

# Cours 3: Circuits. Complexité en temps.

Olivier Bournez

bournez@lix.polytechnique.fr  
LIX, Ecole Polytechnique

# Au menu

La notion de circuit

Complexité en temps

Circuits et algorithmes

Puisque les ordinateurs modernes sont constitués de circuits électroniques digitaux, il est assez naturel que la notion de *circuit* joue un rôle fondamental lorsque l'on parle d'algorithmes et de complexité.

# Sous menu

## La notion de circuit

### Circuits booléens

Relations entre taille et profondeur

Effet Shannon

Bornes inférieures

Circuits à  $m$  sorties

Circuits sur une structure  $\mathfrak{M}$

# Opérations de base

## ■ Négation $\neg$ .

	$\neg$
0	1
1	0



Dans  $\mathbb{Z}_2$ ,  $\neg(x) = 1 + x$ .

## ■ Conjonction $\wedge$

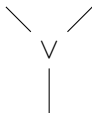
		$\wedge$
0	0	0
0	1	0
1	0	0
1	1	1



Dans  $\mathbb{Z}_2$ ,  $\wedge(x, y) = xy$ .

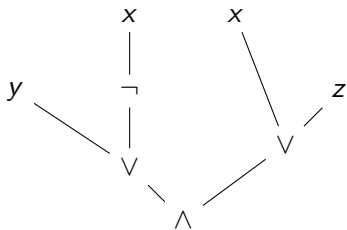
## ■ Disjonction $\vee$

		$\vee$
0	0	0
0	1	1
1	0	1
1	1	1



Dans  $\mathbb{Z}_2$ ,  
 $\vee(x, y) = x + y + xy$ .

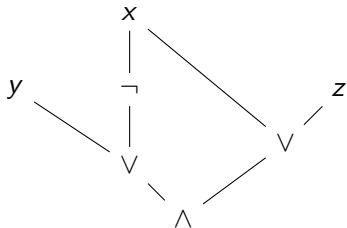
## Exemple : $S(x, y, z)$



■ Table de vérité de  $S(x, y, z)$  (*Sélecteur*)

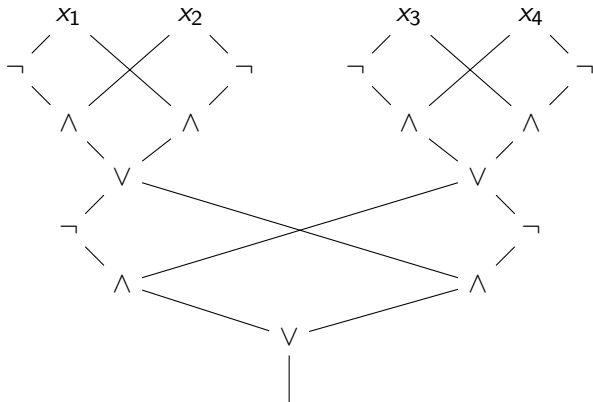
- ▶  $S(\mathbf{0}, y, z) = z$
- ▶  $S(\mathbf{1}, y, z) = y$ .

## Autre représentation du même circuit

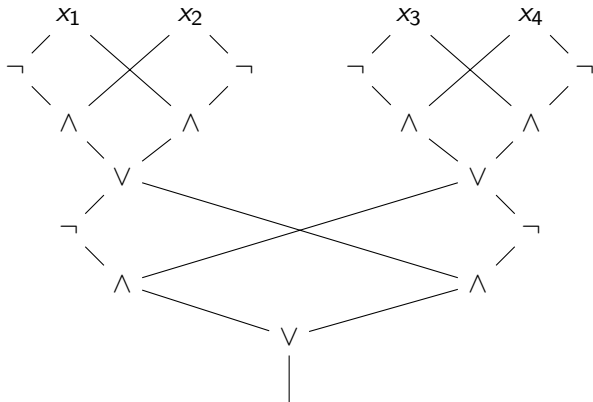


(on partage les entrées)

# Exemple : $PARITY(x_1, x_2, x_3, x_4)$



# Exemple : $PARITY(x_1, x_2, x_3, x_4)$



(vaut 1 ssi un nombre impair de 1 parmi ses arguments)

# Circuit booléen

Un *circuit booléen* a  $n$  entrées et une sortie est un graphe orienté sans cycle (*DAG*) avec

- $n$  entrées, c'est-à-dire  $n$  sommets sans arc entrant.
- exactement une *sortie*, c'est-à-dire un sommet sans arc sortant.
- chaque entrée est étiquetée soit par la constante 0, soit par la constante 1, soit par un symbole de variable  $x_1, x_2, \dots, x_n$ .
- tout autre sommet est appelé une *porte* et est étiqueté soit par  $\vee$ , soit par  $\wedge$ , soit par  $\neg$ . On appelle *fanin* le degré entrant d'une porte.
- les portes étiquetées par  $\vee$  ou  $\wedge$  sont de fanin 2.
- les portes étiquetées par  $\neg$  sont de fanin 1.

## Évaluation d'un circuit

- Étant donné un circuit booléen  $C$  à  $n$  entrées, et  $\bar{x} \in \{0, 1\}^n$ , la *sortie de  $C$*  sur  $\bar{x} = (x_1, \dots, x_n)$ , notée  $C(\bar{x})$  se définit comme on s'y attend.
- La *taille* d'un circuit  $C$ , noté  $taille(C)$  est le nombre de sommets du circuit.
- La *profondeur d'un circuit  $C$* , parfois aussi appelée *hauteur*, notée  $profondeur(C)$  est la longueur du plus long chemin entre une entrée et la sortie.
- On note  $e(C)$  pour le nombre d'entrées du circuit  $C$ .

## Évaluation d'un circuit

- Étant donné un circuit booléen  $C$  à  $n$  entrées, et  $\bar{x} \in \{0, 1\}^n$ , la *sortie de  $C$*  sur  $\bar{x} = (x_1, \dots, x_n)$ , notée  $C(\bar{x})$  se définit comme on s'y attend.
- La *taille* d'un circuit  $C$ , noté  $taille(C)$  est le nombre de sommets du circuit.
  - ▶ Exemple : *PARITY* est de taille 19.
- La *profondeur d'un circuit  $C$* , parfois aussi appelée *hauteur*, notée  $profondeur(C)$  est la longueur du plus long chemin entre une entrée et la sortie.
- On note  $e(C)$  pour le nombre d'entrées du circuit  $C$ .

## Évaluation d'un circuit

- Étant donné un circuit booléen  $C$  à  $n$  entrées, et  $\bar{x} \in \{0, 1\}^n$ , la *sortie de  $C$*  sur  $\bar{x} = (x_1, \dots, x_n)$ , notée  $C(\bar{x})$  se définit comme on s'y attend.
- La *taille* d'un circuit  $C$ , noté  $taille(C)$  est le nombre de sommets du circuit.
  - ▶ Exemple : *PARITY* est de taille 19.
- La *profondeur d'un circuit  $C$* , parfois aussi appelée *hauteur*, notée  $profondeur(C)$  est la longueur du plus long chemin entre une entrée et la sortie.
  - ▶ Exemple : *PARITY* est de profondeur 6.
- On note  $e(C)$  pour le nombre d'entrées du circuit  $C$ .

## Évaluation d'un circuit

- Étant donné un circuit booléen  $C$  à  $n$  entrées, et  $\bar{x} \in \{0, 1\}^n$ , la *sortie de  $C$*  sur  $\bar{x} = (x_1, \dots, x_n)$ , notée  $C(\bar{x})$  se définit comme on s'y attend.
- La *taille* d'un circuit  $C$ , noté  $taille(C)$  est le nombre de sommets du circuit.
  - ▶ Exemple : *PARITY* est de taille 19.
- La *profondeur d'un circuit  $C$* , parfois aussi appelée *hauteur*, notée  $profondeur(C)$  est la longueur du plus long chemin entre une entrée et la sortie.
  - ▶ Exemple : *PARITY* est de profondeur 6.
- On note  $e(C)$  pour le nombre d'entrées du circuit  $C$ .
  - ▶ Exemple : *PARITY* possède 4 entrées.

## Circuits vs Termes

- Un *terme (booléen)* est un circuit à une sortie dans lequel toute porte émet un arc et un seul.
  
- Autrement dit, dans un terme, le graphe a la forme d'un arbre.

# Circuits vs Termes

- Un *terme (booléen)* est un circuit à une sortie dans lequel toute porte émet un arc et un seul.

Exemples :

- ▶ Le circuit pour  $S$  est un terme.
  - ▶ Le circuit pour *PARITY* n'est pas un terme.
- 
- Autrement dit, dans un terme, le graphe a la forme d'un arbre.

# Circuits vs Termes

- Un *terme (booléen)* est un circuit à une sortie dans lequel toute porte émet un arc et un seul.

Exemples :

- ▶ Le circuit pour  $S$  est un terme.
  - ▶ Le circuit pour *PARITY* n'est pas un terme.
- 
- Autrement dit, dans un terme, le graphe a la forme d'un arbre.

Remarque :

- ▶ plusieurs entrées peuvent être étiquetées par un même symbole de constante, ou de variable.

# Circuits vs Termes

- Un *terme (booléen)* est un circuit à une sortie dans lequel toute porte émet un arc et un seul.

Exemples :

- ▶ Le circuit pour  $S$  est un terme.
  - ▶ Le circuit pour *PARITY* n'est pas un terme.
- 
- Autrement dit, dans un terme, le graphe a la forme d'un arbre.

Remarque :

- ▶ plusieurs entrées peuvent être étiquetées par un même symbole de constante, ou de variable.
- ▶  $S$ , avec sa deuxième représentation graphique, est donc bien un terme.

# Sous menu

## La notion de circuit

Circuits booléens

Relations entre taille et profondeur

Effet Shannon

Bornes inférieures

Circuits à  $m$  sorties

Circuits sur une structure  $\mathfrak{M}$

## Relations triviales

### Proposition

*Si aucune entrée n'est une sortie, alors  $\text{taille}(C) \leq 3(\text{taille}(C) - e(C))$ .*

### Proposition

*Si  $C$  est un terme  $T$ , alors  $2e(T) - 1 \leq \text{taille}(T)$  avec égalité si toutes les portes qui ne sont pas des entrées reçoivent deux arcs entrants.*

## ■ Preuve de la première proposition

- ▶ Toute entrée possède au moins un arc sortant, qui va vers une porte  $p$ , qui n'est pas une entrée.
- ▶ Comme  $p$  reçoit au plus deux arcs, elle ne sert que pour deux entrées au plus, soit  $e(C) \leq 2(\text{taille}(C) - e(C))$ , soit  $3e(C) \leq 2\text{taille}(C)$ , ce qui revient à l'inégalité.

## ■ Preuve de la seconde proposition

- ▶ Par récurrence sur la profondeur de  $T$ .
- ▶ C'est clair si  $T$  est de taille 1.
- ▶ Si  $T$  se décompose en deux termes immédiats  $T_1$  et  $T_2$ , alors  $e(T) = e(T_1) + e(T_2)$ , puisque les sous-termes n'ont pas d'entrées communes, et  $\text{taille}(T) = \text{taille}(T_1) + \text{taille}(T_2)$  puisqu'ils sont disjoints.
- ▶ La récurrence fonctionne bien dans ce cas.
- ▶ Si  $T$  ne possède qu'un sous-terme immédiat  $T_1$  alors  $e(T) = e(T_1)$ ,  $\text{taille}(T) = \text{taille}(T_1)$  et la récurrence est encore valide.

# Relations évidentes

## Proposition

*Pour tout circuit  $C$ ,*

$$\text{profondeur}(C) \leq \text{taille}(C) - e(C),$$

$$\text{taille}(C) \leq (2^{\text{profondeur}(C)+1} - 1).$$

*Pour tout terme  $T$ ,  $\log(e(T)) \leq \text{profondeur}(T)$ .*

- La première inégalité vient du fait qu'un chemin maximal de longueur  $\text{profondeur}(C)$  passe par  $\text{profondeur}(C + 1)$  portes, et que la première est une entrée.
- La deuxième inégalité se prouve par récurrence sur  $\text{profondeur}(C)$ . C'est clair si  $\text{profondeur}(C) = 0$ , car la taille vaut alors 1. Sinon, la sortie possède un ou deux sous-circuits immédiats.
- S'il y en a qu'un  $C_1$ , alors  $\text{taille}(C) = \text{taille}(C_1) - 1$ ,  $\text{profondeur}(C_1) \leq \text{profondeur}(C) + 1$ , et donc  $\text{taille}(C) \leq 2^{\text{profondeur}(C)} - 1 + 1 \leq 2^{\text{profondeur}(C)+1} - 1$ .
- S'il y en a deux,  $\text{taille}(C) \leq \text{taille}(C_1) + \text{taille}(C_2) + 1$ , et  $C_1$  et  $C_2$  sont de profondeur au plus  $\text{profondeur}(C) - 1$  si bien que  $\text{taille}(C) \leq 2(2^{\text{profondeur}(C)} - 1) + 1 \leq 2^{\text{profondeur}(C)+1} - 1$ .
- Si  $C$  est un terme  $T$ , alors  $2e(T) - 1 \leq \text{taille}(T)$  par la proposition 2, et donc  $e(T) \leq 2^{\text{profondeur}(P)}$  par ce qui précède.

# Sous menu

## La notion de circuit

Circuits booléens

Relations entre taille et profondeur

**Effet Shannon**

Bornes inférieures

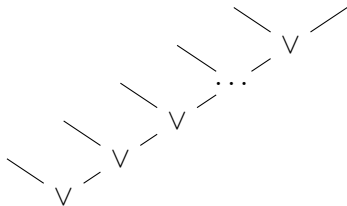
Circuits à  $m$  sorties

Circuits sur une structure  $\mathfrak{M}$

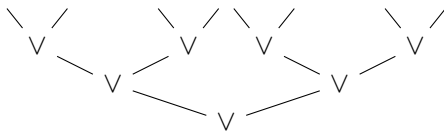
On peut construire un circuit de profondeur  $\log(n) + 1$  qui calcule

1. la fonction disjonction à  $n$  variables : c'est-à-dire la fonction  $x_1 \vee x_2 \cdots \vee x_i \vee \cdots \vee x_n$ , notée  $\bigvee_{i=1}^n x_i$ , de  $\{0, 1\}^n$  dans  $\{0, 1\}$ , qui vaut **1** si et seulement au moins l'un de ses arguments vaut 1.
2. la fonction conjonction à  $n$  variables : c'est-à-dire la fonction  $x_1 \wedge x_2 \cdots \wedge x_i \wedge \cdots \wedge x_n$ , notée  $\bigwedge_{i=1}^n x_i$ , de  $\{0, 1\}^n$  dans  $\{0, 1\}$ , qui vaut **1** si et seulement chacun de ses arguments vaut 1.

- Solution inefficace :



- Solution efficace :



# Effet Shannon

## Proposition

*Toute fonction booléenne en  $n$  variables peut se représenter par un terme booléen de profondeur au plus  $n + \log n + 2$ , et de taille exponentielle en  $n$ .*

- Si la fonction  $f$  est la constante 0, elle se représente par un circuit réduit à un sommet.
- Sinon, on considère chaque valeur  $\bar{e} = (e_1, \dots, e_n)$  telle que  $f(\bar{e}) = 1$  : à une telle valeur, on associe l'expression  $\epsilon_1 x_1 \wedge \dots \wedge \epsilon_i x_i \wedge \dots \wedge \epsilon_n x_n$  où l'on convient que  $\epsilon_i x_i$  vaut  $x_i$  si  $e_i = 1$ , et  $\neg x_i$  si  $e_i = 0$ .
- Pour mettre cette expression sous la forme d'un circuit  $C_{\bar{e}}$ , il faut une profondeur 1 pour nier certaines variables, puis un arbre binaire de profondeur au plus  $\log(n) + 1$  qui prend les conjonctions deux par deux.
- $C_{\bar{e}}$  donne la valeur 1 au uple  $\bar{e}$  et la valeur 0 à tous les autres.
- $f$  s'obtient comme la disjonction de tous les  $C_{\bar{e}}$ .
- Comme il y en a pas plus que  $2^n$ , cela se fait en profondeur  $n$ .

## Exemple

Pour la fonction  $= (x, y)$  de  $\{0, 1\}^2$  dans  $\{0, 1\}$  qui vaut 1 si  $x = y$  :

- $\bar{e}_1 = (1, 1)$  : expression  $x \wedge y$
- $\bar{e}_2 = (0, 0)$  : expression  $\neg x \wedge \neg y$
  
- Expression pour  $=$  :  $(x \wedge y) \vee (\neg x \wedge \neg y)$

- En fait, on ne peut pas espérer faire mieux :
  - ▶ Il y a  $2^{2^n}$  fonctions de  $\{0, 1\}^n$  dans  $\{0, 1\}$ .
  - ▶ Il y a moins de  $2^{p(n)^2 \log(3) + p(n) \log(n+6)}$  circuits de taille  $p(n)$  avec seulement  $n$  entrées.
  - ▶ si  $p(n)$  est un polynôme en  $n$ , ou même une fonction négligeable devant  $2^n$ , on est certain que pour  $n$  grand il y aura des fonctions  $n$ -aires non exprimables par un terme de taille  $p(n)$ ,

- En fait, on ne peut pas espérer faire mieux :
  - ▶ Il y a  $2^{2^n}$  fonctions de  $\{0, 1\}^n$  dans  $\{0, 1\}$ .
  - ▶ Il y a moins de  $2^{p(n)^2 \log(3) + p(n) \log(n+6)}$  circuits de taille  $p(n)$  avec seulement  $n$  entrées.
  - ▶ si  $p(n)$  est un polynôme en  $n$ , ou même une fonction négligeable devant  $2^n$ , on est certain que pour  $n$  grand il y aura des fonctions  $n$ -aires non exprimables par un terme de taille  $p(n)$ ,
  - ▶ ...et même qu'elles constituent la majorité.

# Sous menu

## La notion de circuit

Circuits booléens

Relations entre taille et profondeur

Effet Shannon

**Bornes inférieures**

Circuits à  $m$  sorties

Circuits sur une structure  $\mathfrak{M}$

## Proposition

*L'expression de la fonction  $\text{sup}(x_1, \dots, x_n)$  nécessite un circuit de profondeur au moins  $\log n$ .*

Conséquence du lemme :

### Lemme

*tout circuit acceptant tous les  $00 \dots 010 \dots 00$  et refusant  $00 \dots 00$  possède une profondeur supérieure ou égale à  $\log(n)$ .*

- Par récurrence sur  $n$ .
- Cela est clair pour  $n = 1$ , car toute profondeur vaut au moins  $0 = \log(1)$ .
- Pour le cas général, on considère un tel circuit de profondeur minimal. Sa sortie est étiquetée par la négation, la porte précédant la sortie ne peut être étiquetée par la négation car deux négations se neutralisent.
- L'antécédent est donc étiqueté par une conjonction ou une disjonction. En utilisant les lois de Morgan, on peut donc faire remonter les négations.
- On peut donc supposer sans perte de généralité que l'étiquette de sortie est soit  $\vee$  soit  $\wedge$ . Soient  $A$  et  $B$  ses circuits immédiats. Si l'étiquette est  $\wedge$ , l'un de ces deux sous-circuits au moins refuse  $00 \cdots 00$ , tandis que tous les deux doivent accepter  $00 \cdots 010 \cdots 00$ . Autrement dit, l'un des deux possède la propriété ce qui contredit la minimalité de la profondeur.
- L'étiquette de sortie est donc  $\vee$ . Soit  $X$  l'ensemble des variables  $x_i$  telles que  $A$  donne la valeur 1 sur l'entrée  $00 \cdots 010 \cdots 00$ , le 1 étant à la  $i$ ème place, et soit  $Y$  l'ensemble de variables ayant cette propriété pour  $B$ .
- On doit avoir  $X \cup Y$  égal à toutes les entrées.
- Soit  $A'$  le circuit obtenu à partir de  $A$  en remplaçant par 0 toute variable hors de  $X$ , et  $B'$  celui obtenu à partir de  $B$  en remplaçant par 0 toute variable hors de  $Y$ .
- Comme  $A$  et  $B$  donnent 0 à la suite nulle, et que  $A'$  et  $B'$  ont strictement moins de variables (par minimalité de  $C$ ), la récurrence s'applique et  $\text{profondeur}(A') \geq \log(\text{card}(X))$ ,  $\text{profondeur}(B') \geq \log(\text{card}(Y))$ .
- Donc  $\text{profondeur}(C) \geq \log(\max(\text{card}(X), \text{card}(Y)) + 1) \geq \log(n/2) + \log 2 = \log n$ .

# Sous menu

## La notion de circuit

Circuits booléens

Relations entre taille et profondeur

Effet Shannon

Bornes inférieures

**Circuits à  $m$  sorties**

Circuits sur une structure  $\mathfrak{M}$

## Circuits à $m$ sorties

- On peut considérer des circuits booléens à plusieurs sorties.
- Un circuit à  $n$  entrées et  $m$  sorties calcule une fonction  $C(\bar{x})$  de  $\{0, 1\}^n$  dans  $\{0, 1\}^m$  comme on s'y attend.
- Exemple :
  - ▶  $\text{REPLACE}(s, \bar{x}, \bar{y}) = \bar{y}$  si  $s = 1, \bar{x}$  sinon,  
s'obtient avec un circuit booléen dont la sortie  $i$  vaut  $S(s, y_i, x_i)$ , où  $\bar{x} = (x_1, \dots, x_n)$ ,  $\bar{y} = (y_1, \dots, y_n)$ .

# Sous menu

## La notion de circuit

Circuits booléens

Relations entre taille et profondeur

Effet Shannon

Bornes inférieures

Circuits à  $m$  sorties

Circuits sur une structure  $\mathfrak{M}$

## Circuits sur une structure $\mathfrak{M}$

Un *circuit booléen* a  $n$  entrées et une sortie sur la structure  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  est un graphe orienté sans cycle (*DAG, directly oriented graph*) avec

- $n$  entrées, c'est-à-dire  $n$  sommets sans arc entrant.
- exactement une *sortie*, c'est-à-dire un sommet sans arc sortant.
- chaque entrée est étiquetée soit par une constante de la structure  $\mathfrak{M}$ , par un élément de  $M$  ou par un symbole de variable  $x_1, x_2, \dots, x_n$ .
- tout autre sommet est appelé une *porte* et est étiqueté par une fonction ou une relation de la structure. Le fanin de chaque sommet correspond à l'arité du symbole de fonction ou de relation avec lequel il est étiqueté.

- On définit l'évaluation d'un circuit sur un élément de  $M^n$  comme on s'y attend.
- Une constante étiquetant une entrée du circuit qui ne correspond pas à une fonction 0-aire de la structure est qualifiée de *paramètre*.

# Au menu

La notion de circuit

**Complexité en temps**

Circuits et algorithmes

# Sous menu

Complexité en temps

Mesure du temps de calcul

Version effective de la thèse de Church

## Temps de calcul

- Le *temps de calcul de l'algorithme  $A$  sur l'entrée  $w$* , noté  $\text{TIME}(A, w)$ , est défini comme le nombre d'étapes de l'algorithme  $A$  sur l'entrée  $w$ .

## Temps de calcul

- Le *temps de calcul* de l'algorithme  $A$  sur l'entrée  $w$ , noté  $\text{TIME}(A, w)$ , est défini comme le nombre d'étapes de l'algorithme  $A$  sur l'entrée  $w$ .
  - ▶ À un algorithme  $A$  correspond un système de transitions, c'est-à-dire un ensemble d'états  $S(A)$  et une fonction d'évolution  $\tau_A : S(A) \rightarrow S(A)$ .
  - ▶ A chaque entrée  $w$  est associée un exécution  $X_0, X_1, \dots, X_t$  où  $X_0$  est un état initial qui code  $w$ , et chaque  $X_i$  est un état, et  $X_{i+1} = \tau_A(X_i)$  pour tout  $i$ .
  - ▶ Supposons que  $w$  soit accepté, c'est-à-dire qu'il existe un entier  $t$  avec  $X_t$  terminal.
  - ▶ Le *temps de calcul* sur l'entrée  $w$ , noté  $\text{TIME}(A, w)$ , est défini comme cet entier  $t$ .

# Temps de calcul

- Soit  $A$  un algorithme qui termine sur toute entrée.
- On note  $|w|$  pour la longueur du mot  $w$ .
  
- Le *temps de calcul* de l'algorithme  $A$  est la fonction  $f : \mathbb{N} \rightarrow \mathbb{N}$  telle que

$$f(n) = \max_{|w|=n} \text{TIME}(A, w).$$

## Notations de Landau

Soient  $f$  et  $g$  deux fonctions  $f, g : \mathbb{N} \rightarrow \mathbb{R}^{\geq 0}$ .

- On note  $f(n) = \mathcal{O}(g(n))$  lorsqu'il existe des entiers  $c$  et  $n_0$  tels que pour tout  $n \geq n_0$ ,

$$f(n) \leq cg(n).$$

- On note  $f(n) = o(g(n))$  lorsque pour tout réel  $c$ , il existe un entier  $n_0$  tels que pour tout  $n \geq n_0$ ,

$$f(n) < cg(n).$$

# Sous menu

## Complexité en temps

Mesure du temps de calcul

Version effective de la thèse de Church

## Théorème

*Sur une structure arbitraire  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ , les modèles suivants se simulent deux à deux.*

- 1. Les machines de Turing*
- 2. Les machines à  $k \geq 2$  piles*
- 3. Les machines RAM*
- 4. Les algorithmes*

# Version effective de la thèse de Church

## Théorème

*Sur une structure arbitraire  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ , les modèles suivants se simulent deux à deux en temps polynomial.*

1. *Les machines de Turing*
2. *Les machines à  $k \geq 2$  piles*
3. *Les algorithmes*

■ Formellement :

- ▶ le temps  $T$  d'un modèle est simulé en temps  $T^k$  par l'autre modèle, pour un certain entier  $k$  (qui dépend des modèles).

# Classe P

Soit  $t : \mathbb{N} \rightarrow \mathbb{N}$  une fonction.

- On note

$$\text{TIME}(t(n)) = \{L \mid L \text{ est un langage décidé par un algorithme en temps } \mathcal{O}(t(n))\}.$$

- Temps polynomial :

$$P = \bigcup_k \text{TIME}(n^k).$$

## Classe $P_{\mathfrak{M}}$

Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure.

- Un algorithme à paramètres sur la structure  $\mathfrak{M}$  est un algorithme sur une structure obtenue en étendant  $\mathfrak{M}$  par un nombre fini de constantes.
- Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure.  
On note

$$\text{TIME}_{\mathfrak{M}}(t(n)) = \{L \mid L \text{ est un langage décidé par}$$

un algorithme à paramètres sur  $\mathfrak{M}$  en temps  $\mathcal{O}(t(n))\}$ .

$$P_{\mathfrak{M}} = \bigcup_k \text{TIME}_{\mathfrak{M}}(n^k).$$

# Exemples

- Le problème

$$\text{CIRCUITVALUE} = \{ \langle C, \bar{x} \rangle \mid C \text{ est un circuit booléen et} \\ C(\bar{x}) = 1 \}$$

est dans P.

- Le problème

$$\text{CIRCUITVALUE} = \{ \langle C, \bar{x} \rangle \mid C \text{ est un circuit sur } \mathfrak{M} \text{ et} \\ C(\bar{x}) = 1 \}$$

est dans  $P_{\mathfrak{M}}$ .

# Au menu

La notion de circuit

Complexité en temps

Circuits et algorithmes

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

Premières applications : indécidabilité

Premières applications : bornes inférieures

Reconnaissance par une famille de circuits

Circuits de taille polynomiale

Principe fondamental

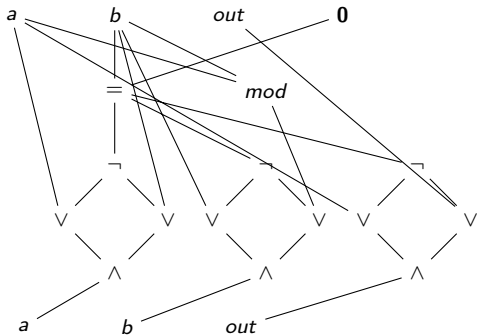
## Proposition

*Soit  $A$  un algorithme (respectivement : sur la structure  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ ). Soit  $n$  un entier. Soit  $t$  un entier. Il existe un circuit  $C_{n,t}$  (resp. sur la structure  $\mathfrak{M}$ ) qui accepte exactement les mots de longueur  $n$  acceptés par l'algorithme  $A$  en  $t$  étapes.*

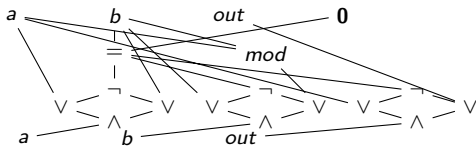
## Exemple : Euclide

```
if b = 0 then out ← a  
else par  
    a ← b  
    b ← a mod b  
endpar
```

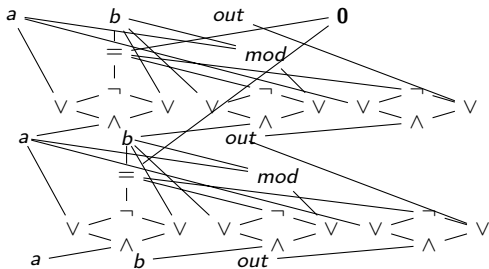
Chaque itération remplace  $(a, b, out)$  par  
 $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



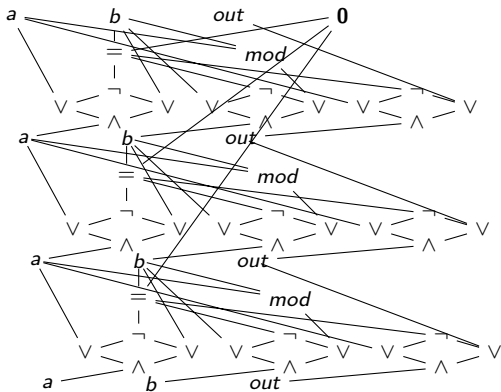
Chaque itération remplace  $(a, b, out)$  par  
 $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



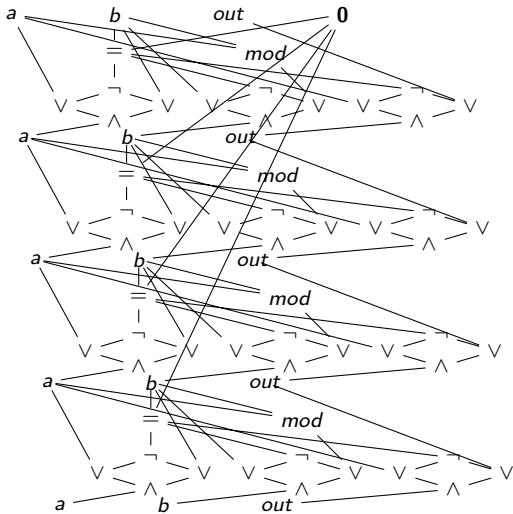
Chaque itération remplace  $(a, b, out)$  par  $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



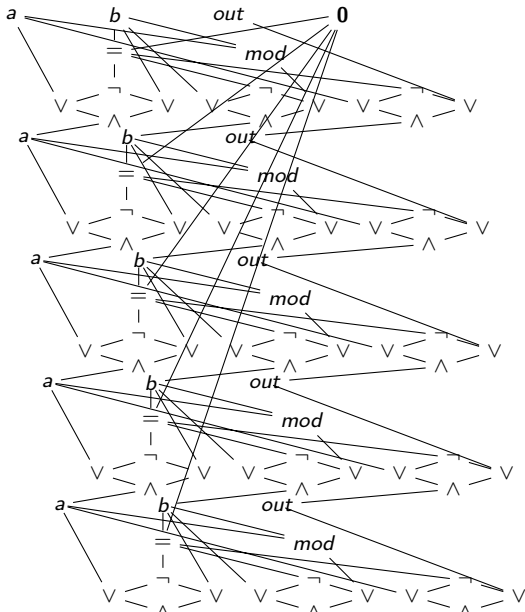
Chaque itération remplace  $(a, b, out)$  par  $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



Chaque itération remplace  $(a, b, out)$  par  $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



Chaque itération remplace  $(a, b, out)$  par  $(S(b = 0, a, b), S(b = 0, b, a \bmod b), S(b = 0, a, out))$ .



# Cas général I/V : notion de configuration

- On dira qu'un emplacement  $(f, \bar{m})$  est *utile*, si  $f$  est un symbole de fonction dynamique, et son contenu  $[[f, \bar{m}]]$  n'est pas **undef**.
- La signature  $\Sigma$  de l'algorithme possède un nombre fini de symboles de fonctions et de relations : on peut donc coder chaque symbole  $f$  de fonction ou de relation par un mot  $\langle f \rangle$  en binaire de  $\{0, 1\}^*$ . On peut supposer que la longueur de ce codage ne dépend pas du symbole  $f$ .
- On appelle *configuration au temps*  $t$  d'un algorithme, notée  $C_t$  la liste des emplacements utiles et de leurs contenus à l'étape  $t$ .
- Puisque chaque itération de l'algorithme (supposé en forme normale) ne modifie qu'un nombre fini d'emplacements, la longueur de la liste  $C_t$  reste affine en  $t$ , majoré par  $k.t + n$ , où  $k$  est une constante.

## Cas général II/V : codage des configurations

- Pour chaque emplacement utile  $(f, \bar{m})$ , on peut coder l'emplacement et son contenu  $[(f, \bar{m})]$  par le mot  $w_{f, \bar{m}} = \langle f \rangle \bar{m} 1 \cdots 1 [(f, \bar{m})]$ , où les 1 sont utilisés pour rendre la longueur  $\ell$  de ce mot indépendante du symbole  $f$ .
- La configuration  $C_t$  se code donc comme un mot  $w_{f_1, \bar{m}_1} w_{f_2, \bar{m}_2} \cdots w_{f_k, \bar{m}_k} 1 \cdots 1$ , où les 1 à droite sont utilisés pour que la longueur de ce mot soit exactement  $k'.t + n$ , où  $k'$  est une constante.
- Il suffit alors de construire pour chaque  $t$  un circuit qui calcule la fonction qui envoie  $C_t$  sur  $C_{t+1}$ .
- En partant de la configuration initiale, en recopiant ce circuit, comme dans l'exemple d'Euclide, pour  $t = 0, 1, \dots, t - 1$ , on obtiendra un circuit qui calcule la configuration au temps  $t$ , et donc le résultat.

## Cas général III/V : briques de base

- EQUAL( $\bar{x}, \bar{y}$ ), où  $\bar{x} = (x_1, \dots, x_n)$  et  $\bar{y} = (y_1, \dots, y_n)$  sont des vecteurs de même longueur, qui vaut 1 si et seulement si  $\bar{x} = \bar{y}$  :
  - ▶ correspond à  $\bigwedge_{i=1}^n (x_i, y_i)$ .
- REPLACE( $s, \bar{x}, \bar{y}$ ), où  $\bar{x} = (x_1, \dots, x_n)$  et  $\bar{y} = (y_1, \dots, y_n)$  sont des vecteurs de même longueur, qui vaut  $\bar{y}$  si  $s = 1$  et  $\bar{x}$  sinon :
  - ▶ correspond à circuit dont la sortie  $i$  vaut  $S(s, y_i, x_i)$ .
- EVALUE( $\langle f \rangle, \bar{m}, C_t$ ), où  $(f, \bar{m})$  est un emplacement, retourne son contenu dans la configuration  $C_t$  :
  - ▶ EVALUE se définit récursivement.
  - ▶ Pour  $|C_t| < l$ , EVALUE est réduit à *undef*.
  - ▶ Pour  $|C_t| \geq l$ , notons  $C_t = w_{\ell-1}vC'_t$ , où  $w_{\ell-1}$  est le préfixe de longueur  $\ell - 1$ , et  $v$  un élément de l'alphabet  $M$ . S'obtient alors par

$$S(\text{EQUAL}(\langle f \rangle, \bar{m}, 1, \dots, 1, w_{\ell-1}), v,$$

$$S(\text{EQUAL}(w_{\ell}, 1, \dots, 1), \text{undef},$$

$$\text{EVALUE}(\langle f \rangle, \bar{m}, C'_t)).$$

## Cas général IV/V : briques de base

- Soit  $t = f(t_1, \dots, t_k)$  un terme.  $TVALUE_t(C_t)$  retourne la valeur du terme  $t$  dans la configuration  $C_t$  :
  - ▶ s'obtient par

$$EVALUE(\langle f \rangle, TVALUE_{t_1}(C_t), \dots, TVALUE_{t_k}(C_t), C_t),$$

si  $f$  est un symbole dynamique, et par

$$f(TVALUE_{t_1}(C_t), \dots, TVALUE_{t_k}(C_t)) \text{ sinon.}$$

- Soit  $f$  un symbole.  $UPDATE_f(\bar{m}, b, C_t)$  retourne la configuration obtenue en effectuant la mise à jour  $(f, \bar{m}) \leftarrow b$  sur  $C_t$  :
  - ▶ Pour  $|C_t| < l$ ,  $UPDATE_f$  est réduit au circuit qui produit  $\langle f \rangle \bar{m}1 \dots 1b$ .
  - ▶ Pour  $|C_t| \geq l$ ,

$$REPLACE(EQUAL(\langle f \rangle, \bar{m}, 1, \dots, 1, w_\ell),$$

$$REPLACE(EQUAL(w_\ell, 1, \dots, 1), \langle f \rangle w_{\ell-1}v, \langle f \rangle \bar{m}1 \dots 1b),$$

$$\langle f \rangle \bar{m}1 \dots 1b).$$

# Cas général V/V : codage des configurations

- Soit  $t \leftarrow t'$  une instruction d'affectation.  $\text{ASSIGN}_{t:=t'}(C_t)$  retourne la configuration obtenue en effectuant cette instruction :
  - ▶ si  $t$  s'écrit  $t = f(t_1, \dots, t_k)$ , s'obtient par le circuit

$$\text{UPDATE}_f(\text{TVALUE}_{t_1}(C_t), \dots, \text{TVALUE}_{t_k}(C_t), \text{TVALUE}_{t'}(C_t), C_t).$$

- Soit **par R endpar** la mise en parallèle d'instructions d'affectation consistantes.  $\text{PAR}_R(C_t)$  retourne la configuration obtenue en effectuant cette instruction :
  - ▶ s'obtient en mettant en parallèle les circuits **ASSIGN** correspondants.
- Soit finalement une instruction **if t then R**, où  $R$  est de la forme précédente :  $\text{DO}(C_t)$  retourne la configuration obtenue en effectuant cette instruction :
  - ▶ s'obtient comme  $\text{REPLACE}(\text{TVALUE}_t(C_t), C_t, \text{PAR}_R(C_t))$
- Le passage de  $C_t$  à  $C_{t+1}$  s'obtient en mettant en parallèle chacun de ces circuits.

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

**Premières applications : indécidabilité**

Premières applications : bornes inférieures

Reconnaissance par une famille de circuits

Circuits de taille polynomiale

Principe fondamental

## Théorème

*Tout ensemble semi-décidable sur la structure*

$$(\mathbb{R}, +, -, *, /, =, <)$$

*possède un nombre dénombrable de composantes connexes.*

- En vertu des propositions précédentes, un ensemble semi-décidable doit s'écrire comme une union dénombrable de langages correspondant à l'union sur  $t$  et sur  $n$  des mots acceptés par les circuits  $C_{n,t}$ .
- Un circuit sur la structure  $(\mathbb{R}, +, -, *, /, =, <)$  reconnaît un ensemble *semi-algébrique* : c'est-à-dire un ensemble qui se définit comme une combinaison booléenne d'inégalités et d'égalités polynomiales.
- On admettra que tout ensemble semi-algébrique possède un nombre fini de composantes connexes.

## Corollary

*L'ensemble de Mandelbrot n'est pas décidable sur structure*

$$(\mathbb{R}, +, -, *, /, =, <)$$

- Cela découle du fait (admis) que le complémentaire de l'ensemble de Mandelbrot ne possède pas un nombre dénombrable de composantes connexes.

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

Premières applications : indécidabilité

**Premières applications : bornes inférieures**

Reconnaissance par une famille de circuits

Circuits de taille polynomiale

Principe fondamental

## Théorème

Sur la structure  $(\mathbb{R}, \mathbf{0}, \mathbf{1}, +, -, =)$ .

le problème KP du sac à dos réel :

- on se donne des réels  $x_1, x_2, \dots, x_n \in \mathbb{R}$
- et on souhaite décider s'il existe un sous ensemble  $S \subset \{1, 2, \dots, n\}$  tel que  $\sum_{i \in S} x_i = 1$

ne peut pas se résoudre en temps polynomial en  $n$ .

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

Premières applications : indécidabilité

Premières applications : bornes inférieures

**Reconnaissance par une famille de circuits**

Circuits de taille polynomiale

Principe fondamental

# Motivation

- Un circuit booléen  $C$  à  $n$  entrées reconnaît un sous-ensemble de  $\{0, 1\}^n$  : il reconnaît les mots  $\bar{x} \in \{0, 1\}^n$  tels que  $C(\bar{x}) = 1$ .
- Avec un circuit, on peut donc parler uniquement de mots d'une longueur fixée.
- Pour pouvoir parler de mots de longueurs arbitraires, on utilise des familles  $(C_n)_{n \in \mathbb{N}}$  de circuits.

- Soit  $(C_n)_{n \in \mathbb{N}}$  une famille de circuits booléens (respectivement : sur une structure  $\mathfrak{M}$ ), où le circuit  $C_n$  possède exactement  $n$  entrées.
- Le langage  $L$  reconnu par cette famille est défini par la propriété suivante :

$$w \in L \text{ si et seulement si } C_{|w|}(w) = \mathbf{1}.$$

- Le langage  $L$  reconnu par cette famille avec paramètres  $c_1, \dots, c_k \in M$  est défini par la propriété suivante :

$$w \in L \text{ si et seulement si } C_{|w|+k}(c_1, \dots, c_k, w) = \mathbf{1}.$$

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

Premières applications : indécidabilité

Premières applications : bornes inférieures

Reconnaissance par une famille de circuits

**Circuits de taille polynomiale**

Principe fondamental

## Circuits de taille polynomiale

- On dira qu'un langage est dans la classe  $\mathbb{P}$  s'il est reconnu par une famille de circuits booléens de taille polynomiale.
- Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure.  
On dira qu'un langage est dans la classe  $\mathbb{P}_{\mathfrak{M}}$  s'il est reconnu par une famille de circuits sur  $\mathfrak{M}$  à paramètres de taille polynomiale.

# Sous menu

## Circuits et algorithmes

Principe fondamental : version simple

Premières applications : indécidabilité

Premières applications : bornes inférieures

Reconnaissance par une famille de circuits

Circuits de taille polynomiale

**Principe fondamental**

# Principe fondamental

## Théorème

Soit  $A$  un algorithme (respectivement : sur la structure  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ ), qui termine sur toute entrée en temps  $t(n)$ .

Alors :

1. Il existe alors une famille  $(C_n)_{n \in \mathbb{N}}$  de circuits booléens (resp : de circuits sur la structure  $\mathfrak{M}$ ), de taille  $\mathcal{O}(t(n)^k)$  pour un certain entier  $k$ , qui reconnaît le langage décidé par  $A$ .
2. La fonction qui à l'entier  $n$  associe la description  $\langle C_n \rangle$  du circuit  $C_n$  est calculable en un temps polynomial en  $n$ .

- On sait qu'il existe une famille  $(C_{n,t})_{n,t}$ , où  $C_{n,t}$  accepte les mots de longueur  $n$  acceptés en moins de  $t$  étapes.
- Considérer la famille  $(C_n)_{n \in \mathbb{N}}$  donnée par  $C_n = C_{n,t(n)}$  pour tout  $n$ .
- Et observer que la description de cette famille est bien calculable en temps polynomial en  $n$ .

- En particulier si  $t(n)$  est polynomial, alors la famille est de taille polynomiale.
- Le point 2. est ce que l'on appelle une condition *d'uniformité* : la suite des circuits est calculable, même calculable en temps polynomial en  $n$ .

- En particulier si  $t(n)$  est polynomial, alors la famille est de taille polynomiale.
- Le point 2. est ce que l'on appelle une condition *d'uniformité* : la suite des circuits est calculable, même calculable en temps polynomial en  $n$ .
  - ▶ Dans la deuxième condition, on parle bien d'un temps polynomial en l'entier  $n$  :
    - la fonction qui à  $\mathbf{1}^n$  (le mot constitué de  $n$  fois la lettre 1) associe la description  $\langle C_n \rangle$  du circuit  $C_n$  est calculable en un temps polynomial.

# Principe fondamental : Caractérisation

## Théorème

*Un problème est dans  $P$  (resp.  $P_{\text{PT}}$ ) si et seulement si*

- 1. il est dans  $\mathbb{P}$*
- 2. la famille est uniforme :*
  - ▶ la famille de circuits booléens correspondante est calculable en temps polynomial (resp. avec paramètres) : on peut produire  $\langle C_n \rangle$  en un temps polynomial en  $n$ .*

L'implication inverse vient de la remarque suivante :

- soit  $L$  un langage dans  $\mathbb{P}$  (respectivement :  $\mathbb{P}_{\mathfrak{M}}^0, \mathbb{P}_{\mathfrak{M}}$ ).
- On peut déterminer si  $w \in L$  par l'algorithme qui consiste à produire le circuit  $C_{|w|}$ , puis à simuler ce circuit sur l'entrée  $w$  (resp. avec ses paramètres  $c_1, \dots, c_k$ ).
- Par hypothèse, tout cela se fait en temps polynomial, car **CIRCUITVALUE** est polynomial.