Circuits booléens.

1 Définitions.

Définition 1. Un *circuit booléen* est un DAG[1] dont les sommets sont étiquetés et de degré entrant 0, 1 ou 2 :

- ▷ les sommets de degré entrant 2 sont étiquetés par $\wedge$ ou $\vee$;
- ▷ les sommets de degré entrant 1 sont étiquetés par $\neg$;
- ▷ les sommets de degré 0 sont étiquetés par 0, 1 ou des variables booléennes $x_1, \ldots, x_n$.

Les sommets sont appelés *portes*, et les sommets de degré 0 sont des *portes d'entrées*.

Définition 2. Soit C un circuit booléen avec des variables d'entrée $x_1, \ldots, x_n$. Étant donné une *valuation* $a \in \{0,1\}^n$, pour chaque porte α de C, on définit la *valeur* prise par α sur l'entrée a par :

- ▷ pour les portes d'entrées, on pose

$$\mathrm{val}(x_i) := a_i, \quad \mathrm{val}(0) := \texttt{0}, \quad \mathrm{val}(1) := \texttt{1} \ ;$$

- ▷ pour les autres portes, on pose
  - $\mathrm{val}(\alpha \vee \beta) = \mathrm{val}(\alpha)$ `||` $\mathrm{val}(\beta)$,
  - $\mathrm{val}(\alpha \wedge \beta) = \mathrm{val}(\alpha)$ `&&` $\mathrm{val}(\beta)$,
  - $\mathrm{val}(\neg\alpha) =$ `not` $\mathrm{val}(\alpha)$.

1. *Directed Acyclic Graph*, un graphe orienté acyclique

Définition 3. Si C n'a qu'une seule porte α de degré sortant 0, on dit une α est *porte de sortie* de C, et on pose $\mathrm{val}(C) := \mathrm{val}(\alpha)$.

On peut donc dire que C calcule une fonction $f : \{0,1\}^n \to \{0,1\}$ lorsque $f(a)$ est la valeur prise par C sur l'entrée a.

Plus généralement, si C a s portes de sorties, le circuit C peut calculer une fonction $f : \{0,1\}^n \to \{0,1\}^s$.

Remarque 1. Toute fonction booléenne peut être calculée par un circuit : il suffit de considérer l'arbre de syntaxe de celle-ci.

Exercice 1. Donner une famille de fonctions booléennes « explicites » $(f_n)_{n\geq 1}$ qui ne sont pas calculables par des circuits de taille polynomiale.

Lemme 1. On a ValCirc $\in$ P où

ValCirc	**Entrée.** Une circuit booléen C et $a \in \{0,1\}^n$ **Sortie.** Est-ce que $\mathrm{val}(C) = 1$ sous l'entrée a ?

Preuve. On utilise l'algorithme suivant :

1 : Calculer un tri topologique de C (pour la boucle ci-dessous).
2 : **pour** toute porte α de C **faire**
3 : Évaluer α à l'aide des valeurs déjà calculées [2]
4 : **retourner** la valeur de la porte de sortie

□

2 Simulation de machines de Turing par les circuits.

2. Par l'ordre topologique, on sait que les entrées de α ont déjà été évaluées.

Proposition 1. Soit M une machine à un ruban fonctionnant en temps $T(n)$. On peut simuler M sur les entrées de taille n par un circuit de taille $\mathrm{O}(T(n)^2)$.

Remarque 2. On peut donner la borne de $\mathrm{O}(T(n)\log T(n))$ au lieu de $\mathrm{O}(T(n)^2)$, même pour des machines à plusieurs rubans.

Définition 4. Un *diagramme espace-temps* est un diagramme 2D représentant l'état d'un ruban de M au cours du temps (figure 1)

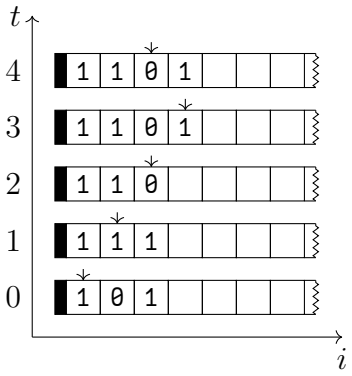

Figure 1 | *Exemple de diagramme espace-temps*

Preuve (Simulation par circuit). Considérons un diagramme de taille $p \times p$, où $p = T(n) + 1$. Par la localité du calcul, le contenu de la cellule (i, t) ne dépend que de trois cellules sur le ruban du dessous : celle directement en dessous, celle en dessous et à gauche, et celle en dessous et à droite.

Définissons une variable $\ell_{a,i,t}$ qui dit « à l'instant t, la case i contient la lettre a » pour tout a, i, t. Définissions aussi $q_{r,i,t}$ indiquant « à l'instant t la machine est dans l'état r et la tête de lecture est sur la case i ».

Comme indiqué, on peut déterminer la valeur de ces variables à partir des valeurs de ces variables avec $(i-1, t-1)$, $(i, t-1)$ et $(i+1, t-1)$. Ceci nous permet d'en déduire une fonction booléenne

$$f : \{0,1\}^{3p} \to \{0,1\}^{p}.$$

Cette fonction ne dépendant que des transitions de M, on en déduit un circuit C calculant f.

En réalisant $O(T(n)^2)$ copies de C, on obtient le circuit final. L'entrée est acceptée si $\bigvee_{i=0}^{T(n)} q_{q_{\text{accepte}},i,T(n)}$ a pour valeur 1. □

Définition 5 (Famille de circuits uniforme). Soit $(C_n)_{n\geq 1}$ une famille de circuits sur n variables. Cette famille est *uniforme* s'il existe une machine de Turing qui, sur l'entrée 1^n, calcule une description complète de C_n en temps polynomial. C'est-à-dire, pour chaque porte α de C_n, elle calcule

- ▷ le type de porte ;
- ▷ si c'est une porte d'entrée, son étiquette ;
- ▷ si ce n'est pas une porte d'entrée, les numéros des portes en entrée de α.

Remarque 3. Si $(C_n)_{n\geq 1}$ est une famille uniforme, alors C_n est de taille polynomiale en n car sa description complète est écrite en temps polynomial.

Théorème 1. Un langage $L \subseteq \{0,1\}^\star$ est dans P si et seulement si L est reconnu par une famille uniforme de circuits booléens de taille polynomiale.

Preuve.

- ▷ « $\implies$ ». Soit $L \in$ P. Dans la preuve précédente, on construit une famille de circuits taille polynomiale. Elle est uniforme car il suffit d'itérer sur i et sur t, pour calculer les

valeurs de $\ell_{a,i,t}$ et $q_{r,i,t}$. Ceci se fait en temps polynomial.

- ▷ « $\Longleftarrow$ ». Pour tester si $x \in \{0,1\}^n$ est dans L, on construit C_n (en temps polynomial), puis on évalue C_n sur x (qui se fait en temps polynomial).

□

3 Premiers problèmes NP-complets.

Théorème 2. Le problème CircuitSat est NP-complet où

CircuitSat	**Entrée.** Une circuit booléen C avec n variables **Sortie.** Existe-t-il a tel que $C(a) = 1$?

Preuve.

- ▷ D'une part, CircuitSat $\in$ NP avec a le certificat et ValCirc le problème de vérification.
- ▷ D'autre part, soit $L \in$ NP. Il existe une machine de Turing non-déterministe M fonctionnant en temps polynômial et qui reconnaît L. Soit $x \in \{0,1\}^n$. On doit construire en temps polynomial un circuit C_x qui est satisfiable ssi $x \in L$.

  On peut simuler M sur l'entrée x par un circuit de taille $O(T(n)^2)\ldots$ mais ce n'est pas suffisant : il faut modéliser le non-déterminisme.

  On ajoute des variables d'entrée supplémentaires $y_1, \ldots, y_{T(n)}$ qui modélisent les choix non-déterministes de la machine M.

  On en déduit C_x (construit en temps polynomial) qui simule M sur l'entrée x avec les $(y_i)_i$ pour les choix non-déterministes.

  On a que C_x est satisfiable si et seulement s'il existe une suite de choix non-déterministes (les valeurs des y_i) telle que M accepte l'entrée x, c'est-à-dire ssi $x \in L$.

□

Théorème 3 (Cook-Levin). Le problème 3-Sat est NP-complet où

3-Sat	**Entrée.** Une formule booléenne ϕ sous 3-CNF **Sortie.** La formule ϕ est-elle satisfiable ?

Preuve.

- ▷ On a que 3-Sat $\in$ NP car on peut utiliser la valuation comme certificat.
- ▷ On fait une réduction de CircuitSat à 3-Sat. Soit C un circuit booléen avec des variables $x_1, \dots, x_n$. Pour chaque porte α, on crée une variable z_α qui représente la valeur prise par le porte α. On utilise l'identité $(P \Rightarrow Q) \Leftrightarrow \bar{P} \vee Q$ pour construire les clauses qui contraignent les variables z_α à respecter le fonctionnement des portes. Par exemple :
  - si $\alpha = \beta \wedge \gamma$, on ajoute les clauses

$$\underbrace{(\bar{z}_\beta \vee \bar{z}_\gamma \vee z_\alpha)}_{\beta\wedge\gamma\Rightarrow\alpha}, \quad \underbrace{(z_\beta \vee \bar{z}_\alpha)}_{\alpha\Rightarrow\beta} \quad \text{et} \quad \underbrace{(z_\gamma \vee \bar{z}_\alpha)}_{\gamma\Rightarrow\alpha} ;$$

  - si $\alpha = \beta \vee \gamma$, on ajoute les clauses

$$\underbrace{(z_\beta \vee z_\gamma \vee \bar{z}_\alpha)}_{\alpha\Rightarrow\beta\vee\gamma}, \quad \underbrace{(\bar{z}_\beta \vee z_\alpha)}_{\beta\Rightarrow\alpha} \quad \text{et} \quad \underbrace{(\bar{z}_\gamma \vee z_\alpha)}_{\gamma\Rightarrow\alpha} ;$$

  - si $\alpha = \neg\beta$, on ajoute les clauses

$$\underbrace{(\bar{z}_\beta \vee \bar{z}_\alpha)}_{\alpha\Rightarrow\bar{\beta}} \quad \text{et} \quad \underbrace{(z_\beta \vee z_\alpha)}_{\bar{\alpha}\Rightarrow\beta} .$$

Enfin, on ajoute la clause $z_{\alpha_{\text{sortie}}}$ pour forcer la porte de sortie à être vraie.

□