Bornes inférieures non-uniformes

Sylvain Perifel

LIAFA, Université Paris 7

Journée de rentrée Automates
17 octobre 2008

Plan

1. Circuits booléens

2. Diagonalisation simple

3. Bornes inférieures pour le permanent

Plan

Le décor

- Habituellement : un unique algorithme pour toutes les tailles d'entrée – uniformité
  Exemple : $\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k)$

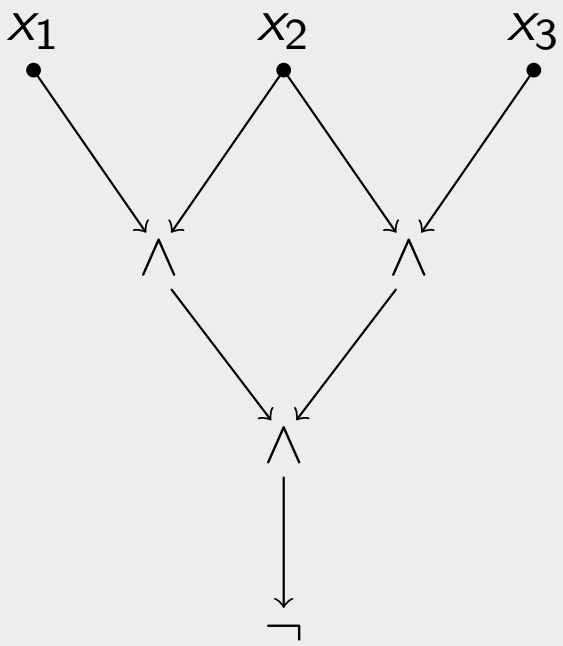
Le décor

- Habituellement : un unique algorithme pour toutes les tailles d'entrée – uniformité
  Exemple : $\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k)$
- Non-uniformité : un algorithme différent pour chaque taille d'entrée

Le décor

- Habituellement : un unique algorithme pour toutes les tailles d'entrée – uniformité
  Exemple : $\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k)$
- Non-uniformité : un algorithme différent pour chaque taille d'entrée

Formalisation : circuits booléens
(portes $\wedge$, $\vee$ et $\neg$)



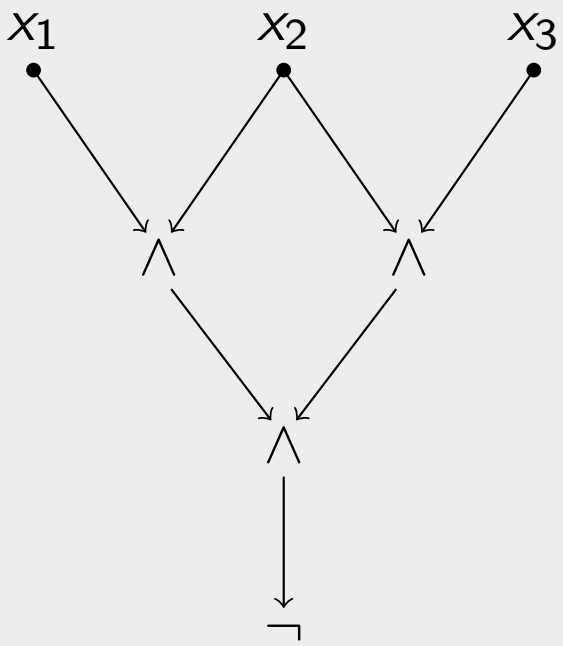
Le décor

- Habituellement : un unique algorithme pour toutes les tailles d'entrée – uniformité
  Exemple : $\mathrm{P} = \bigcup_k \mathrm{DTIME}(n^k)$
- Non-uniformité : un algorithme différent pour chaque taille d'entrée

Formalisation : circuits booléens
(portes $\wedge$, $\vee$ et $\neg$)

Pour reconnaître un langage :

- famille $(C_n)_{n\in\mathbb{N}}$ de circuits
- C_n a n entrées et une sortie
- $x \in A \iff C_{|x|}(x) = 1$
- Taille d'un circuit = nb portes



Les protagonistes

- P/poly : ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme

Les protagonistes

- P/poly : ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme
  Remarque : si les circuits sont constructibles en temps polynomial, alors on obtient la classe P.

Les protagonistes

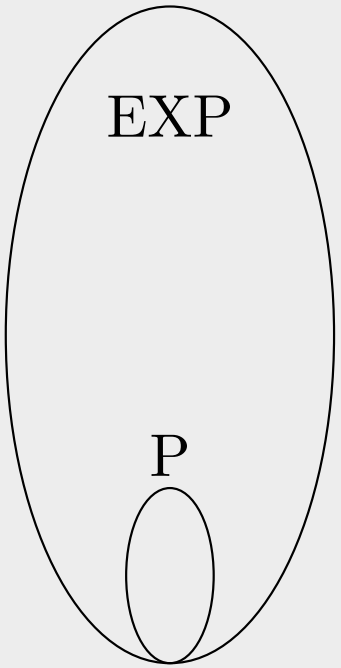
- $\mathrm{P/poly}$: ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme
- EXP : ensemble des langages reconnus par machine de Turing déterministe en temps exponentiel — uniforme

$$\mathrm{EXP} = \cup_{k \geq 0} \mathrm{DTIME}(2^{n^k}).$$

Les protagonistes

- $\mathrm{P/poly}$: ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme
- EXP : ensemble des langages reconnus par machine de Turing déterministe en temps exponentiel — uniforme

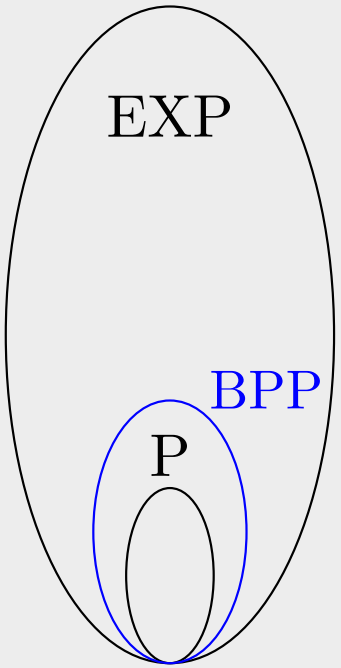
$$\mathrm{EXP} = \cup_{k \geq 0} \mathrm{DTIME}(2^{n^k}).$$



Les protagonistes

- $\mathrm{P/poly}$: ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme
- EXP : ensemble des langages reconnus par machine de Turing déterministe en temps exponentiel — uniforme

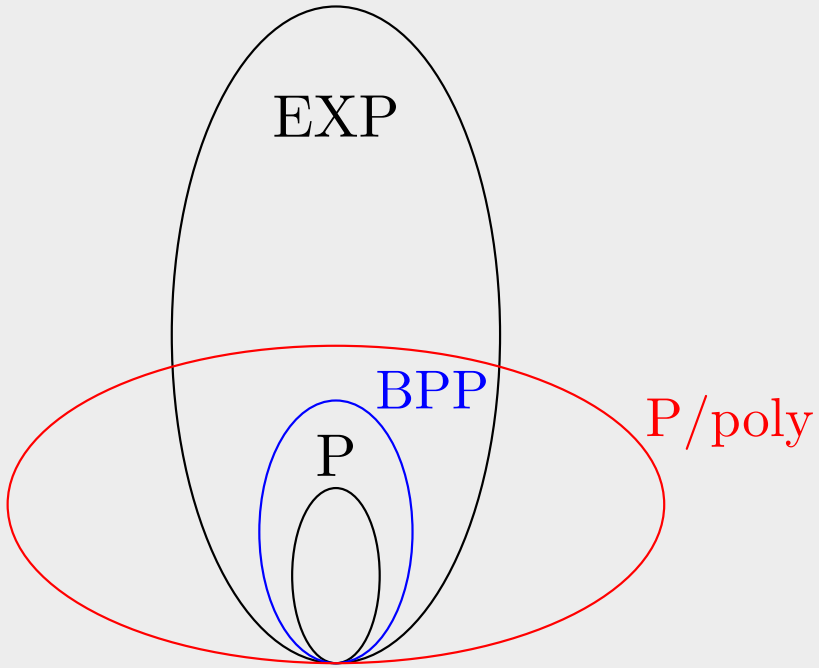
$$\mathrm{EXP} = \cup_{k \geq 0} \mathrm{DTIME}(2^{n^k}).$$



Les protagonistes

- $\mathrm{P/poly}$: ensemble des langages reconnus par une famille de circuits de taille polynomiale — non-uniforme
- EXP : ensemble des langages reconnus par machine de Turing déterministe en temps exponentiel — uniforme

$$\mathrm{EXP} = \cup_{k \geq 0} \mathrm{DTIME}(2^{n^k}).$$



- Théorème de hiérarchie en temps (Hartmanis & Stearns, 1965) :

$$\mathrm{EXP} \neq \mathrm{P}.$$

- Théorème de hiérarchie en temps (Hartmanis & Stearns, 1965) :

$$\mathrm{EXP} \neq \mathrm{P}.$$

- Bornes inférieures non-uniformes :
  Existe-t-il des langages ayant un algorithme exponentiel mais pas de circuits de taille polynomiale ? Ouvert

$$\mathrm{EXP} \not\subseteq \mathrm{P/poly} \,?$$

- Théorème de hiérarchie en temps (Hartmanis & Stearns, 1965) :

$$\mathrm{EXP} \neq \mathrm{P}.$$

- Bornes inférieures non-uniformes :
  Existe-t-il des langages ayant un algorithme exponentiel mais pas de circuits de taille polynomiale ? Ouvert

$$\mathrm{EXP} \not\subseteq \mathrm{P/poly} \,?$$

- Remarque : $\mathrm{EXP} \neq \mathrm{P/poly}$ (il y a des langages indécidables dans $\mathrm{P/poly}$).

Liens avec la dérandomisation

- Dérandomisation (nf) – *barbarisme*. Fait de rendre déterministe un algorithme probabiliste. Associer à un algorithme probabiliste polynomial un algorithme déterministe polynomial équivalent. *Verbe :* dérandomiser.

Liens avec la dérandomisation

- Dérandomisation (nf) – *barbarisme*. Fait de rendre déterministe un algorithme probabiliste. Associer à un algorithme probabiliste polynomial un algorithme déterministe polynomial équivalent. *Verbe :* dérandomiser.
  Question cruciale : peut-on dérandomiser tout algorithme probabiliste ($\mathrm{P} = \mathrm{BPP}$) ? L'aléatoire accélère-t-il le calcul ?

Liens avec la dérandomisation

- Dérandomisation (nf) – *barbarisme*. Fait de rendre déterministe un algorithme probabiliste. Associer à un algorithme probabiliste polynomial un algorithme déterministe polynomial équivalent. *Verbe :* dérandomiser.
  Question cruciale : peut-on dérandomiser tout algorithme probabiliste ($\mathrm{P} = \mathrm{BPP}$) ? L'aléatoire accélère-t-il le calcul ?

- Approche des générateurs pseudo-aléatoires (Yao 1982) :
  - Impagliazzo & Wigderson 1997 : si EXP nécessite des circuits de taille exponentielle, alors $\mathrm{BPP} = \mathrm{P}$.
  - Babai, Fortnow, Nisan & Wigderson 1993 : si $\mathrm{EXP} \not\subset \mathrm{P/poly}$ alors BPP a des algorithmes sous-exponentiels.

Liens avec la dérandomisation

- Dérandomisation (nf) – *barbarisme*. Fait de rendre déterministe un algorithme probabiliste. Associer à un algorithme probabiliste polynomial un algorithme déterministe polynomial équivalent. *Verbe :* dérandomiser.
  Question cruciale : peut-on dérandomiser tout algorithme probabiliste ($\mathrm{P} = \mathrm{BPP}$) ? L'aléatoire accélère-t-il le calcul ?

- Approche des générateurs pseudo-aléatoires (Yao 1982) :
  - Impagliazzo & Wigderson 1997 : si EXP nécessite des circuits de taille exponentielle, alors $\mathrm{BPP} = \mathrm{P}$.
  - Babai, Fortnow, Nisan & Wigderson 1993 : si $\mathrm{EXP} \not\subset \mathrm{P/poly}$ alors BPP a des algorithmes sous-exponentiels.

- Autre direction, Kabanets & Impagliazzo 2003 : si $\mathrm{P} = \mathrm{BPP}$ alors NEXP n'a pas de circuits de taille polynomiale.

Notre mouton principal : “$\mathrm{EXP} \subset \mathrm{P/poly}$?”

- Une simple diagonalisation échoue (trop de circuits).

Notre mouton principal : “$\mathrm{EXP} \subset \mathrm{P/poly}$?”

- Une simple diagonalisation échoue (trop de circuits).
- Kannan 1982 : $\mathrm{NEXP}^{\mathrm{NP}} \not\subset \mathrm{P/poly}$;

Notre mouton principal : “$\mathrm{EXP} \subset \mathrm{P/poly}$?”

- Une simple diagonalisation échoue (trop de circuits).
- Kannan 1982 : $\mathrm{NEXP}^{\mathrm{NP}} \not\subset \mathrm{P/poly}$;
- Homer & Mocas 1995 : $\forall c > 0, \mathrm{EXP}$ n'a pas de circuits de taille n^c.

Notre mouton principal : “$\mathrm{EXP} \subset \mathrm{P/poly}$?”

- Une simple diagonalisation échoue (trop de circuits).
- Kannan 1982 : $\mathrm{NEXP}^{\mathrm{NP}} \not\subset \mathrm{P/poly}$;
- Homer & Mocas 1995 : $\forall c > 0, \mathrm{EXP}$ n'a pas de circuits de taille n^c.
- Une hypothèse de symétrie de l'information ($\mathrm{SI_p}$) implique $\mathrm{EXP} \not\subset \mathrm{P/poly}$.

Notre mouton principal : “$\mathrm{EXP} \subset \mathrm{P/poly}$?”

- Une simple diagonalisation échoue (trop de circuits).
- Kannan 1982 : $\mathrm{NEXP}^{\mathrm{NP}} \not\subset \mathrm{P/poly}$;
- Homer & Mocas 1995 : $\forall c > 0, \mathrm{EXP}$ n'a pas de circuits de taille n^c. ← dans la suite
- Une hypothèse de symétrie de l'information ($\mathrm{SI_p}$) implique $\mathrm{EXP} \not\subset \mathrm{P/poly}$. ← dans la suite

Plan

Circuits de taille n^c

- But du jeu : construire un langage A ayant un algorithme exponentiel mais pas de circuits de taille n^c.
- On ordonne les mots de $\{0,1\}^n$: $x_1 < x_2 < \cdots < x_{2^n}$.

Circuits de taille n^c

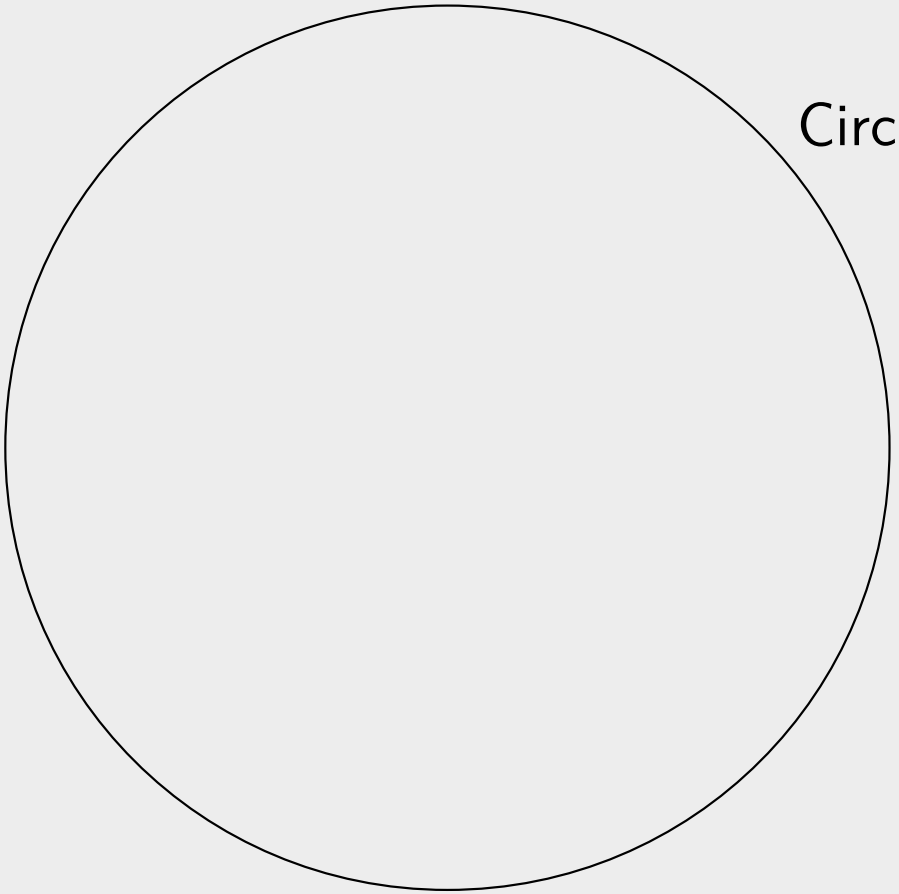
Proposition

Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.

Circuits de taille n^c

Proposition

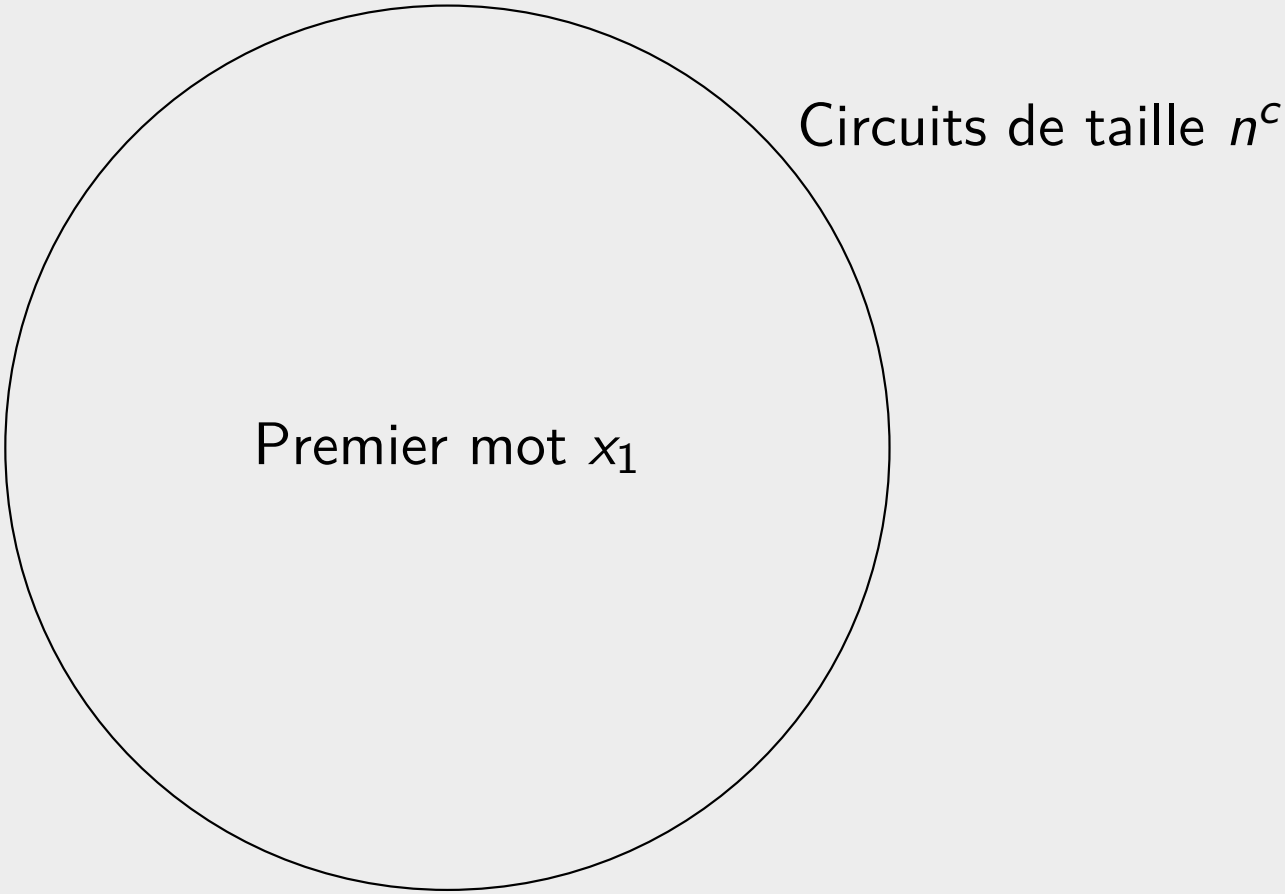
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

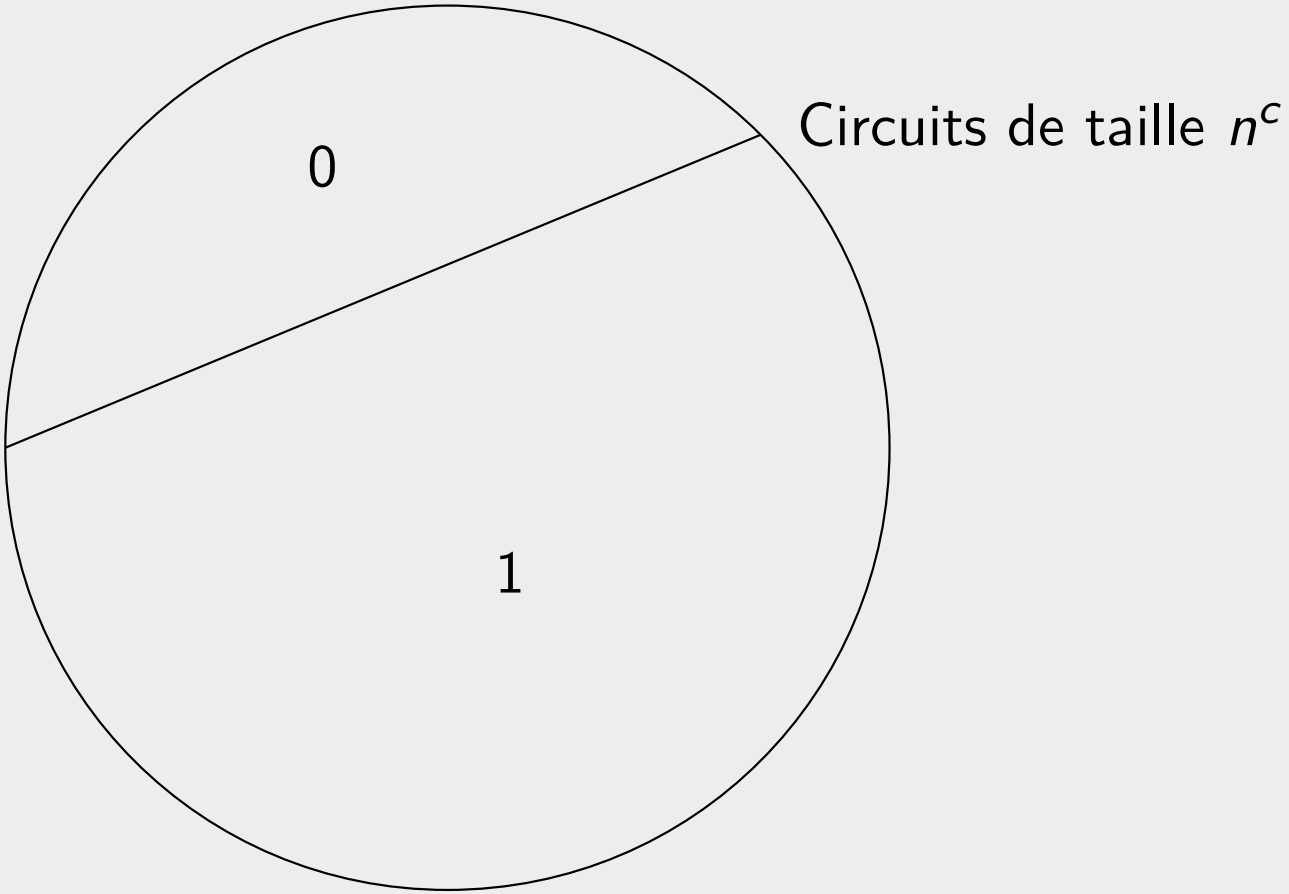
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

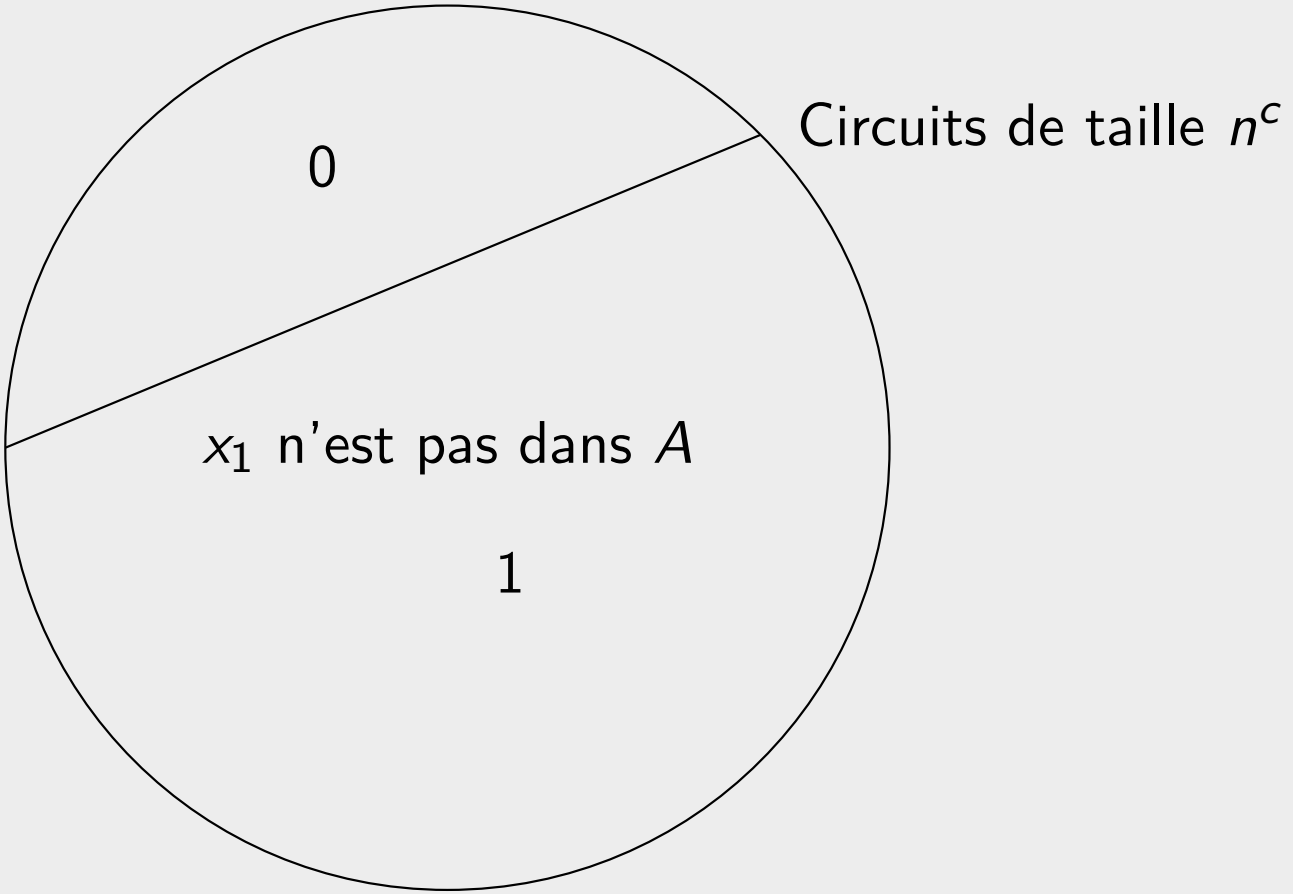
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

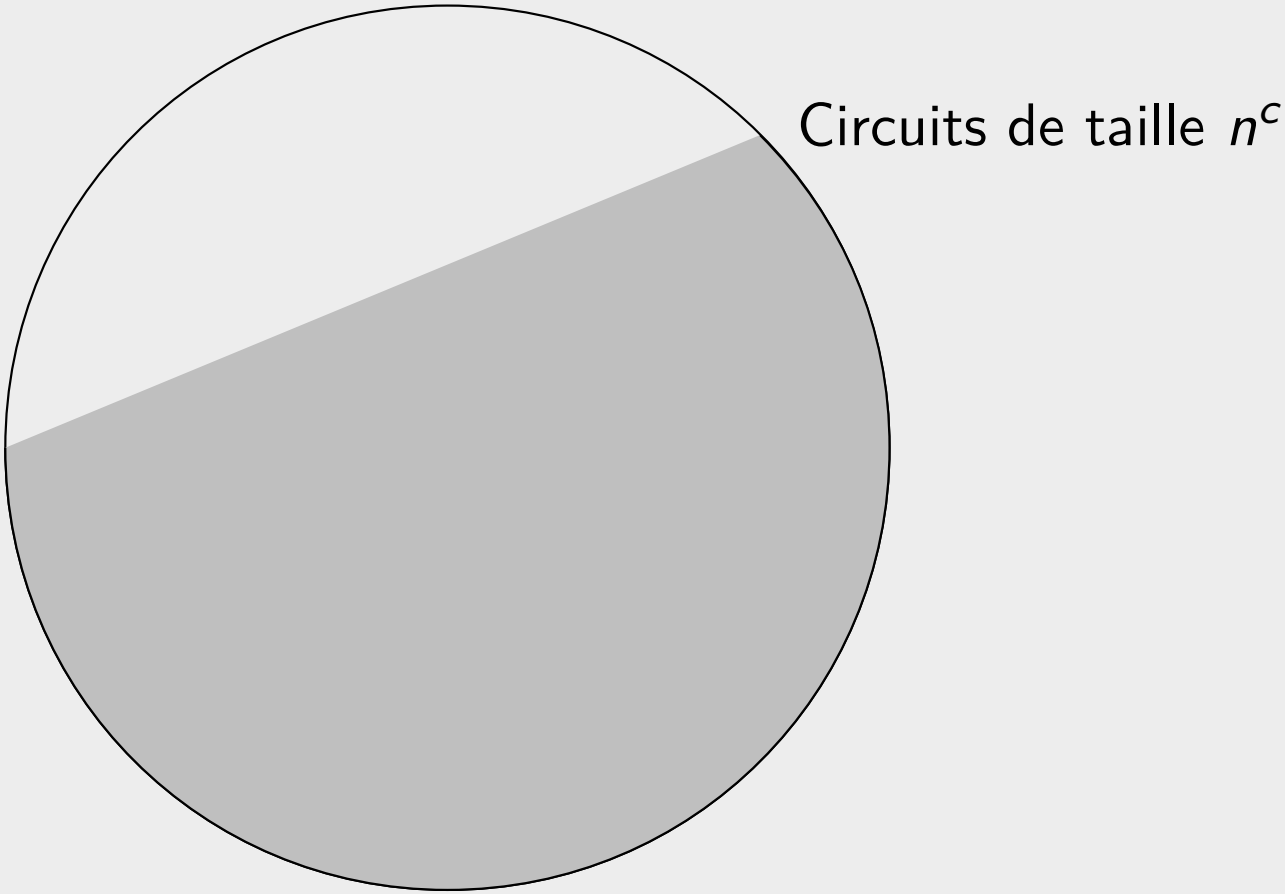
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

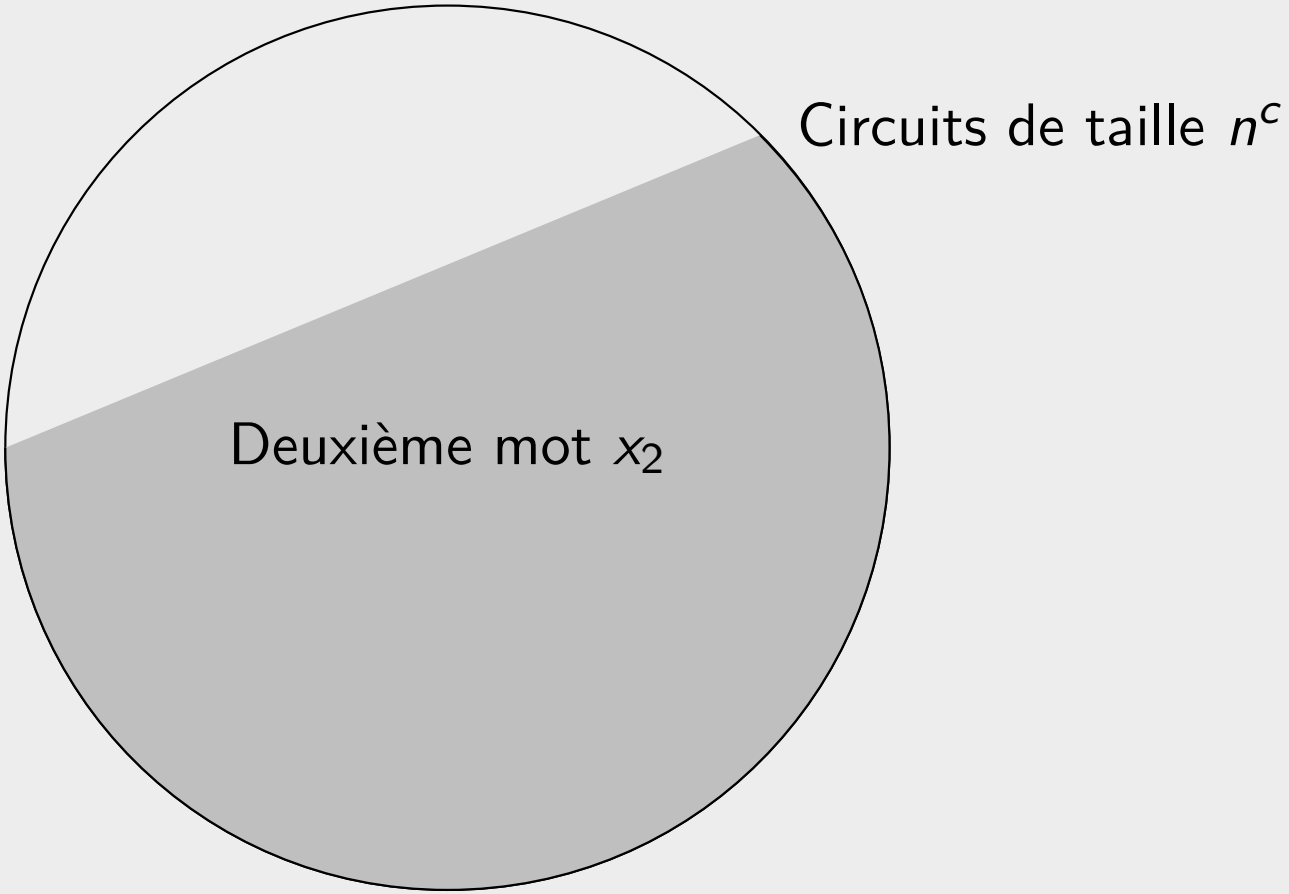
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

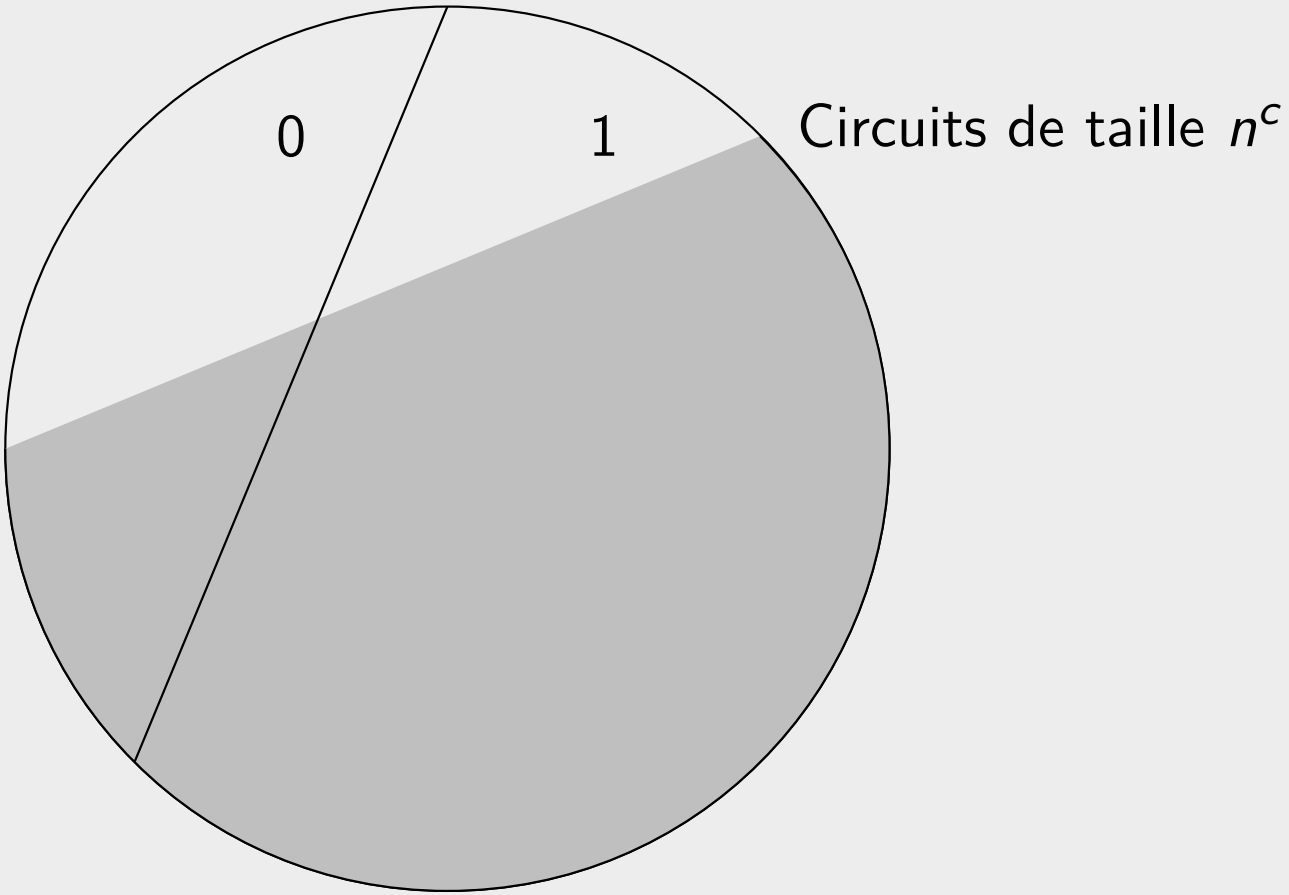
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

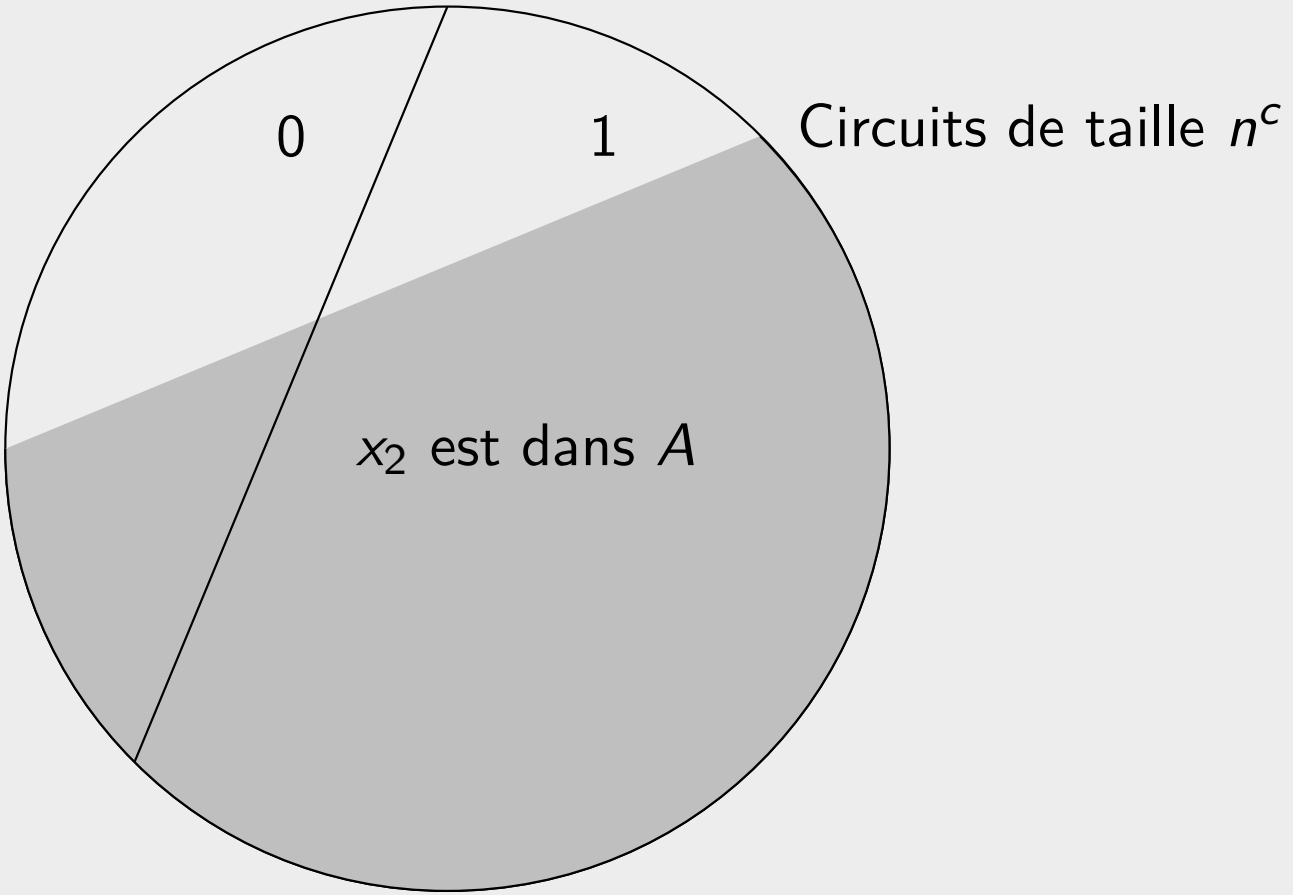
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

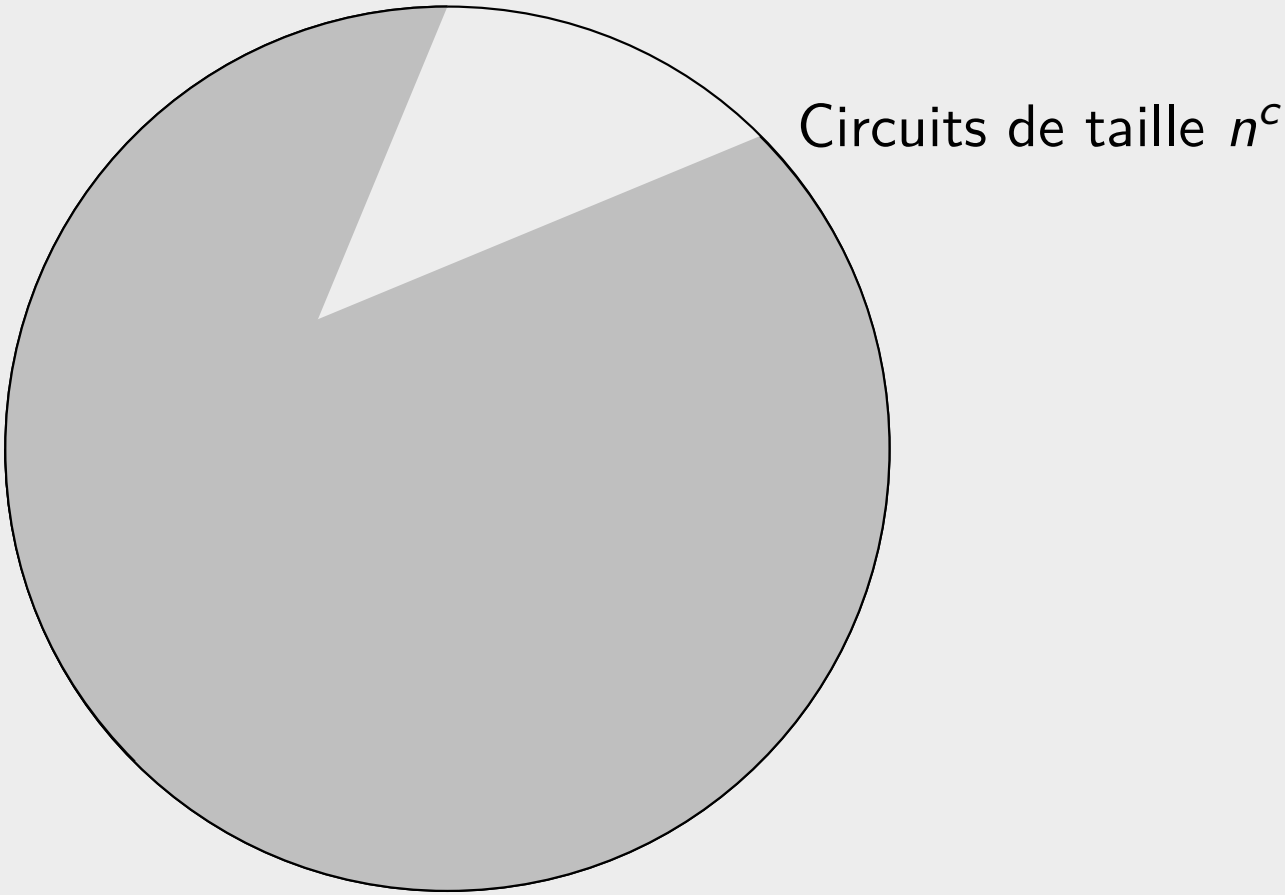
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

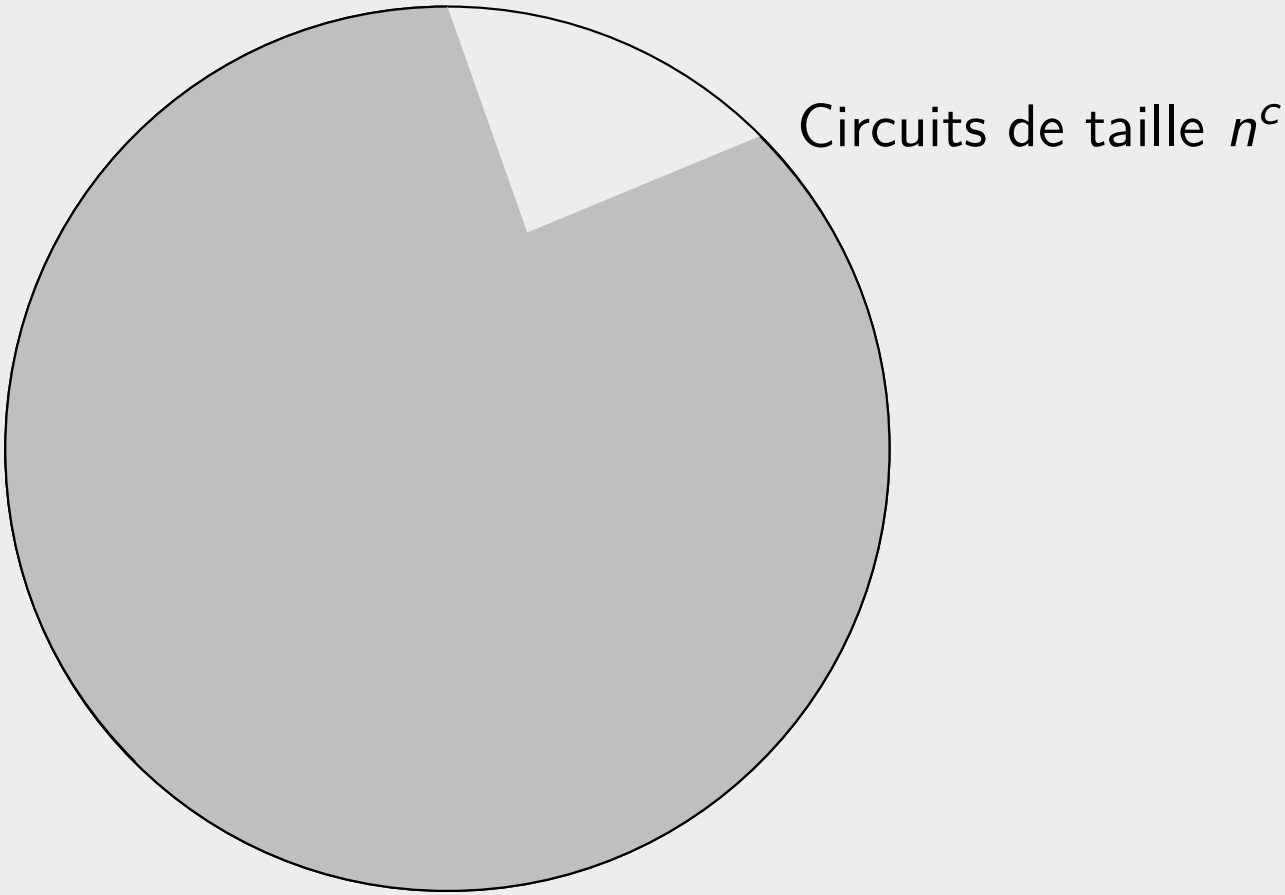
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

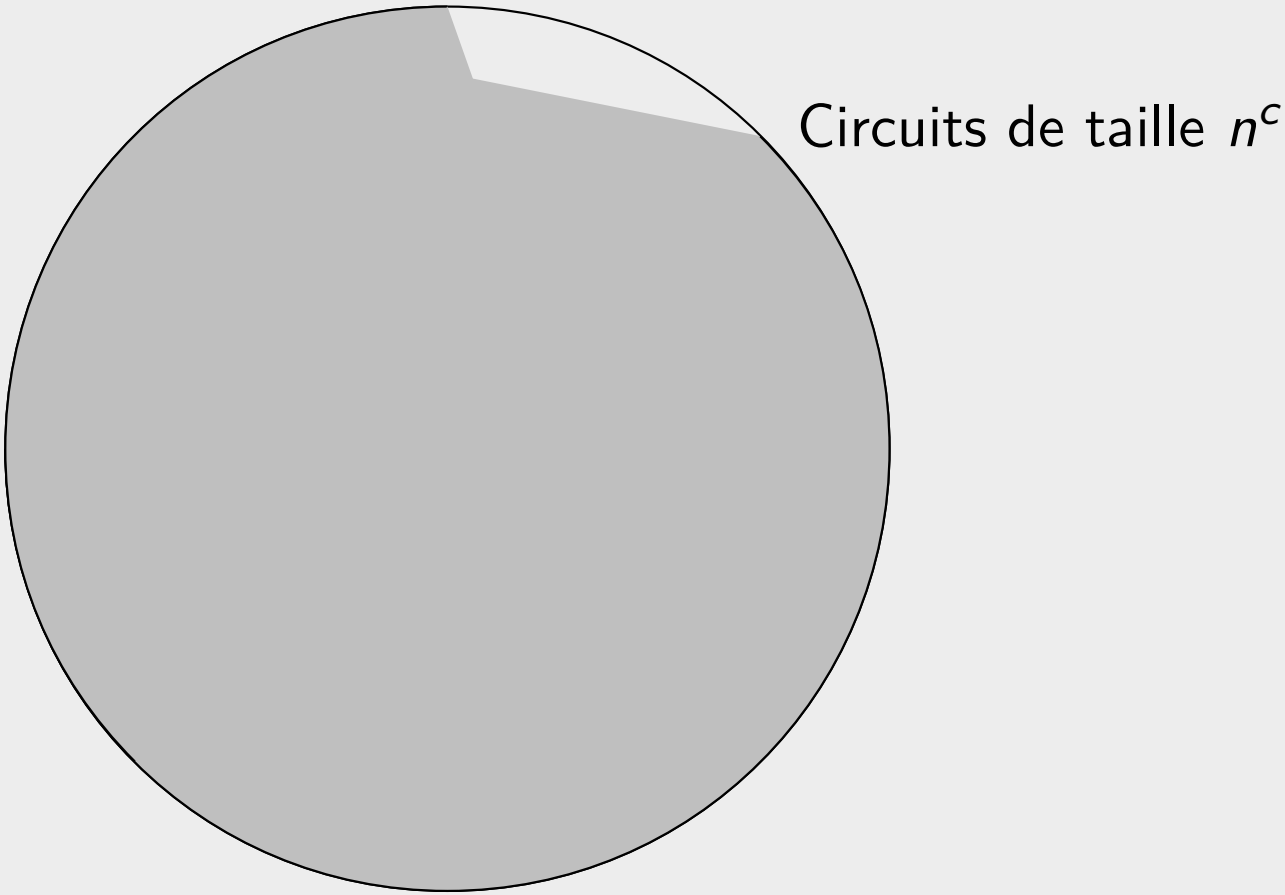
Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Circuits de taille n^c

Proposition

Pour toute constante $c > 0$, il existe un langage $A \in \mathrm{EXP}$ n'ayant pas de circuits de taille n^c.



Symétrie de l'information

- Levin & Kolmogorov : si x et y sont "indépendants", alors l'information contenue dans (x, y) est la somme de celles contenues dans x et dans y.

$$C(x, y) \simeq C(x) + C(y|x).$$

Symétrie de l'information

- Levin & Kolmogorov : si x et y sont "indépendants", alors l'information contenue dans (x, y) est la somme de celles contenues dans x et dans y.

$$C(x, y) \simeq C(x) + C(y|x).$$

- Hypothèse $(\mathrm{SI_p})$: idem en temps polynomial.

Proposition

$(\mathrm{SI_p}) \Rightarrow \mathrm{EXP} \not\subseteq \mathrm{P/poly}.$

- Levin & Kolmogorov : si x et y sont "indépendants", alors l'information contenue dans (x, y) est la somme de celles contenues dans x et dans y.

$$C(x, y) \simeq C(x) + C(y|x).$$

- Hypothèse $(\mathrm{SI_p})$: idem en temps polynomial.

Proposition

$(\mathrm{SI_p}) \Rightarrow \mathrm{EXP} \not\subseteq \mathrm{P/poly}$.

Preuve (de loin)

- On coupe les circuits de taille $n^{\log n}$ en morceaux indépendants de taille n. Dans EXP on peut diagonaliser sur ces morceaux.
- On "recolle" les morceaux avec $(\mathrm{SI_p})$. □

Plan

Permanent

- Déterminant (polynôme en n^2 variables) :

$$\det_n(x_{1,1}, x_{1,2}, \ldots, x_{n,n}) = \sum_{\sigma} \epsilon(\sigma) \prod_{i=1}^{n} x_{i,\sigma(i)}$$

- Permanent (polynôme en n^2 variables) :

$$\mathrm{per}_n(x_{1,1}, x_{1,2}, \ldots, x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

Permanent

- Permanent (polynôme en n^2 variables) :

$$\mathrm{per}_n(x_{1,1}, x_{1,2}, \ldots, x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

- Le permanent compte le nombre de couplages parfaits d'un graphe biparti.
  Remarque : algorithme polynomial pour décider si un graphe a un couplage parfait.

Permanent

- Permanent (polynôme en n^2 variables) :

$$\mathrm{per}_n(x_{1,1}, x_{1,2}, \ldots, x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

- Le permanent compte le nombre de couplages parfaits d'un graphe biparti.
  Remarque : algorithme polynomial pour décider si un graphe a un couplage parfait.
- Calculer le permanent d'un matrice : $\#\mathrm{P}$-complet.
  Remarque : $\#\mathrm{P}$ contient NP, $\mathrm{NP}^{\mathrm{NP}}$, ..., PH (Toda, 1989)

Permanent

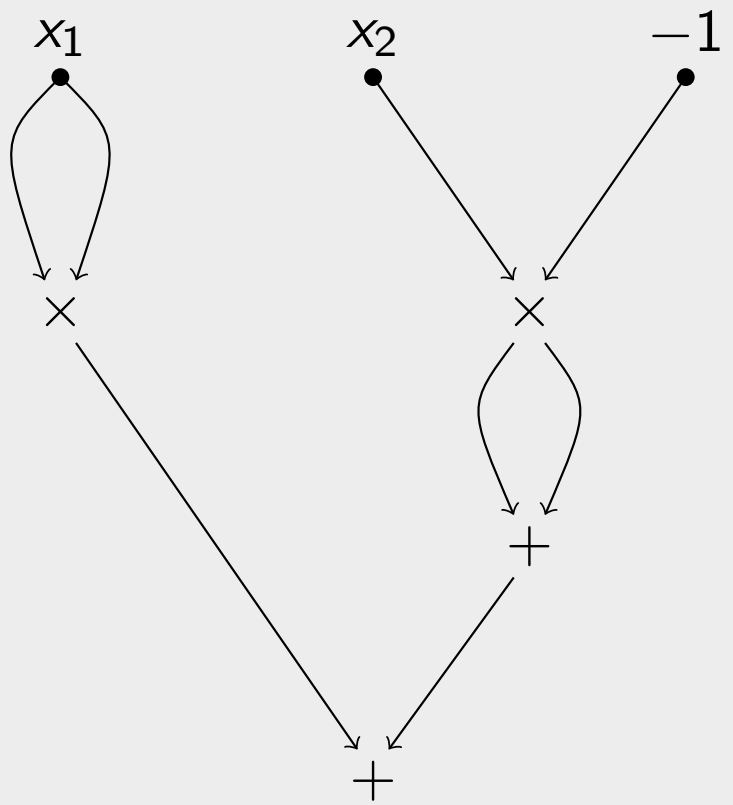
- Permanent (polynôme en n^2 variables) :

$$\mathrm{per}_n(x_{1,1}, x_{1,2}, \ldots, x_{n,n}) = \sum_{\sigma} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

- Le permanent compte le nombre de couplages parfaits d'un graphe biparti.
  Remarque : algorithme polynomial pour décider si un graphe a un couplage parfait.
- Calculer le permanent d'un matrice : $\#\mathrm{P}$-complet.
  Remarque : $\#\mathrm{P}$ contient NP, $\mathrm{NP}^{\mathrm{NP}}$, ..., PH (Toda, 1989)
- Borne inférieure sur la complexité du calcul du permanent ?
  Sur la taille de circuits pour le permanent ?

Circuits arithmétiques :

- portes $+$ et $\times$;
- entrées : variables $x_1, \ldots, x_n$ et constantes $\alpha \in K$;
- $\longrightarrow$ polynôme à plusieurs variables à coefficients dans K.



Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ?

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile

Kabanets & Impagliazzo, 2003 : soit NEXP n'a pas de circuits booléens de taille polynomiale, soit le permanent n'a pas de circuits arithmétiques de taille polynomiale.

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ?

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques de profondeur constante (fan-in arbitraire) et de taille polynomiale ?

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ trop difficile

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de profondeur constante (fan-in arbitraire) et de taille polynomiale ?

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ NON ! (Allender et Gore, 1994)

Questions

- Question centrale : le permanent a-t-il des circuits arithmétiques de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ trop difficile
- Le permanent a-t-il des circuits arithmétiques uniformes de profondeur constante (fan-in arbitraire) et de taille polynomiale ? $\longrightarrow$ NON ! (Allender et Gore, 1994)
- Et pour une profondeur non constante ?
  $\longrightarrow$ Le permanent n'a pas de circuits arithmétiques uniformes de profondeur $o(\log \log n)$ (fan-in arbitraire) et de taille polynomiale.

Proposition

Le permanent n'a pas de circuits arithmétiques uniformes de taille polynomiale et de profondeur $o(\log \log n)$.

Preuve (de loin)

1. Si le permanent a des circuits de taille polynomiale, alors on sait calculer en temps n^α des produits et sommes de taille exponentielle (Bürgisser 2007, Hesse & Allender 2002).
2. Soit $A \in \mathrm{EXP}$. Par complétude du permanent, il s'exprime comme un permanent de taille exponentielle. Donc A a des circuits de taille exponentielle et de profondeur $o(\log n)$.
3. On évalue ces circuits de profondeur $o(\log n)$ en itérant $o(\log n)$ fois 1. À chaque itération la complexité est élevée à la puissance α.
4. En tout, on décide A en temps $n^{\alpha^{o(\log n)}} = n^{o(n)}$: contradiction avec le théorème de hiérarchie en temps. □

Conclusion

- En complexité booléenne ou algébrique, les bornes inférieures non-uniformes sont des questions centrales et difficiles.

Conclusion

- En complexité booléenne ou algébrique, les bornes inférieures non-uniformes sont des questions centrales et difficiles.

- Piste dérandomisation :
  - étudier les outils récents (extracteurs, expandeurs...)
  - dérandomisation de la nullité d'un polynôme...

Conclusion

- En complexité booléenne ou algébrique, les bornes inférieures non-uniformes sont des questions centrales et difficiles.

- Piste dérandomisation :
  - étudier les outils récents (extracteurs, expandeurs...)
  - dérandomisation de la nullité d'un polynôme...

- Piste algébrique :
  - si un polynôme est évalué aux points entiers par un circuit booléen de taille polynomiale, a-t-il un circuit arithmétique de taille polynomiale ?