout* $0 \le i < n$.

Démonstration. Nous procédons par induction sur i. Le cas de base est trivial. Soit $i \ge 1$. Nous avons

$$\begin{aligned}
\mathsf{prof}(r_i) &= \mathsf{prof}((a_i \wedge b_i) \vee ((a_i \vee b_i) \wedge r_{i-1})) \\
&= \max(\mathsf{prof}(a_i \wedge b_i), \mathsf{prof}((a_i \vee b_i) \wedge r_{i-1})) + 1 \\
&= \max(1, \max(\mathsf{prof}(a_i \vee b_i), \mathsf{prof}(r_{i-1})) + 1) + 1 \\
&= \max(1, \max(1, \mathsf{prof}(r_{i-1})) + 1) + 1 \\
&= \max(2, 3, \mathsf{prof}(r_{i-1}) + 2) \\
&= \max(2, 3, 2(i-1)+3) && \text{(par induction)} \\
&= 2(i-1) + 3 && \text{(car } i \ge 1\text{)} \\
&= 2i + 1.
\end{aligned}$$

Afin de compléter la preuve, remarquons que $\mathsf{prof}(c_0) = 1$, et que, pour tout $i > 0$, nous avons

$$\mathsf{prof}(c_i) = \max(1, \mathsf{prof}(r_i)) + 1 = \max(1, 2i+1) + 1 = \max(2, 2i+2) = 2i+2.$$

□

Proposition 3. *Nous avons* $r_i = r'_i$ *pour tout* $0 \le i < n$, *et* $c_i = c'_i$ *pour tout* $0 \le i \le n$.

Démonstration. Il suffit de montrer que $r_i = r'_i$, puisque $c_i = c'_i$ en découle directement. Nous avons triviale-

ment $r_0 = r'_0$. Soit $i > 0$. Nous avons

$$\begin{aligned}
r_i &= (a_i \wedge b_i) \vee ((a_i \vee b_i) \wedge r_{i-1}) \\
&= (a_i \wedge b_i) \vee ((a_i \vee b_i) \wedge r'_{i-1}) && \text{(par induction)} \\
&= (a_i \wedge b_i) \vee \left((a_i \vee b_i) \wedge \bigvee_{i-1 \geq j \geq 0} \left((a_j \wedge b_j) \wedge \bigwedge_{i-1 \geq k > j} (a_k \vee b_k)\right)\right) && \text{(par déf. de } r'_{i-1}) \\
&= (a_i \wedge b_i) \vee \bigvee_{i-1 \geq j \geq 0} \left((a_j \wedge b_j) \wedge \bigwedge_{i \geq k > j} (a_k \vee b_k)\right) && \text{(par distributivité)} \\
&= \bigvee_{i \geq j \geq 0} \left((a_j \wedge b_j) \wedge \bigwedge_{i \geq k > j} (a_k \vee b_k)\right) \\
&= r'_i && \text{(par déf. de } r'_i). \qquad \square
\end{aligned}$$

Théorème 1. ADD $\in$ FAC^0.

Démonstration. Nous analysons plus rigoureusement la profondeur du circuit. Les autres détails ont déjà été prouvés. Il est clair que prof$(r_0) = 1$. Pour $i > 0$, nous avons

$$\begin{aligned}
\text{prof}(r'_i) &= \text{prof}\Big(\bigvee_{i \geq j \geq 0}((a_j \wedge b_j) \wedge \bigwedge_{i \geq k > j} (a_k \vee b_k))\Big) \\
&= \max\left\{\text{prof}((a_j \wedge b_j) \wedge \bigwedge_{i \geq k > j} (a_k \vee b_k)) + 1 : i \geq j \geq 0\right\} + 1 \\
&= \max\left\{\max(1, \max\left\{\text{prof}(a_k \vee b_k) : i \geq k > j\right\} + 1) + 1 : i \geq j \geq 0\right\} + 1 \\
&= \max\left\{\max(1, \max\left\{1 : i \geq k > j\right\} + 1) + 1 : i \geq j \geq 0\right\} + 1 \\
&= 4. \qquad \square
\end{aligned}$$

Lemme 1. *Soit $G = (V, E)$ un graphe dirigé. S'il existe un chemin de u vers v dans G, alors il existe un chemin de u vers v de longueur au plus $2^{\lceil \log |V| \rceil}$.*

Démonstration. Soit $u \xrightarrow{e_1 e_2 \cdots e_k} v$ un chemin. Si $k \leq n$, alors nous avons fini car $|V| = 2^{\log |V|} \leq 2^{\lceil \log |V| \rceil}$.

Autrement, par le principe du pigeonnier, au moins un sommet w est répété sur le chemin. Plus précisément, il existe $i < j$ tels que

$$u \xrightarrow{e_1 \cdots e_i} w \xrightarrow{e_{i+1} \cdots e_j} w \xrightarrow{e_{j+1} \cdots e_k} v.$$

Ainsi, nous pouvons raccourcir le chemin en retirant le cycle: $u \xrightarrow{e_1 \cdots e_i \, e_{j+1} \cdots e_k} v$. En répétant ce processus, on obtient un chemin sans répétition et ainsi de longueur au plus $|V|$. $\square$