IF228 - Calculabilité et Complexité

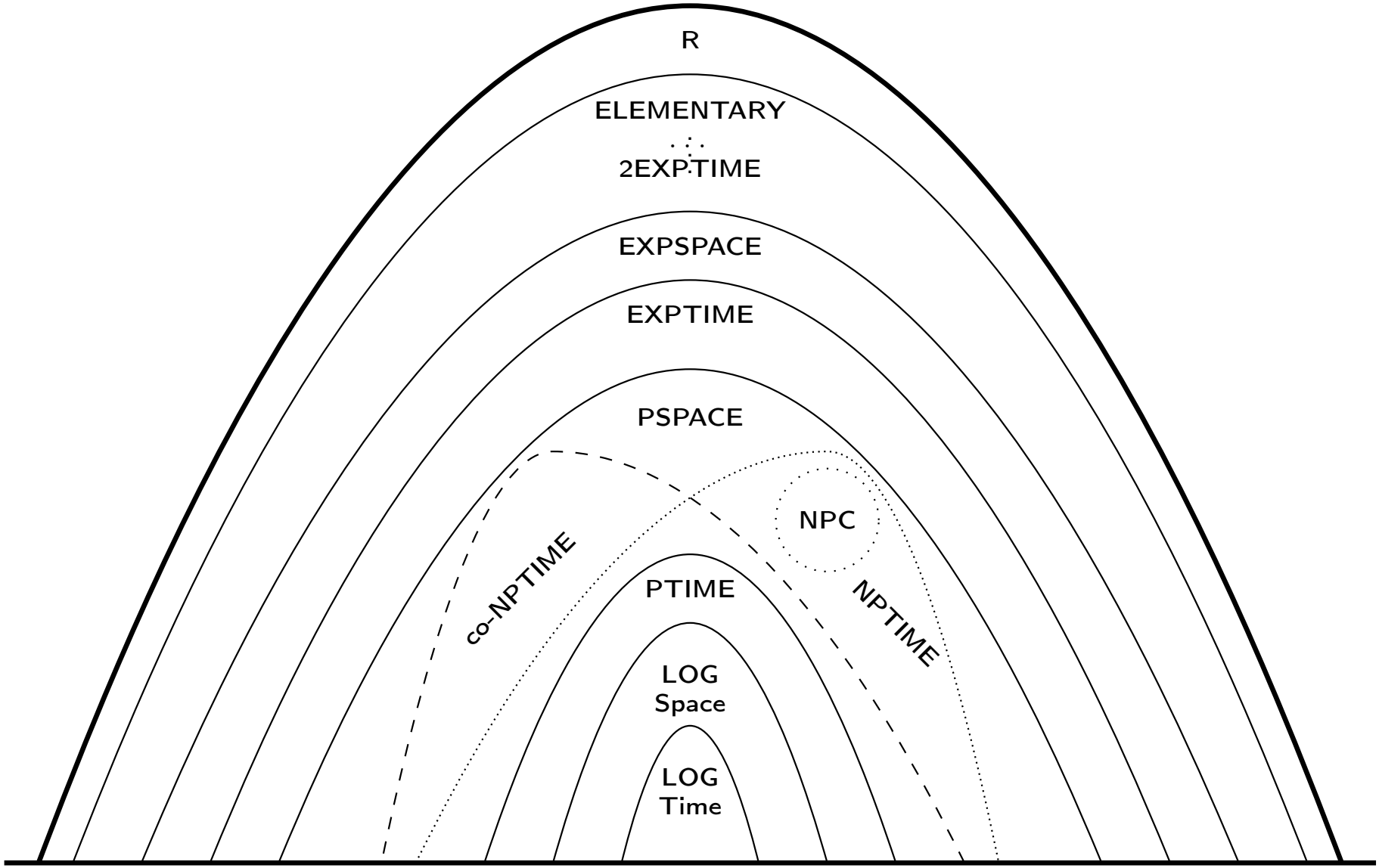
Cours #7 : complexité en espace, PSPACE

Frédéric Herbreteau
frederic.herbreteau@bordeaux-inp.fr
(d'après les supports de Corentin Travers)

8 janvier 2025

Complexité en espace

Classes de complexité



Complexité en espace

Soit M une MT qui **s'arrête pour tout mot d'entrée**.

Complexité en espace

Soit M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

La **complexité en espace** de M est la fonction $T : \mathbb{N} \to \mathbb{N}$ telle que $T(|x|)$ est le **nombre de cellules distinctes modifiées** par M sur l'entrée $x \in \{0,1\}^*$.

Complexité en espace

Soit M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

La **complexité en espace** de M est la fonction $T : \mathbb{N} \to \mathbb{N}$ telle que $T(|x|)$ est le **nombre de cellules distinctes modifiées** par M sur l'entrée $x \in \{0,1\}^*$.

NB :

- complexité **au pire**

Complexité en espace

Soit M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

La **complexité en espace** de M est la fonction $T : \mathbb{N} \to \mathbb{N}$ telle que $T(|x|)$ est le **nombre de cellules distinctes modifiées** par M sur l'entrée $x \in \{0,1\}^*$.

NB :

- complexité **au pire**
- M non-déterministe : **maximum** des cellules utilisées sur **chaque branche** de l'arbre d'exécution de M sur x

Complexité en espace

Soit M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

La **complexité en espace** de M est la fonction $T : \mathbb{N} \to \mathbb{N}$ telle que $T(|x|)$ est le **nombre de cellules distinctes modifiées** par M sur l'entrée $x \in \{0,1\}^*$.

NB :

- complexité **au pire**
- M non-déterministe : **maximum** des cellules utilisées sur **chaque branche** de l'arbre d'exécution de M sur x
- si M est de complexité temporelle $T(n)$ et de complexité en espace $S(n)$, alors $S(n) \leq T(n)$ pour tout $n \in \mathbb{N}$.

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace :

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(1)$

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(1)$

- Une machine qui calcule la longueur de son mot d'entrée w, en binaire, sur un 2ème ruban ?

  $\rightarrow$ Complexité en espace :

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(1)$

- Une machine qui calcule la longueur de son mot d'entrée w, en binaire, sur un 2ème ruban ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(\log_2 |w|)$

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(1)$

- Une machine qui calcule la longueur de son mot d'entrée w, en binaire, sur un 2ème ruban ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(\log_2 |w|)$

- Une machine qui détecte si son mot d'entrée w est un palindrome en effaçant les lettres qui correspondent ?

  $\rightarrow$ Complexité en espace :

Quelques exemples

- Une machine qui lit son ruban de gauche à droite pour détecter si son mot d'entrée contient un 0 ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(1)$

- Une machine qui calcule la longueur de son mot d'entrée w, en binaire, sur un 2ème ruban ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(\log_2 |w|)$

- Une machine qui détecte si son mot d'entrée w est un palindrome en effaçant les lettres qui correspondent ?

  $\rightarrow$ Complexité en espace : $\mathcal{O}(|w|)$

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Exemple :

- $DSPACE(\log_2 n)$: langages décidables en espace log.

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Exemple :

- $DSPACE(\log_2 n)$: langages décidables en espace log.
- $DSPACE(n^2)$: langages décidables en espace quadratique

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Exemple :

- $DSPACE(\log_2 n)$: langages décidables en espace log.
- $DSPACE(n^2)$: langages décidables en espace quadratique
- $DSPACE(2^n)$: langages décidables en espace exponentiel

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Exemple :

- $DSPACE(\log_2 n)$: langages décidables en espace log.
- $DSPACE(n^2)$: langages décidables en espace quadratique
- $DSPACE(2^n)$: langages décidables en espace exponentiel

$$DSPACE(\log_2 n) \subseteq DSPACE(n^2) \subseteq DSPACE(2^n)$$

DSPACE

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** DSPACE(T(n)) si et seulement si L est décidé par une MT **déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Exemple :

- $DSPACE(\log_2 n)$: langages décidables en espace log.
- $DSPACE(n^2)$: langages décidables en espace quadratique
- $DSPACE(2^n)$: langages décidables en espace exponentiel

$$DSPACE(\log_2 n) \subseteq DSPACE(n^2) \subseteq DSPACE(2^n)$$

Remarque : $DTIME(T(n)) \subseteq DSPACE(T(n))$

Logique prop. quantifiée

- **Logique propositionnelle quantifiée :**
  - variables propositionnelles : $VP = \{p, q, r, \dots\}$
  - connecteurs booléens : $\wedge, \vee, \neg, \implies, \iff$
  - quantificateurs : $\forall, \exists$

  Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$

Logique prop. quantifiée

- **Logique propositionnelle quantifiée :**
  - variables propositionnelles : $VP = \{p, q, r, \dots\}$
  - connecteurs booléens : $\wedge, \vee, \neg, \implies, \iff$
  - quantificateurs : $\forall, \exists$

  Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$

- formules **closes** : toutes les variables sont quantifiées **une et une seule fois**

Logique prop. quantifiée

- **Logique propositionnelle quantifiée :**
  - variables propositionnelles : $VP = \{p, q, r, \dots\}$
  - connecteurs booléens : $\wedge, \vee, \neg, \implies, \iff$
  - quantificateurs : $\forall, \exists$

  Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$

- formules **closes** : toutes les variables sont quantifiées **une et une seule fois**

- formules en **forme prénexe** : les quantificateurs sont **en tête** de la formule

Logique prop. quantifiée

- **Logique propositionnelle quantifiée :**
  - variables propositionnelles : $VP = \{p, q, r, \dots\}$
  - connecteurs booléens : $\wedge, \vee, \neg, \implies, \iff$
  - quantificateurs : $\forall, \exists$

  Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$

- formules **closes** : toutes les variables sont quantifiées **une et une seule fois**

- formules en **forme prénexe** : les quantificateurs sont **en tête** de la formule

- **Valuation** v : $VP \to \{0, 1\}$

  Exemple : v_1 : $p \mapsto 1,\ q \mapsto 1,\ r \mapsto 0$

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

ssi $\forall q.\exists r.(q \vee \neg r)$ vraie

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

ssi $\forall q.\exists r.(q \vee \neg r)$ vraie

ssi $\exists r.(q \vee \neg r)$ vraie pour $q \mapsto 0$ **et** pour $q \mapsto 1$

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

ssi $\forall q.\exists r.(q \vee \neg r)$ vraie

ssi $\exists r.(q \vee \neg r)$ vraie pour $q \mapsto 0$ **et** pour $q \mapsto 1$

ssi $\exists r.(\neg r)$ vraie

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

ssi $\forall q.\exists r.(q \vee \neg r)$ vraie

ssi $\exists r.(q \vee \neg r)$ vraie pour $q \mapsto 0$ **et** pour $q \mapsto 1$

ssi $\exists r.(\neg r)$ vraie

ssi $\neg r$ vraie pour $r \mapsto 0$ **ou** pour $r \mapsto 1$

Valeur d'une QBF

Une formule est **vraie** dans une valuation v :

- $\forall x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **et** dans $v[x \mapsto 1]$
- $\exists x.\phi$: si ϕ est vraie dans $v[x \mapsto 0]$ **ou** dans $v[x \mapsto 1]$

Exemple : $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$ est vraie

ssi $\forall q.\exists r.(p \implies q \vee \neg r)$ vraie pour $p \mapsto 0$ **et** pour $p \mapsto 1$

ssi $\forall q.\exists r.(q \vee \neg r)$ vraie

ssi $\exists r.(q \vee \neg r)$ vraie pour $q \mapsto 0$ **et** pour $q \mapsto 1$

ssi $\exists r.(\neg r)$ vraie

ssi $\neg r$ vraie pour $r \mapsto 0$ **ou** pour $r \mapsto 1$

✓

QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

- Qu'est-ce qu'un certificat pour $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$?

QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

- Qu'est-ce qu'un certificat pour $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$?

  $\rightarrow$ on ne sait pas si QSAT est dans NP

QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

- Qu'est-ce qu'un certificat pour $\forall p.\forall q.\exists r.(p \implies q \vee \neg r)$?

  $\rightarrow$ on ne sait pas si QSAT est dans NP

- On peut résoudre QSAT en **espace polynomial** :

```
function qsat(φ, v):
  match φ with
    ∀x.ψ: return qsat(ψ, v[x ↦ 0]) ∧ qsat(ψ, v[x ↦ 1])
    ∃x.ψ: return qsat(ψ, v[x ↦ 0]) ∨ qsat(ψ, v[x ↦ 1])
    else: return value(φ, v)
```

Accessibilité dans un graphe

PATH

ENTRÉE : un graphe orienté fini $G = (V, E)$ et $v_1, v_2 \in V$

QUESTION : existe-t-il un chemin de v_1 à v_2 dans G ?

Accessibilité dans un graphe

PATH

ENTRÉE : un graphe orienté fini $G = (V, E)$ et $v_1, v_2 \in V$

QUESTION : existe-t-il un chemin de v_1 à v_2 dans G ?

- Le problème PATH est-il décidable ?

Accessibilité dans un graphe

PATH

ENTRÉE : un graphe orienté fini $G = (V, E)$ et $v_1, v_2 \in V$

QUESTION : existe-t-il un chemin de v_1 à v_2 dans G ?

- Le problème PATH est-il décidable ?
  $\rightarrow$ **oui :** énumérer les chemins de longueur au plus $|V|$

Accessibilité dans un graphe

PATH

ENTRÉE : un graphe orienté fini $G = (V, E)$ et $v_1, v_2 \in V$

QUESTION : existe-t-il un chemin de v_1 à v_2 dans G ?

- Le problème PATH est-il décidable ?
  $\rightarrow$ **oui :** énumérer les chemins de longueur au plus $|V|$
- Algorithme **dichotomique** en espace $\mathcal{O}((\log_2 |V|)^2)$

```
function has_path(v1, v2, k):
    if k == 0: return (v1 == v2)
    elif k == 1: return (v1, v2) ∈ E
    else: for each u ∈ V:
            if has_path(v1, u, ⌊k/2⌋) and has_path(u, v2, ⌈k/2⌉):
                return true
          return false
```

  $\rightarrow$ $\mathcal{O}(\log_2 |V|)$ récursions et $\mathcal{O}(\log_2 |V|)$ bits pour v_1, v_2, k, u

  $\rightarrow$ tester has_path(v_1, v_2, k) pour $k \in [0; |V|]$ (espace réutilisé)

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

- Le problème BOUNDED-ACCEPT est-il décidable ?

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

- Le problème BOUNDED-ACCEPT est-il décidable ?
  $\rightarrow$ **oui :** il suffit de simuler k transitions de M sur w

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

- Le problème BOUNDED-ACCEPT est-il décidable ?
  $\rightarrow$ **oui :** il suffit de simuler k transitions de M sur w

- Taille max d'une configuration en fonction de $\langle k \rangle$?

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

- Le problème BOUNDED-ACCEPT est-il décidable ?
  $\rightarrow$ **oui :** il suffit de simuler k transitions de M sur w

- Taille max d'une configuration en fonction de $\langle k \rangle$?
  $\rightarrow$ $k = \mathcal{O}(2^{|\langle k \rangle|})$ (hyp : $k \geq |w|$)

Terminaison en temps borné

BOUNDED-ACCEPT

ENTRÉE : une MT déterministe M, un mot w et un entier $k \in \mathbb{N}$ **encodé en binaire**

QUESTION : M s'arrête-t-elle sur w en au plus k transitions ?

- Le problème BOUNDED-ACCEPT est-il décidable ?
  $\rightarrow$ **oui :** il suffit de simuler k transitions de M sur w
- Taille max d'une configuration en fonction de $\langle k \rangle$?
  $\rightarrow$ $k = \mathcal{O}(2^{|\langle k \rangle|})$ (hyp : $k \geq |w|$)
- Simuler M requiert un **espace non polynomial en** $|\langle k \rangle|$

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

Ex :

	$\in$ PSPACE?
SAT	
QSAT	
PATH	
BOUNDED-ACCEPT	

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

Ex :

	$\in$ PSPACE?	
SAT	oui	car dans NP
QSAT		
PATH		
BOUNDED-ACCEPT		

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

Ex :

	$\in$ PSPACE?	
SAT	oui	car dans NP
QSAT	oui	cf. fonction qsat
PATH		
BOUNDED-ACCEPT		

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

Ex :

	$\in$ PSPACE?	
SAT	oui	car dans NP
QSAT	oui	cf. fonction qsat
PATH	oui	cf. fonction has_path
BOUNDED-ACCEPT		

La classe PSPACE

$$\text{PSPACE} = \bigcup_{c \geq 1} \text{DSPACE}(n^c)$$

Classe des langages décidables en **espace polynomial** par une MT **déterministe**

Ex :

	$\in$ PSPACE?	
SAT	oui	car dans NP
QSAT	oui	cf. fonction qsat
PATH	oui	cf. fonction has_path
BOUNDED-ACCEPT	???	EXPTIME-complet

Espace et non-déterminisme

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** NSPACE(T(n)) si et seulement si L est décidé par une MT **non-déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Espace et non-déterminisme

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** NSPACE(T(n)) si et seulement si L est décidé par une MT **non-déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Théorème (Savitch, 1970)

Pour toute fonction $T : \mathbb{N} \to \mathbb{N}$ *telle que* $T(n) \geq n$
$NSPACE(T(n)) \subseteq DSPACE(T^2(n))$

Espace et non-déterminisme

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** NSPACE(T(n)) si et seulement si L est décidé par une MT **non-déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Théorème (Savitch, 1970)

Pour toute fonction $T : \mathbb{N} \to \mathbb{N}$ *telle que* $T(n) \geq n$
$NSPACE(T(n)) \subseteq DSPACE(T^2(n))$

Rappel : $NTIME(T(n)) \subseteq DTIME(2^{\mathcal{O}(T(n))})$

Espace et non-déterminisme

Définition

Soit $T : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage L est dans la **classe** NSPACE(T(n)) si et seulement si L est décidé par une MT **non-déterministe** en espace $c \cdot T(n)$, où $c > 0$ est une constante.

Théorème (Savitch, 1970)

Pour toute fonction $T : \mathbb{N} \to \mathbb{N}$ *telle que* $T(n) \geq n$
$NSPACE(T(n)) \subseteq DSPACE(T^2(n))$

Rappel : $NTIME(T(n)) \subseteq DTIME(2^{\mathcal{O}(T(n))})$

$\to$ Une MT **déterministe** peut **réutiliser** l'espace, pas le temps

Théorème de Savitch

Preuve

Théorème de Savitch

Preuve

- Soit $L \in NSPACE(T(n))$ avec $T(n) \geq n$

Théorème de Savitch

Preuve

- Soit $L \in NSPACE(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$

Théorème de Savitch

Preuve

- Soit $L \in \mathit{NSPACE}(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$
- **Graphe des configurations bornées par** $T(|w|)$ G de M sur w ?

Théorème de Savitch

Preuve

- Soit $L \in \mathit{NSPACE}(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$
- **Graphe des configurations bornées par** $T(|w|)$ G de M sur w ?
  $\rightarrow$ Taille ?

Théorème de Savitch

Preuve

- Soit $L \in \mathit{NSPACE}(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$
- **Graphe des configurations bornées par** $T(|w|)$ G de M sur w ?
  $\rightarrow$ Taille ? $2^{\mathcal{O}(T(|w|))}$ sommets

Théorème de Savitch

Preuve

- Soit $L \in NSPACE(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$
- **Graphe des configurations bornées par** $T(|w|)$ G de M sur w ?
  $\rightarrow$ Taille ? $2^{\mathcal{O}(T(|w|))}$ sommets

  $\rightarrow$ $w \in L$ ssi il existe un **chemin dans** G de (q_0, w) à (q_{yes}, w') avec $w' \in \{0,1\}^{T(|w|)}$

Théorème de Savitch

Preuve

- Soit $L \in NSPACE(T(n))$ avec $T(n) \geq n$
- Il existe une MT **non-déterministe** M qui décide L en espace $T(|w|)$ sur entrée $w \in \{0,1\}^*$
- **Graphe des configurations bornées par** $T(|w|)$ G de M sur w ?
  $\rightarrow$ Taille ? $2^{\mathcal{O}(T(|w|))}$ sommets

  $\rightarrow$ $w \in L$ ssi il existe un **chemin dans** G de (q_0, w) à (q_{yes}, w') avec $w' \in \{0,1\}^{T(|w|)}$
- Décidé par "has_path" en **espace** $T^2(n)$ **déterministe**

$$\mathcal{O}\left(\left(\log_2 2^{\mathcal{O}(T(|w|))}\right)^2\right) = \mathcal{O}\left(T^2(|w|)\right)$$

PSPACE-difficile, PSPACE-complet

Définition

Soit $A \subseteq \{0,1\}^*$. On dit que

- A est **PSPACE-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in$ PSPACE

PSPACE-difficile, PSPACE-complet

Définition

Soit $A \subseteq \{0,1\}^*$. On dit que

- A est **PSPACE-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in$ PSPACE
- A est **PSPACE-complet** ssi A est PSPACE-difficile **et** $A \in$ PSPACE

PSPACE-difficile, PSPACE-complet

Définition

Soit $A \subseteq \{0,1\}^*$. On dit que

- A est **PSPACE-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in$ PSPACE
- A est **PSPACE-complet** ssi A est PSPACE-difficile **et** $A \in$ PSPACE

Remarque :

- Un langage A PSPACE-complet, est au moins aussi difficile que **tous les autres langages dans PSPACE**

PSPACE-difficile, PSPACE-complet

Définition

Soit $A \subseteq \{0,1\}^*$. On dit que

- A est **PSPACE-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in$ PSPACE
- A est **PSPACE-complet** ssi A est PSPACE-difficile **et** $A \in$ PSPACE

Remarque :

- Un langage A PSPACE-complet, est au moins aussi difficile que **tous les autres langages dans PSPACE**
- Un langage PSPACE-difficile est **plus difficile que les langages NP ou co-NP** (hyp : NP $\neq$ PSPACE)

Retour sur QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

Théorème

QSAT est PSPACE-complet

Retour sur QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

Théorème

QSAT est PSPACE-complet

Preuve

- QSAT $\in$ PSPACE (cf. fonction qsat)

Retour sur QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

Théorème

QSAT est PSPACE-complet

Preuve

- QSAT $\in$ PSPACE (cf. fonction qsat)
- QSAT est PSPACE-difficile

Retour sur QSAT

QSAT

ENTRÉE : une formule propositionnelle quantifiée ϕ close et en forme prénexe, un seul quantificateur par variable

QUESTION : ϕ est-elle **vraie** ?

Théorème

QSAT est PSPACE-complet

Preuve

- QSAT $\in$ PSPACE (cf. fonction qsat)
- QSAT est PSPACE-difficile
  $\rightarrow$ Mq tout langage PSPACE se réduit à QSAT en temps polynomial

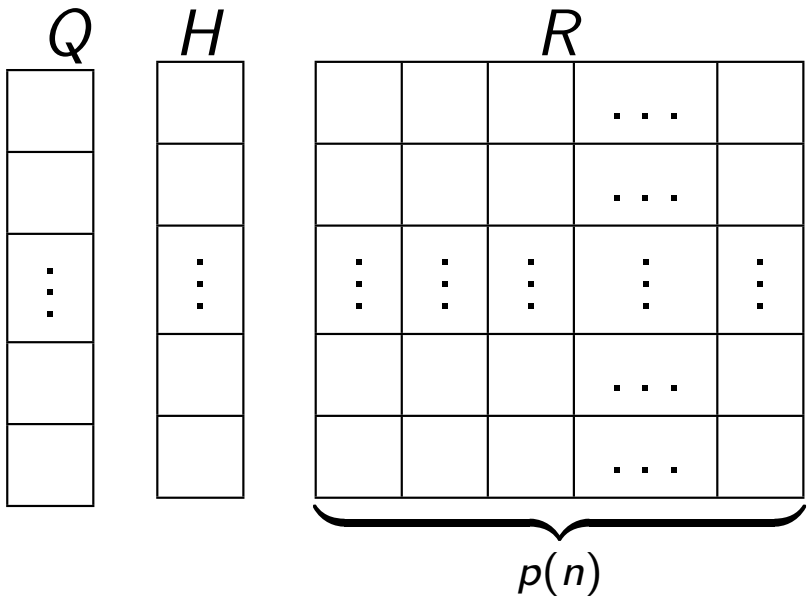
QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

1ère idée : encoder un tableau par une formule QBF ϕ tq ϕ **est vraie ssi** M **accepte** w (à la Cook-Levin)



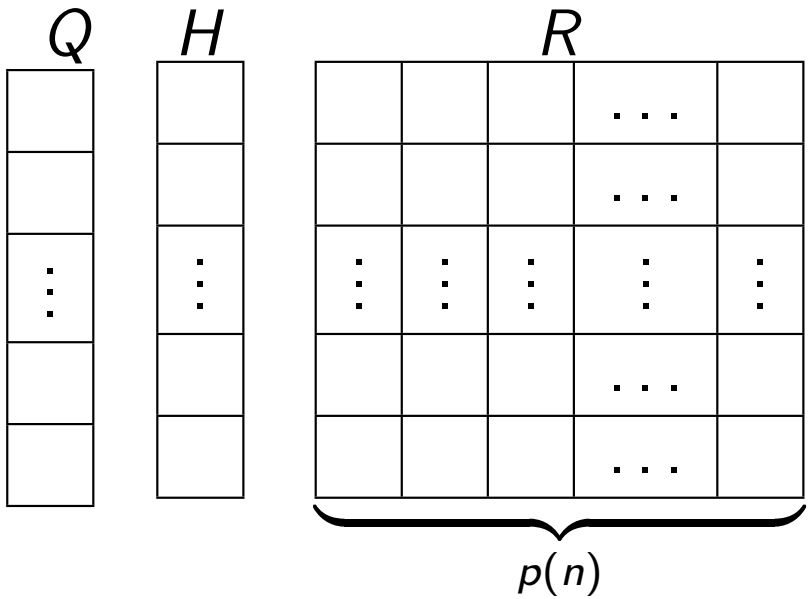
QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

1ère idée : encoder un tableau par une formule QBF ϕ tq ϕ **est vraie ssi** M **accepte** w (à la Cook-Levin)



Pourquoi cela ne marche-t-il pas ?

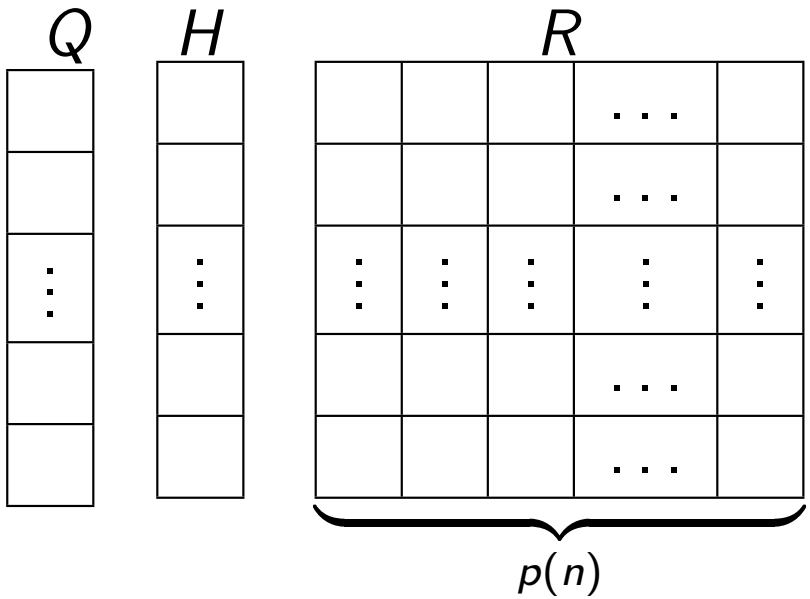
QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

1ère idée : encoder un tableau par une formule QBF ϕ tq ϕ **est vraie ssi** M **accepte** w (à la Cook-Levin)



Pourquoi cela ne marche-t-il pas ?

$\rightarrow$ jusqu'à $2^{\mathcal{O}(p(n))}$ **configurations**

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

2ème idée : **dichotomie** sur l'exécution de M sur w

$$(q_0, w) \longrightarrow \cdots \xrightarrow{k} \bullet \longrightarrow \cdots \longrightarrow (q_{yes}, w')$$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$
$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

2ème idée : **dichotomie** sur l'exécution de M sur w

$$\frac{(q_0, w) \overset{\lfloor \frac{k}{2} \rfloor}{\longrightarrow \cdots \longrightarrow} (q, x) \overset{\lceil \frac{k}{2} \rceil}{\longrightarrow \cdots \longrightarrow} (q_{yes}, w')}{\overset{k}{}}$$

- si $k = 0$: $\phi_{c_1, c_2, 0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1, c_2, 1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1, c_2, k} = \exists c. \left(\phi_{c_1, c, \lfloor \frac{k}{2} \rfloor} \wedge \phi_{c, c_2, \lceil \frac{k}{2} \rceil} \right)$

NB : M accepte w ssi $\phi_{(q_0, w), (q_{yes}, w'), 2^{c \cdot p(n)}}$ est vraie pour une constante $c \in \mathbb{N}$ et $w' \in \{0,1\}^{c \cdot p(n)}$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

2ème idée : **dichotomie** sur l'exécution de M sur w

$$(q_0, w) \overset{k}{\longrightarrow} \cdots \longrightarrow (q, x) \overset{k}{\longrightarrow} \cdots \longrightarrow (q_{yes}, w')$$

- si $k = 0$: $\phi_{c_1,c_2,0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1,c_2,1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1,c_2,k} = \exists c. \left(\phi_{c_1,c,\lfloor \frac{k}{2} \rfloor} \wedge \phi_{c,c_2,\lceil \frac{k}{2} \rceil} \right)$

NB : M accepte w ssi $\phi_{(q_0,w),(q_{yes},w'),2^{c \cdot p(n)}}$ est vraie pour une constante $c \in \mathbb{N}$ et $w' \in \{0,1\}^{c \cdot p(n)}$

Pourquoi cela ne marche-t-il pas ?

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

2ème idée : **dichotomie** sur l'exécution de M sur w

$$(q_0, w) \longrightarrow \cdots \longrightarrow (q, x) \overset{k}{\longrightarrow} \cdots \longrightarrow (q_{yes}, w')$$

- si $k = 0$: $\phi_{c_1,c_2,0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1,c_2,1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1,c_2,k} = \exists c. \left(\phi_{c_1,c,\lfloor \frac{k}{2} \rfloor} \wedge \phi_{c,c_2,\lceil \frac{k}{2} \rceil}\right)$

NB : M accepte w ssi $\phi_{(q_0,w),(q_{yes},w'),2^{c \cdot p(n)}}$ est vraie pour une constante $c \in \mathbb{N}$ et $w' \in \{0,1\}^{c \cdot p(n)}$

Pourquoi cela ne marche-t-il pas ?

$\rightarrow$ chaque récursion **double** la taille de ϕ jusqu'à $2^{\mathcal{O}(p(n))}$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

3ème idée : **dichotomie** sur l'exécution de M sur w

$$\overbrace{\underbrace{(q_0, w) \longrightarrow \cdots \longrightarrow}_{\lfloor \frac{k}{2} \rfloor} (q, x) \underbrace{\longrightarrow \cdots \longrightarrow (q_{yes}, w')}_{\lceil \frac{k}{2} \rceil}}^{k}$$

- si $k = 0$: $\phi_{c_1, c_2, 0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1, c_2, 1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1, c_2, k} =$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

3ème idée : **dichotomie** sur l'exécution de M sur w

$$\frac{\overset{k}{\overbrace{\underbrace{(q_0, w) \longrightarrow \cdots \longrightarrow}_{\lfloor \frac{k}{2} \rfloor} (q, x) \underbrace{\longrightarrow \cdots \longrightarrow (q_{yes}, w')}_{\lceil \frac{k}{2} \rceil}}}}{}$$

- si $k = 0$: $\phi_{c_1, c_2, 0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1, c_2, 1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1, c_2, k} = \exists c. \forall (c', c'') \in \{(c_1, c), (c, c_2)\}. \left(\phi_{c', c'', \frac{k}{2}}\right)$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

3ème idée : **dichotomie** sur l'exécution de M sur w

$$\frac{(q_0, w) \overset{\lfloor \frac{k}{2} \rfloor}{\longrightarrow \cdots \longrightarrow} (q, x) \overset{\lceil \frac{k}{2} \rceil}{\longrightarrow \cdots \longrightarrow} (q_{yes}, w')}{k}$$

- si $k = 0$: $\phi_{c_1, c_2, 0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1, c_2, 1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1, c_2, k} = \exists c. \forall (c', c'') \in \{(c_1, c), (c, c_2)\}. \left(\phi_{c', c'', \frac{k}{2}}\right)$
- M accepte w ssi $\phi_{(q_0, w), (q_{yes}, w'), 2^{c \cdot p(n)}}$ est vraie pour une constante $c \in \mathbb{N}$ et $w' \in \{0,1\}^{c \cdot p(n)}$

QSAT est PSPACE-difficile

Preuve

Soit $L \subseteq \{0,1\}^*$ dans PSPACE et $w \in \{0,1\}^*$

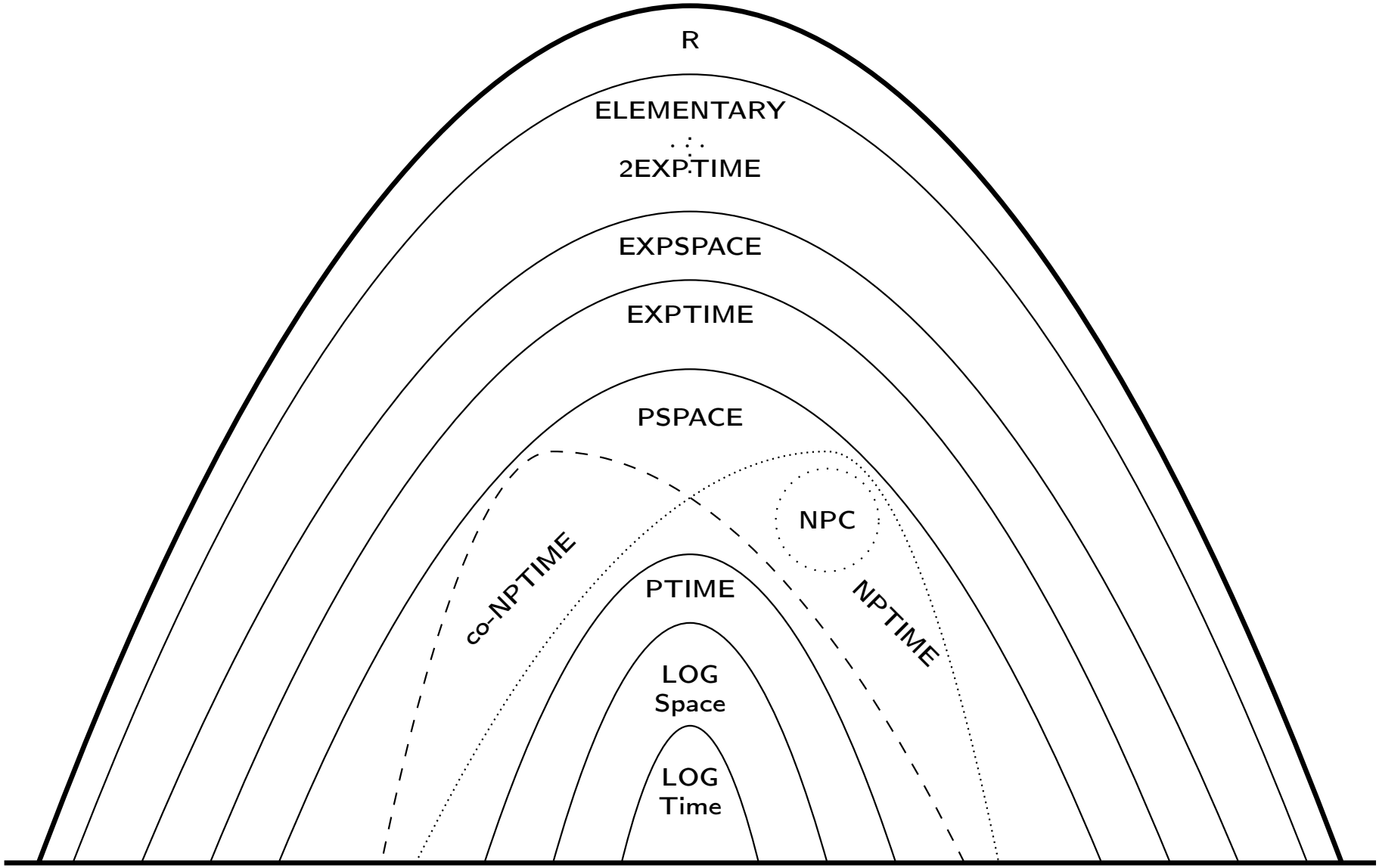
$\rightarrow$ décidé par MT dét. M en **espace polynomial** $\mathcal{O}(p(n))$

3ème idée : **dichotomie** sur l'exécution de M sur w

$$\frac{\overset{k}{\overbrace{\underbrace{(q_0,w) \longrightarrow \cdots \longrightarrow}_{\lfloor \frac{k}{2} \rfloor} (q,x) \underbrace{\longrightarrow \cdots \longrightarrow (q_{yes},w')}_{\lceil \frac{k}{2} \rceil}}}}{}$$

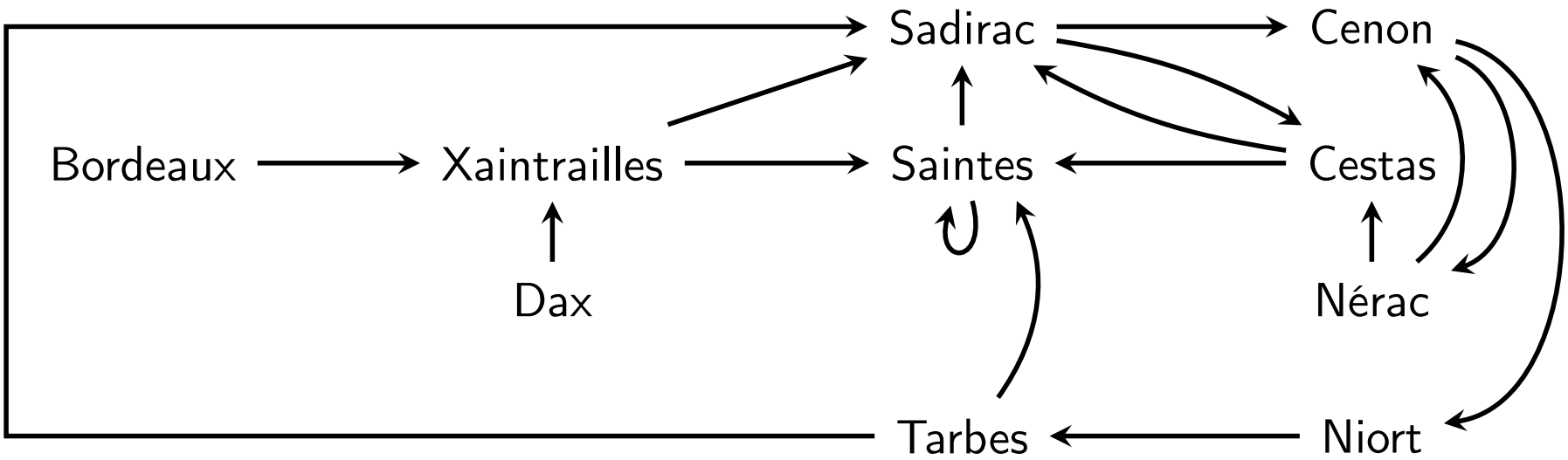
- si $k = 0$: $\phi_{c_1,c_2,0}$ encode $c_1 = c_2$
- si $k = 1$: $\phi_{c_1,c_2,1}$ encode $c_1 \rightarrow c_2$ (cf. ϕ_T)
- si $k > 1$: $\phi_{c_1,c_2,k} = \exists c. \forall (c',c'') \in \{(c_1,c),(c,c_2)\}. \left(\phi_{c',c'',\frac{k}{2}}\right)$
- M accepte w ssi $\phi_{(q_0,w),(q_{yes},w'),2^{c.p(n)}}$ est vraie pour une constante $c \in \mathbb{N}$ et $w' \in \{0,1\}^{c.p(n)}$
- $\phi_{(q_0,w),(q_{yes},w'),2^{c.p(n)}}$ est de taille $\mathcal{O}(p(n))$

Classes de complexité



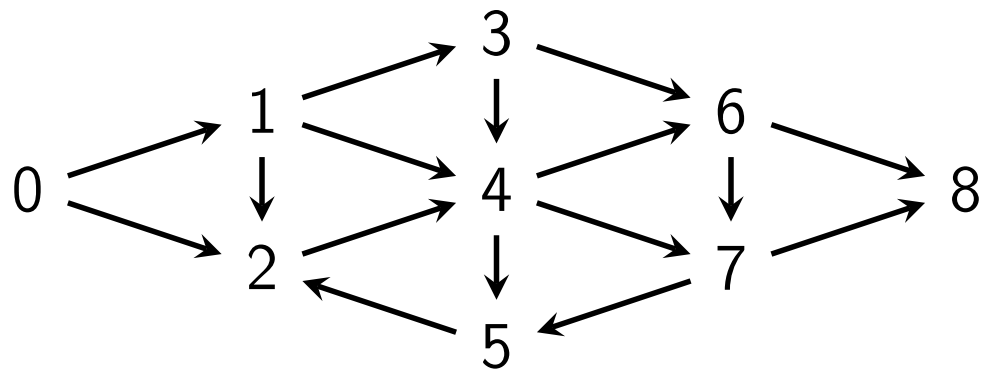
Jeux à 2 joueurs

"Géographie"

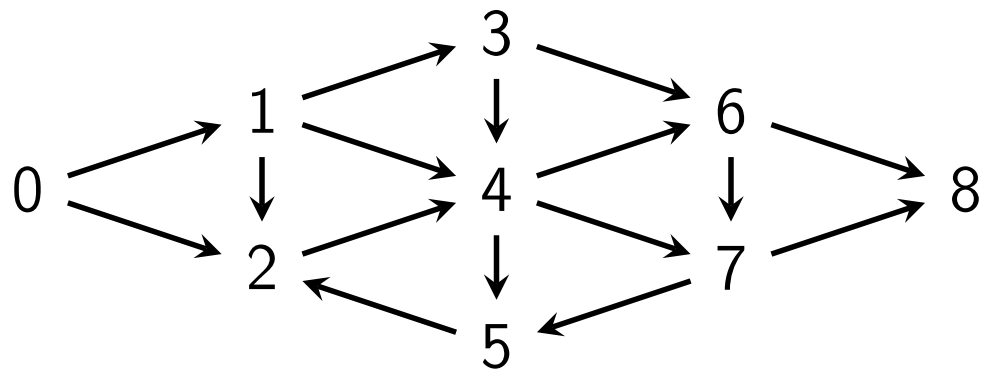


- Chaque joueur **choisit une ville** à tour de rôle
- On ne doit **pas passer 2 fois par la même ville**
- Le 1er **joueur bloqué a perdu**

"Géographie" généralisé

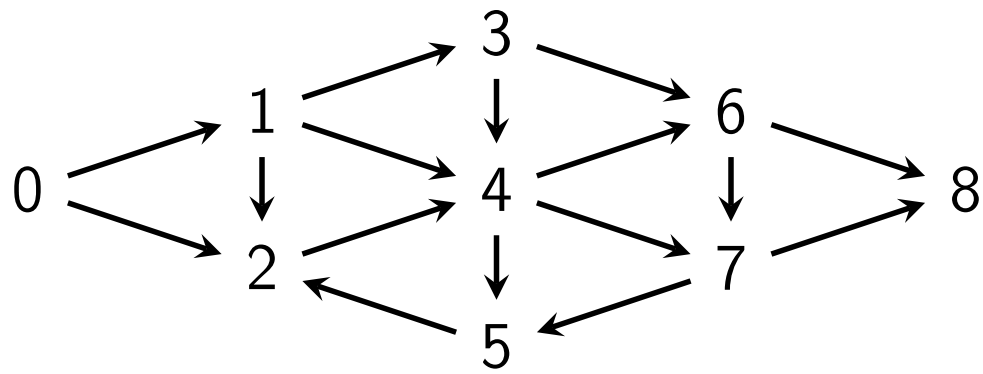


"Géographie" généralisé



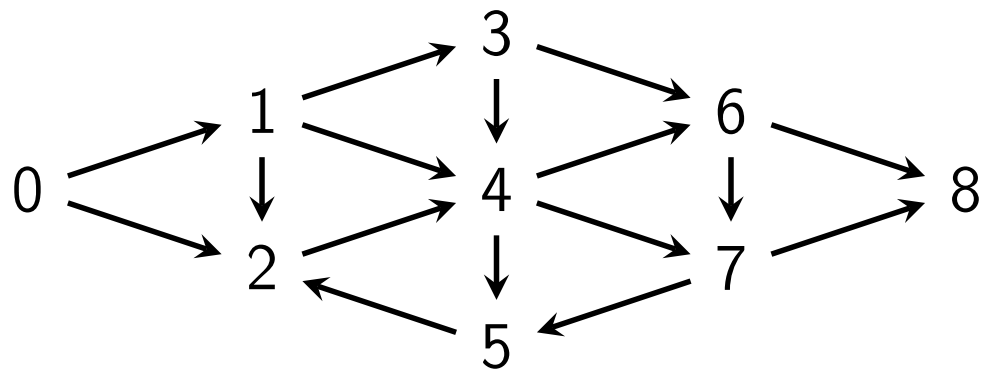
- Partie gagnante : séquence $v_0, \ldots, v_n$ **maximale** de sommets **sans répétition**, tq Joueur 1 joue **le dernier**

"Géographie" généralisé



- Partie gagnante : séquence $v_0, \ldots, v_n$ **maximale** de sommets **sans répétition**, tq Joueur 1 joue **le dernier**
- Stratégie gagnante : fonction $\sigma : V \to V$ tq toute partie avec $v_{i+1} = \sigma(v_i)$ pour i pair, est **gagnante**

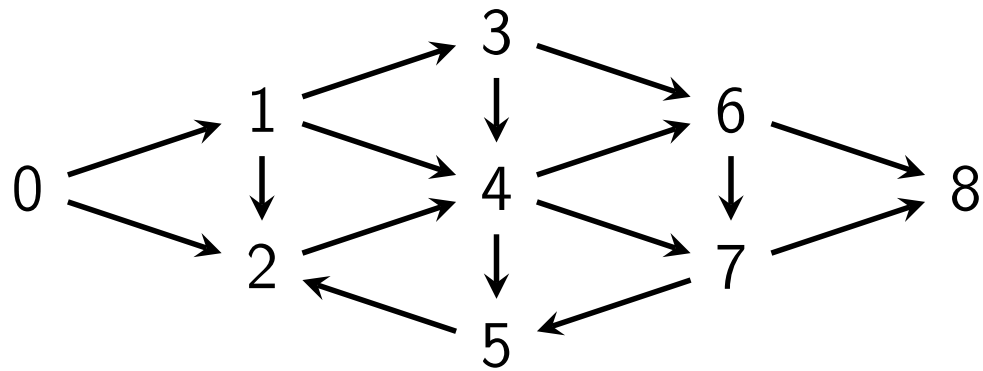
"Géographie" généralisé



- Partie gagnante : séquence $v_0, \ldots, v_n$ **maximale** de sommets **sans répétition**, tq Joueur 1 joue **le dernier**
- Stratégie gagnante : fonction $\sigma : V \to V$ tq toute partie avec $v_{i+1} = \sigma(v_i)$ pour i pair, est **gagnante**

$\to$ stratégie gagnante dans le jeu ci-dessus ?

"Géographie" généralisé



- Partie gagnante : séquence $v_0, \ldots, v_n$ **maximale** de sommets **sans répétition**, tq Joueur 1 joue **le dernier**
- Stratégie gagnante : fonction $\sigma : V \to V$ tq toute partie avec $v_{i+1} = \sigma(v_i)$ pour i pair, est **gagnante**

GEOGRAPHIE

ENTRÉE : graphe orienté $G = (V, E)$ et sommet initial $v_0 \in V$

QUESTION : existe-t-il une stratégie gagnante pour Joueur 1 ?

Complexité de "Géographie"

GEOGRAPHIE

ENTRÉE : graphe orienté $G = (V, E)$ et sommet initial $v_0 \in V$

QUESTION : existe-t-il une stratégie gagnante pour Joueur 1 ?

Théorème

GEOGRAPHIE est PSPACE-complet

Preuve

Complexité de "Géographie"

GEOGRAPHIE

ENTRÉE : graphe orienté $G = (V, E)$ et sommet initial $v_0 \in V$

QUESTION : existe-t-il une stratégie gagnante pour Joueur 1 ?

Théorème

GEOGRAPHIE est PSPACE-complet

Preuve

- GEOGRAPHIE est dans PSPACE

```
function win(G, v):
  if out_degree(v) = 0: return false
  else:
    let N = {v' | v → v'} and G' = G \ {v}
    return ¬(⋀_{v'∈N} win(G', v'))
```

Complexité de "Géographie"

GEOGRAPHIE

ENTRÉE : graphe orienté $G = (V, E)$ et sommet initial $v_0 \in V$

QUESTION : existe-t-il une stratégie gagnante pour Joueur 1 ?

Théorème

GEOGRAPHIE est PSPACE-complet

Preuve

- GEOGRAPHIE est dans PSPACE

```
function win(G, v):
  if out_degree(v) = 0: return false
  else:
    let N = {v' | v → v'} and G' = G \ {v}
    return ¬(⋀_{v'∈N} win(G', v'))
```

$\rightarrow$ win(G, v) ssi il existe une stratégie gagnante en v

Complexité de "Géographie"

GEOGRAPHIE

ENTRÉE : graphe orienté $G = (V, E)$ et sommet initial $v_0 \in V$

QUESTION : existe-t-il une stratégie gagnante pour Joueur 1 ?

Théorème

GEOGRAPHIE est PSPACE-complet

Preuve

- GEOGRAPHIE est dans PSPACE

```
function win(G, v):
  if out_degree(v) = 0: return false
  else:
    let N = {v' | v → v'} and G' = G \ {v}
    return ¬(⋀_{v'∈N} win(G', v'))
```

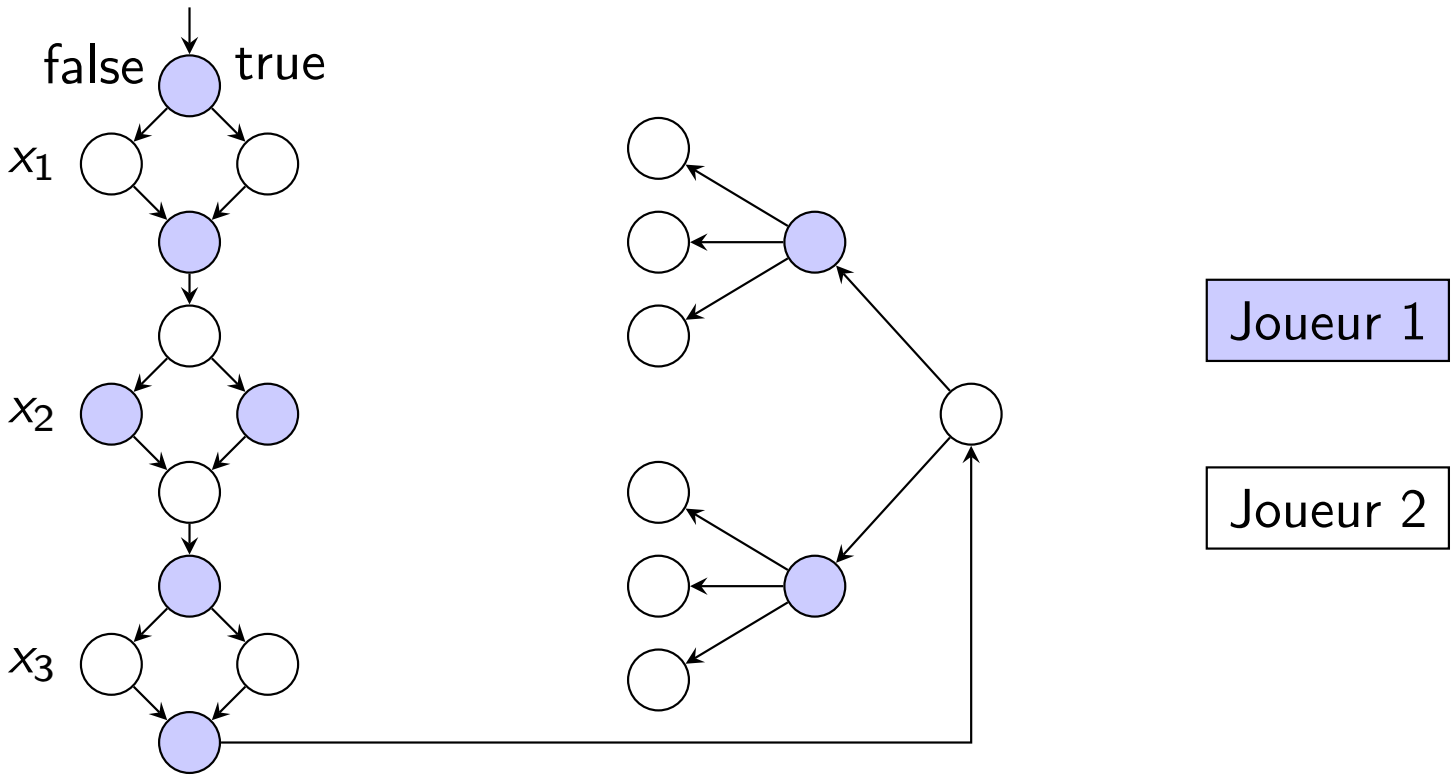
  → win(G, v) ssi il existe une stratégie gagnante en v

  → win calcule en **espace polynomial**

"Géographie" est PSPACE-difficile

Preuve

Ex : $\exists x_1.\forall x_2.\exists x_3.\,((x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3))$



Réduction polynomiale : étant donnée ϕ QBF calcule G tq :

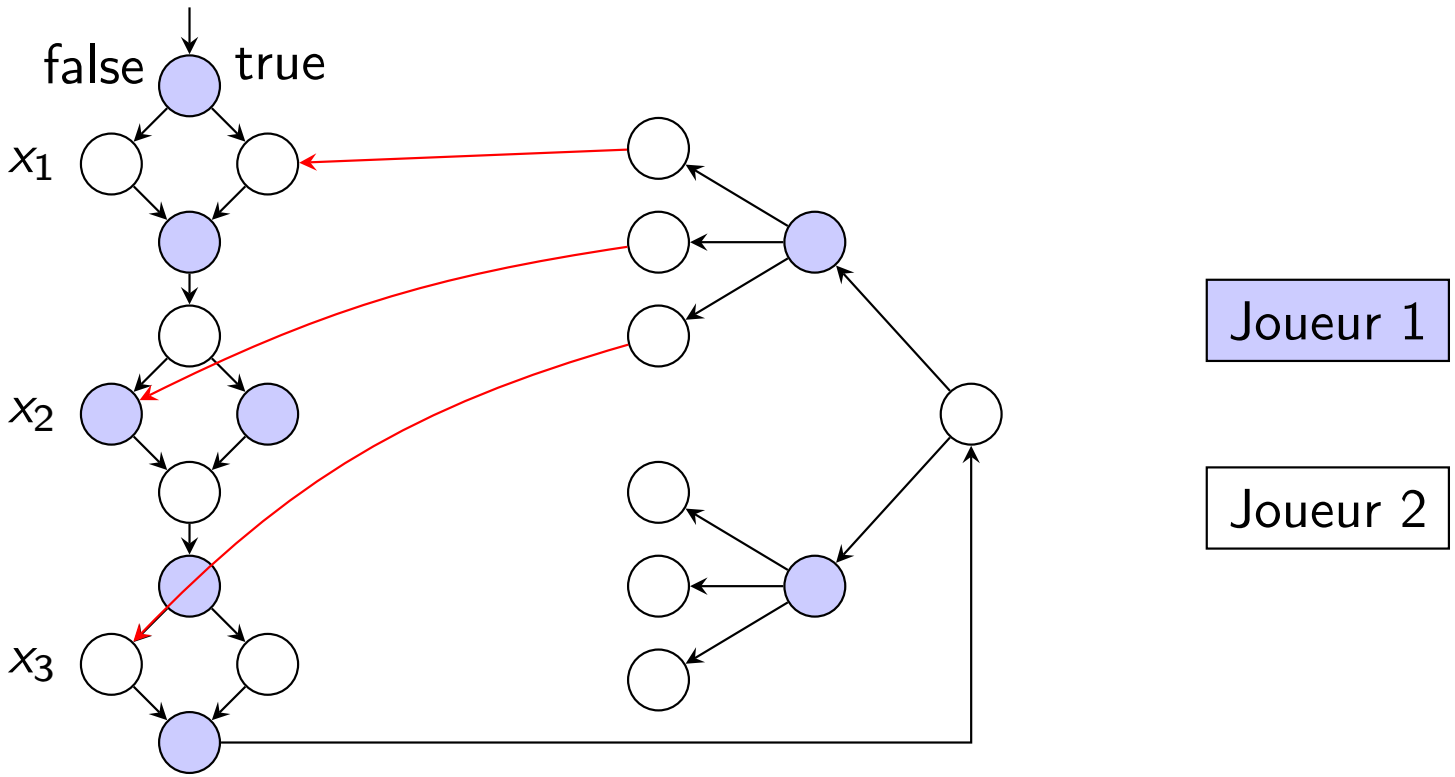
$\rightarrow$ Joueur 1 a une stratégie gagnante dans G ssi ϕ vraie

$\rightarrow$ G calculé en temps $\mathcal{O}(|\phi|)$

"Géographie" est PSPACE-difficile

Preuve

Ex : $\exists x_1.\forall x_2.\exists x_3.\,((x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3))$



Réduction polynomiale : étant donnée ϕ QBF calcule G tq :

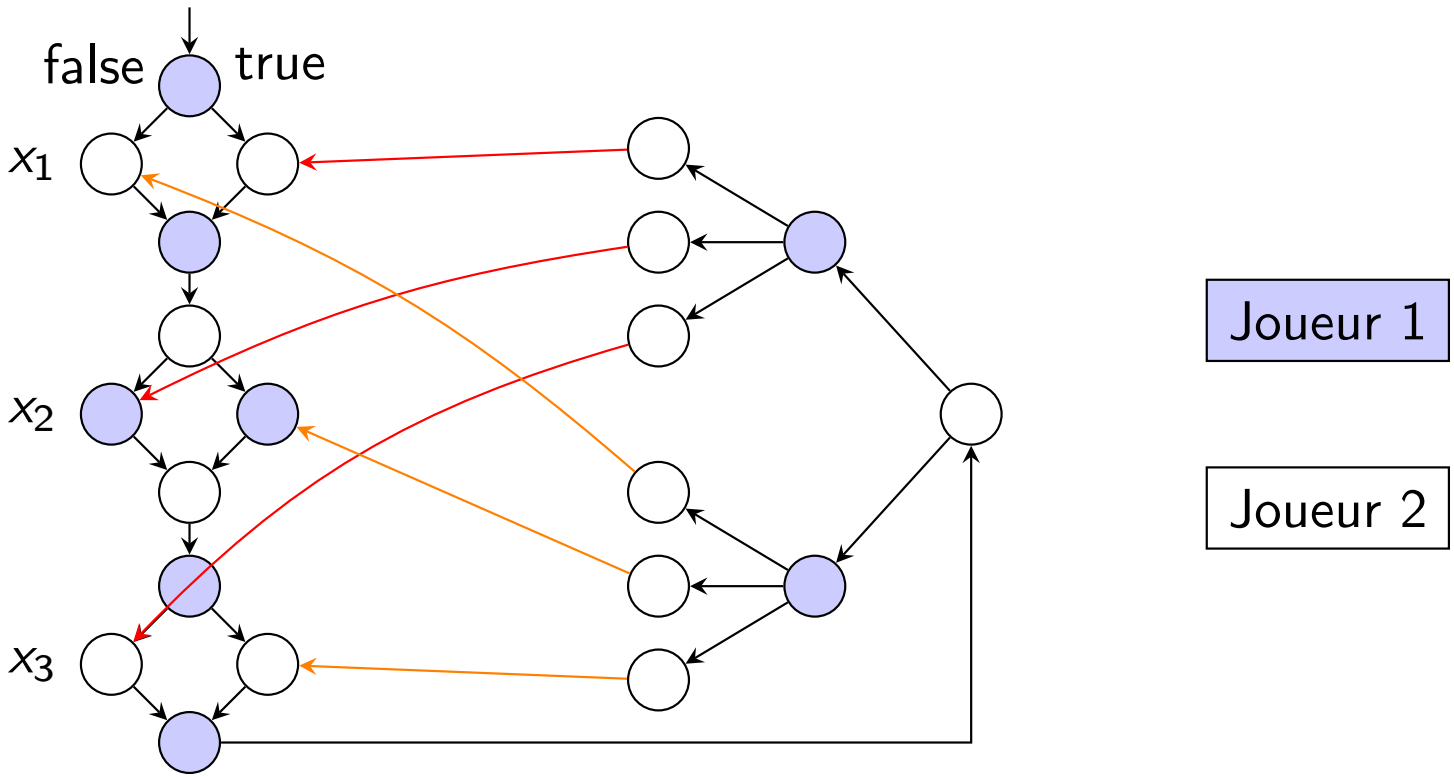
$\rightarrow$ Joueur 1 a une stratégie gagnante dans G ssi ϕ vraie

$\rightarrow$ G calculé en temps $\mathcal{O}(|\phi|)$

"Géographie" est PSPACE-difficile

Preuve

Ex : $\exists x_1.\forall x_2.\exists x_3.\,((x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3))$



Réduction polynomiale : étant donnée ϕ QBF calcule G tq :

$\rightarrow$ Joueur 1 a une stratégie gagnante dans G ssi ϕ vraie

$\rightarrow$ G calculé en temps $\mathcal{O}(|\phi|)$

Un aperçu

- De nombreux jeux (généralisés) sont PSPACE-complet :
  - dames
  - tic-tac-toe
  - hex
  - reversi
  - Lemmings
  - Super Mario Bros
  - ...

Un aperçu

- De nombreux jeux (généralisés) sont PSPACE-complet :
  - dames
  - tic-tac-toe
  - hex
  - reversi
  - Lemmings
  - Super Mario Bros
  - ...
- D'autres sont encore plus durs (EXPTIME-difficile) :
  - les échecs
  - go
  - ...