IF228 - Calculabilité et Complexité

Cours #6 : réduction polynomiale, NP-complétude

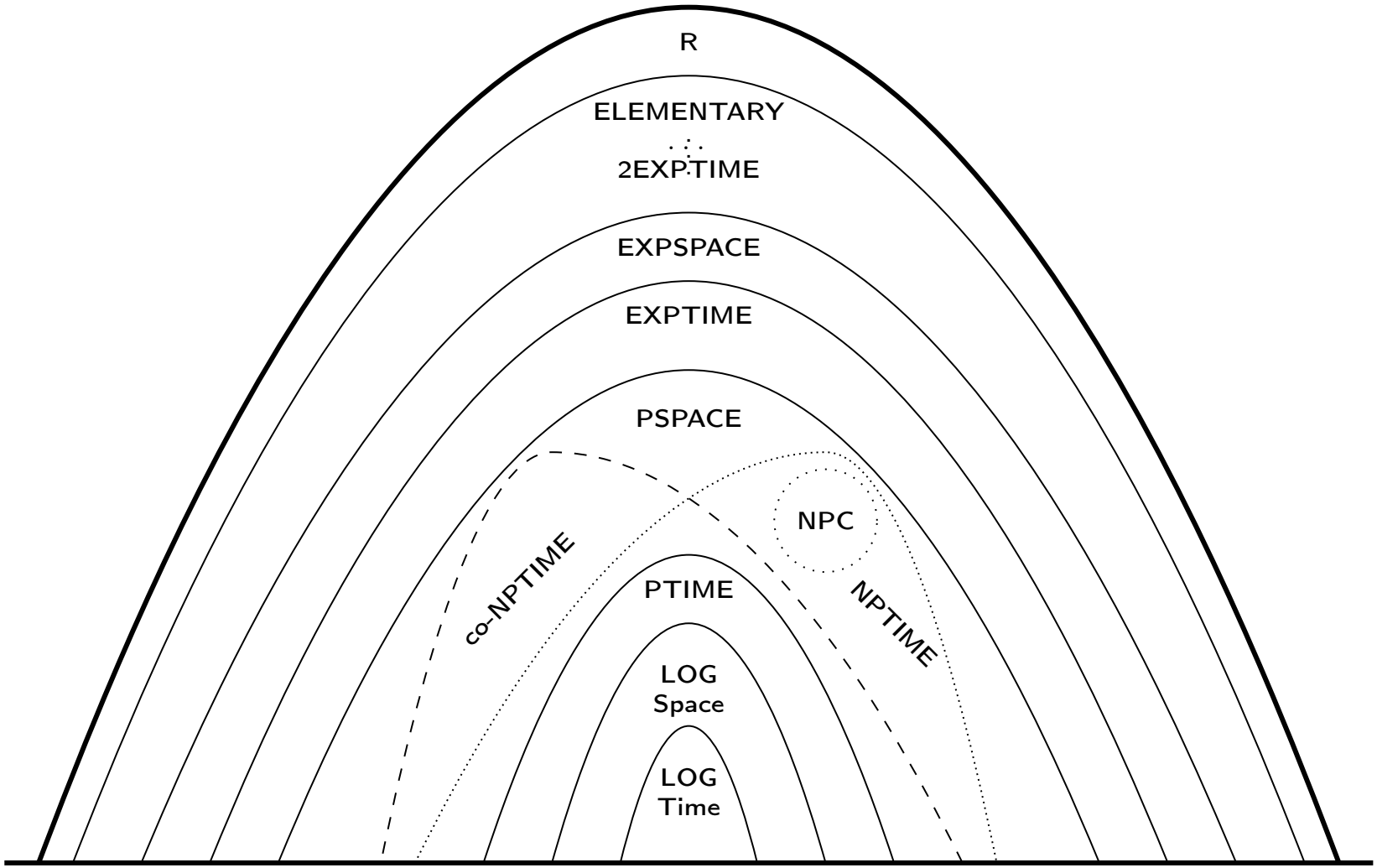
Frédéric Herbreteau
frederic.herbreteau@bordeaux-inp.fr
(d'après les supports de Corentin Travers)

18 février 2025

Réductions polynomiales, la class NPC

Classes de complexité

Bornes inférieures : **difficulté** des problèmes de décision



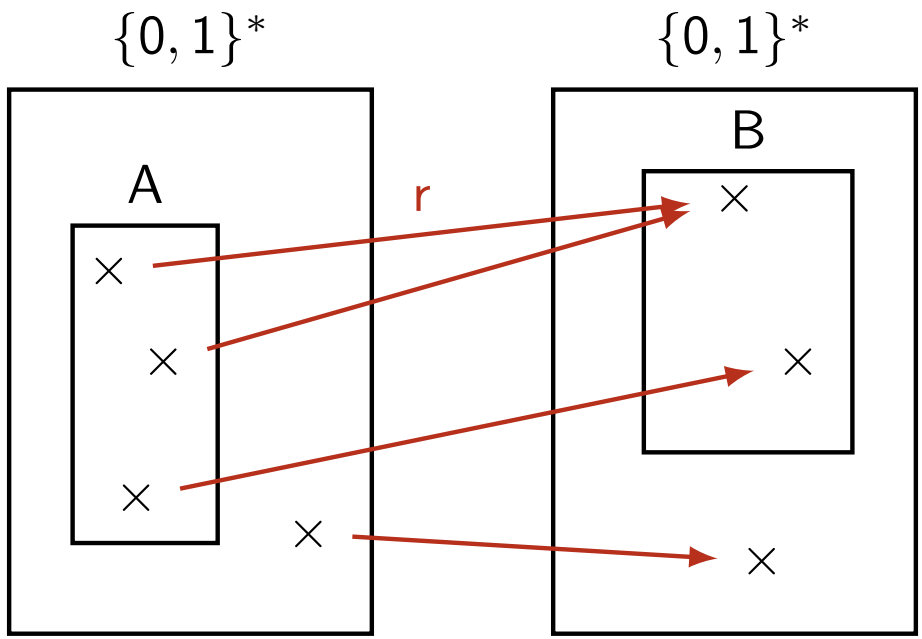
Réduction (many-one)

Soient A et B deux langages sur $\{0,1\}$.

Définition

Une **réduction (many-one)** de A à B est une fonction totale $r : \{0,1\}^* \to \{0,1\}^*$ telle que :

- r est **calculable**
- et pour tout mot $w \in \{0,1\}^*$, $w \in A \iff r(w) \in B$



Réduction polynomiale $A \leq_p B$

Objectif : montrer que deux langages A et B sont dans la **même classe** de complexité

Réduction polynomiale $A \leq_p B$

Objectif : montrer que deux langages A et B sont dans la **même classe** de complexité

L'existence d'une réduction de A à B est **insuffisante**

Exemple : réduction de SUDOKU à $\{1\}$

Réduction polynomiale $A \leq_p B$

Objectif : montrer que deux langages A et B sont dans la **même classe** de complexité

L'existence d'une réduction de A à B est **insuffisante**

Exemple : réduction de SUDOKU à $\{1\}$

Définition

Une **réduction (many-one) polynomiale** de A à B est une fonction totale $r : \{0,1\}^* \to \{0,1\}^*$ telle que :

- r est calculable **en temps polynomial**
- et pour tout mot $w \in \{0,1\}^*$, $w \in A \iff r(w) \in B$

Exemple de réduction polynomiale

SUDOKU

ENTRÉE : une grille G de taille n, partiellement remplie
QUESTION : G peut-elle complétée de façon valide ?

se réduit polynomialement à

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle satisfiable ?

Logique propositionnelle

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

Exemple : $p \implies q \wedge \neg r$

Logique propositionnelle

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

  Exemple : $p \implies q \wedge \neg r$

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

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

Logique propositionnelle

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

  Exemple : $p \implies q \wedge \neg r$

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

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

- Une formule est **vraie** dans v si elle a la valeur 1

  Exemple : $p \implies q \wedge \neg r$ est vraie dans v_1

Logique propositionnelle

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

  Exemple : $p \implies q \wedge \neg r$

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

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

- Une formule est **vraie** dans v si elle a la valeur 1

  Exemple : $p \implies q \wedge \neg r$ est vraie dans v_1

- Une formule est **satisfiable** si **il existe** une valuation v qui la rend **vraie**

  Exemple : $p \implies q \wedge \neg r$ est satisfiable

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G **solution** $\iff$ ϕ **sat.**

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G **solution** $\iff$ ϕ **sat.**

Encodage d'un remplissage de G : variables propositionnelles $p_{i,j,k}$ avec :

$p_{i,j,k}$ représente la proposition "La case (i, j) contient k"

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G solution $\iff$ ϕ sat.

Une et une seule valeur dans chaque case :

$$\phi_1 : \bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left(\left(\bigvee_{1 \leq k \leq n} p_{i,j,k} \right) \wedge \bigwedge_{1 \leq k_1 \neq k_2 \leq n} \neg (p_{i,j,k_1} \wedge p_{i,j,k_2}) \right)$$

NB : $|\phi_1|$ polynomiale en n

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G solution $\iff$ ϕ sat.

Unicité des valeurs dans les lignes :

$$\phi_L : \bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq k \leq n \\ 1 \leq j_1 \neq j_2 \leq n}} \neg \left(p_{i,j_1,k} \wedge p_{i,j_2,k} \right)$$

NB : $|\phi_L|$ polynomiale en n

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G **solution** $\iff$ ϕ **sat.**

Unicité des valeurs dans les colonnes :

$$\phi_C : \bigwedge_{\substack{1 \leq j \leq n \\ 1 \leq k \leq n \\ 1 \leq i_1 \neq i_2 \leq n}} \neg \left(p_{i_1,j,k} \wedge p_{i_2,j,k} \right)$$

NB : $|\phi_C|$ polynomiale en n

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G solution $\iff$ ϕ sat.

Unicité dans les carrées :

$$\phi_S : \bigwedge_{\substack{1 \leq i_1 \leq n \\ 1 \leq j_1 \leq n \\ 1 \leq k \leq n}} \bigwedge_{\substack{(i_2,j_2) \in square(i_1,j_1) \\ (i_2,j_2) \neq (i_1,j_1)}} \neg \left(p_{i_1,j_1,k} \wedge p_{i_2,j_2,k} \right)$$

NB : $|\phi_S|$ polynomiale en n

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$$

G solution $\iff$ ϕ sat.

Remplissage partiel :

$$\phi_I : \; p_{1,1,5} \wedge p_{1,2,3} \wedge p_{1,5,7} \wedge \cdots \wedge p_{9,9,9}$$

NB : $|\phi_I|$ polynomiale en n

$SUDOKU \leq_p SAT$

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

$\phi = \phi_1 \wedge \phi_L \wedge \phi_C \wedge \phi_S \wedge \phi_I$

G solution $\iff$ ϕ sat.

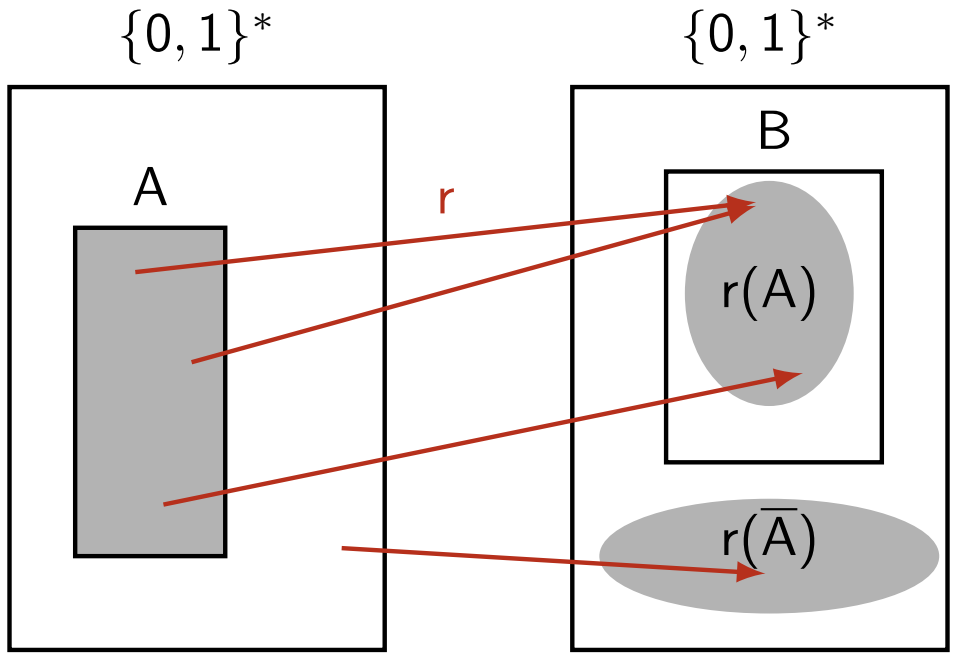
ϕ calculée en temps polynomial en $|G|$

Remplissage partiel :

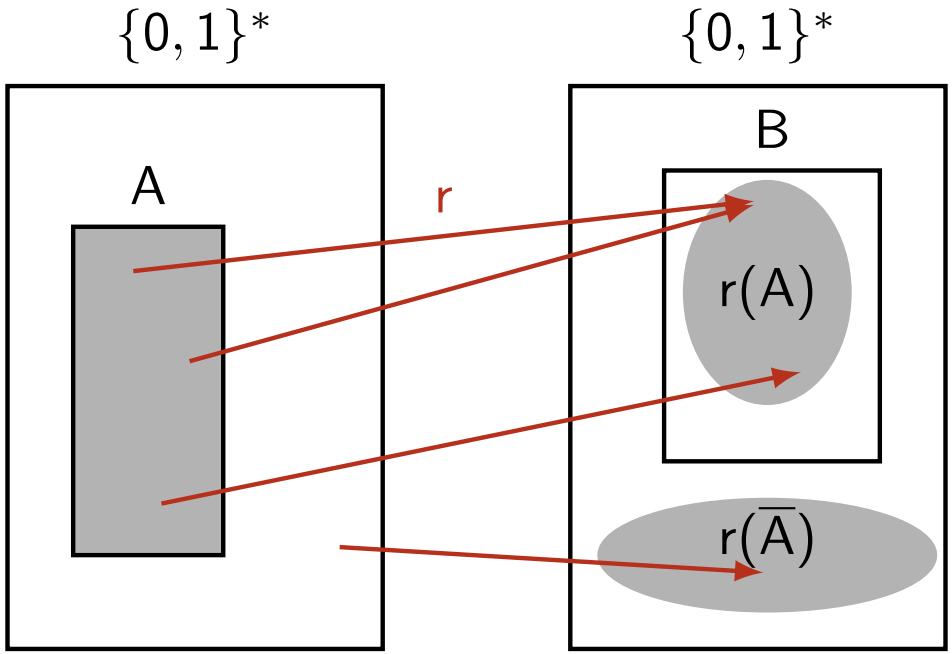
$$\phi_I : \; p_{1,1,5} \wedge p_{1,2,3} \wedge p_{1,5,7} \wedge \cdots \wedge p_{9,9,9}$$

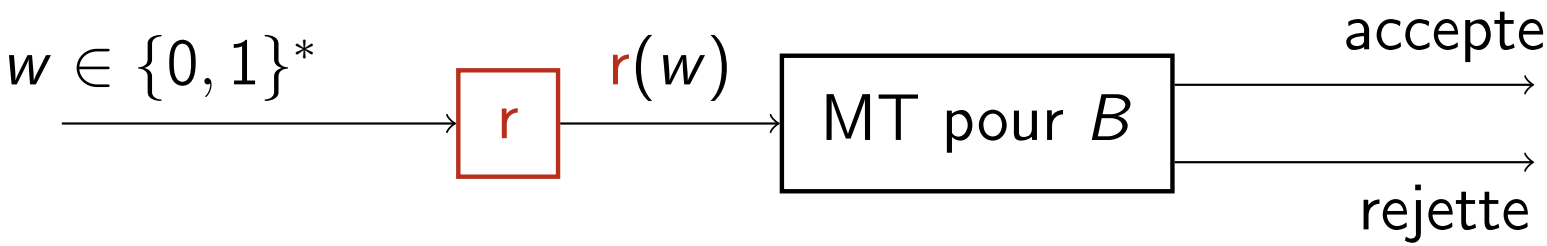
NB : $|\phi_I|$ polynomiale en n

$A \leq_p B$: algorithme polynomial ?

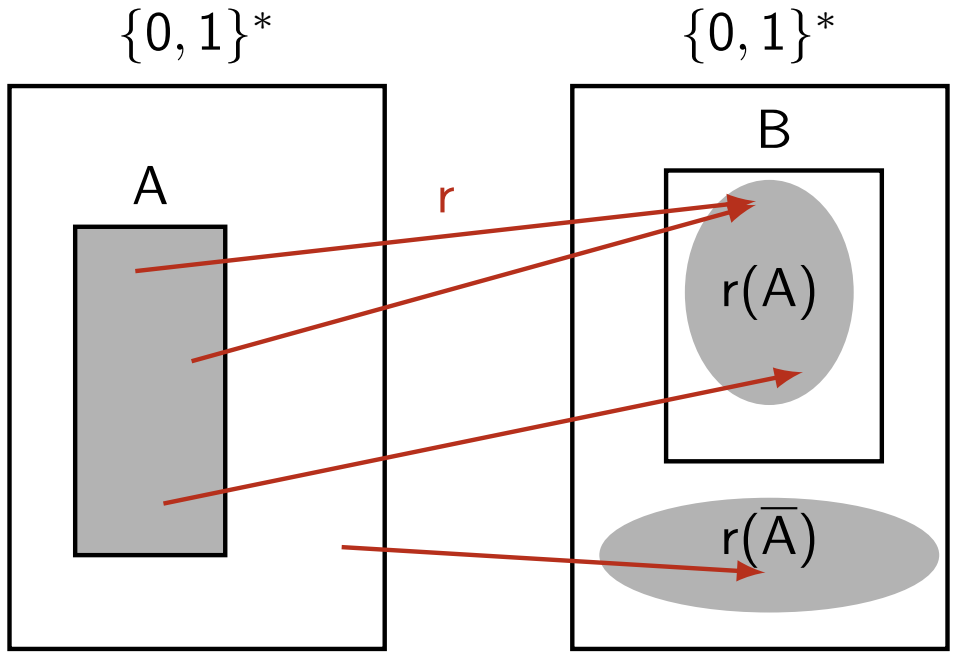


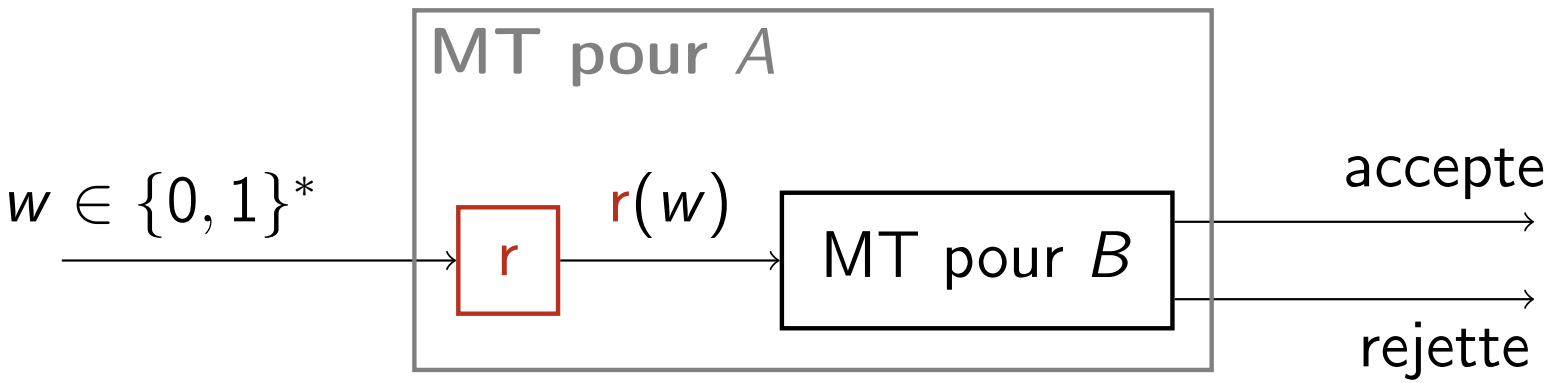
$A \leq_p B$: algorithme polynomial ?





$A \leq_p B$: algorithme polynomial ?

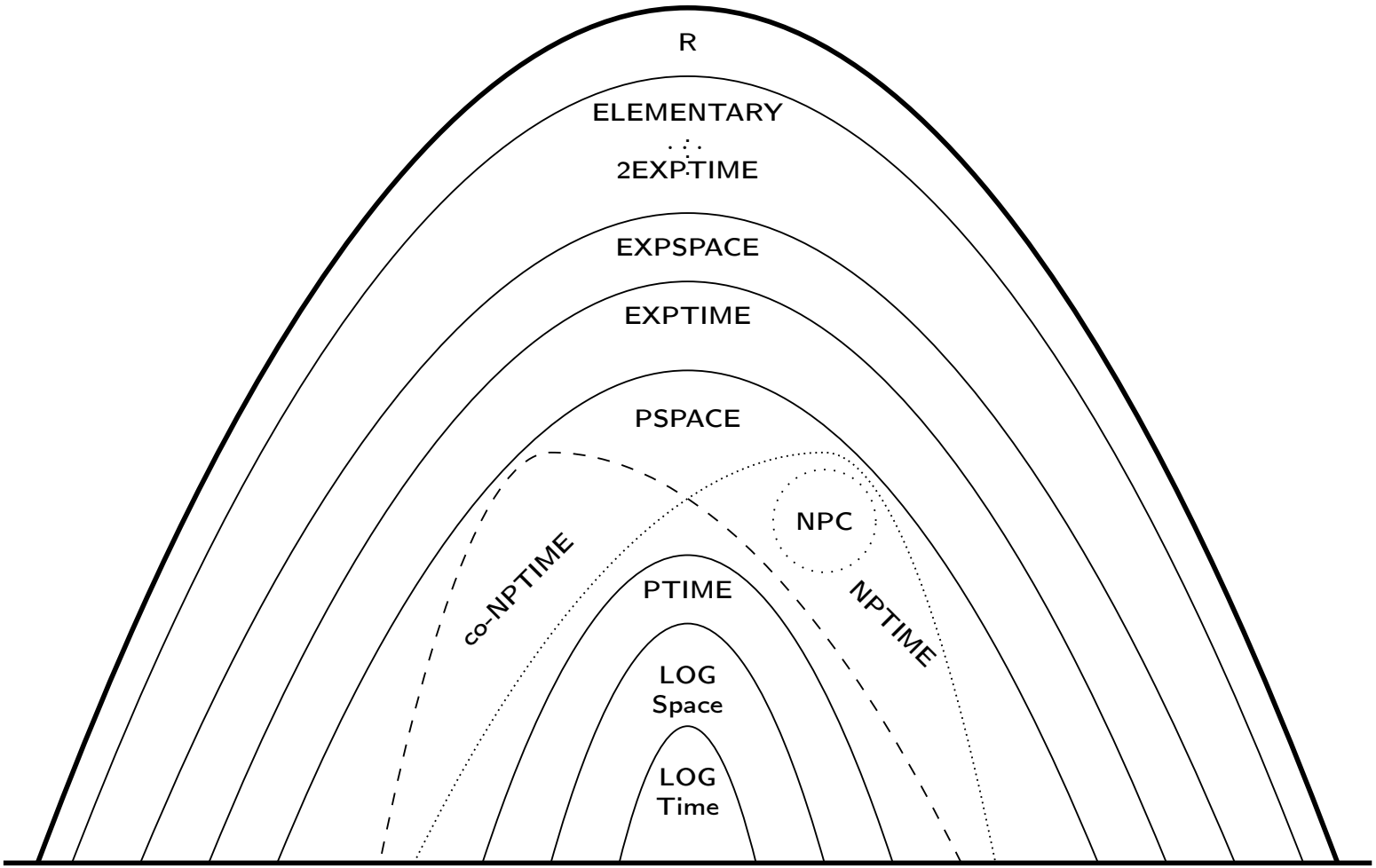




NP-difficile, NP-complet

Problématique : existe-t-il des problèmes **plus difficiles** que d'autres ? Comment le démontrer ?

→ **borne inférieure** de complexité



NP-difficile, NP-complet

Problématique : existe-t-il des problèmes **plus difficiles** que d'autres ? Comment le démontrer ?

$\rightarrow$ **borne inférieure** de complexité

Définition

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

- A est **NP-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in \mathrm{NP}$

NP-difficile, NP-complet

Problématique : existe-t-il des problèmes **plus difficiles** que d'autres ? Comment le démontrer ?

$\rightarrow$ **borne inférieure** de complexité

Définition

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

- A est **NP-difficile** ssi $B \leq_p A$ pour **tout** langage $B \in \text{NP}$
- A est **NP-complet** ssi A est NP-difficile **et** $A \in \text{NP}$

Notions généralisables à **d'autres classes** de complexité

$A \leq_p B$: quelques propriétés

Soient $A, B, C \subseteq \{0,1\}^*$ trois langages

1. Si $B \in \mathrm{P}$ et $A \leq_p B$ alors $A \in \mathrm{P}$

$A \leq_p B$: quelques propriétés

Soient $A, B, C \subseteq \{0,1\}^*$ trois langages

1. Si $B \in \mathrm{P}$ et $A \leq_p B$ alors $A \in \mathrm{P}$

2. Transitivité : si $A \leq_p B$ et $B \leq_p C$ alors $A \leq_p C$

$A \leq_p B$: quelques propriétés

Soient $A, B, C \subseteq \{0,1\}^*$ trois langages

1. Si $B \in \mathrm{P}$ et $A \leq_p B$ alors $A \in \mathrm{P}$

2. Transitivité : si $A \leq_p B$ et $B \leq_p C$ alors $A \leq_p C$

3. Si A est **NP-difficile** alors $A \in \mathrm{P} \implies \mathrm{P} = \mathrm{NP}$

$A \leq_p B$: quelques propriétés

Soient $A, B, C \subseteq \{0,1\}^*$ trois langages

1. Si $B \in \mathrm{P}$ et $A \leq_p B$ alors $A \in \mathrm{P}$
2. Transitivité : si $A \leq_p B$ et $B \leq_p C$ alors $A \leq_p C$
3. Si A est **NP-difficile** alors $A \in \mathrm{P} \implies \mathrm{P} = \mathrm{NP}$
4. Si A est **NP-complet** alors $\mathrm{A} \in P \iff \mathrm{P} = \mathrm{NP}$

Langages NP-difficiles/complets

- Question : existe-t-il des langages de NP qui ne sont pas NP-complets ?

Langages NP-difficiles/complets

- Question : existe-t-il des langages de NP qui ne sont pas NP-complets ?

  $\rightarrow$ **Oui**, tous les langages de P, mais également :

  PRIME FACTORISATION

  ENTRÉE : deux entiers $n, k \in \mathbb{N}$
  QUESTION : n admet-il un facteur premier $< k$?

  dans NP $\cap$ co-NP, mais P ? NP-complet ?

Langages NP-difficiles/complets

- Question : existe-t-il des langages de NP qui ne sont pas NP-complets ?

  $\rightarrow$ **Oui**, tous les langages de P, mais également :

  PRIME FACTORISATION

  ENTRÉE : deux entiers $n, k \in \mathbb{N}$
  QUESTION : n admet-il un facteur premier $< k$?

  dans NP $\cap$ co-NP, mais P ? NP-complet ?

- Question : existe-t-il un langage NP-complet ?

Théorème de Cook-Levin

Logique propositionnelle

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

Exemple : $p \implies q \wedge \neg r$

Logique propositionnelle

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

  Exemple : $p \implies q \wedge \neg r$

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

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

Logique propositionnelle

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

  Exemple : $p \implies q \wedge \neg r$

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

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

- Une formule est **vraie** dans v si elle a la valeur 1

  Exemple : $p \implies q \wedge \neg r$ est vraie dans v_1

Satisfiabilité

Une formule est **satisfiable** si **il existe** une valuation v qui la rend **vraie**

Exemple : $p \implies q \wedge \neg r$ est satisfiable

Satisfiabilité

Une formule est **satisfiable** si **il existe** une valuation v qui la rend **vraie**

Exemple : $p \implies q \wedge \neg r$ est satisfiable

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle satisfiable ?

Satisfiabilité

Une formule est **satisfiable** si **il existe** une valuation v qui la rend **vraie**

Exemple : $p \implies q \wedge \neg r$ est satisfiable

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle satisfiable ?

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

SAT est NP-complet

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

Preuve

- Appartient : SAT est **dans NP**

SAT est NP-complet

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

Preuve

- Appartient : SAT est **dans NP**
- Difficile : SAT est **NP-difficile**

SAT est NP-complet

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

Preuve

- Appartient : SAT est **dans NP**
- Difficile : SAT est **NP-difficile**
  - Mq : pour tout langage $L \in NP$, on a $L \leq_p SAT$

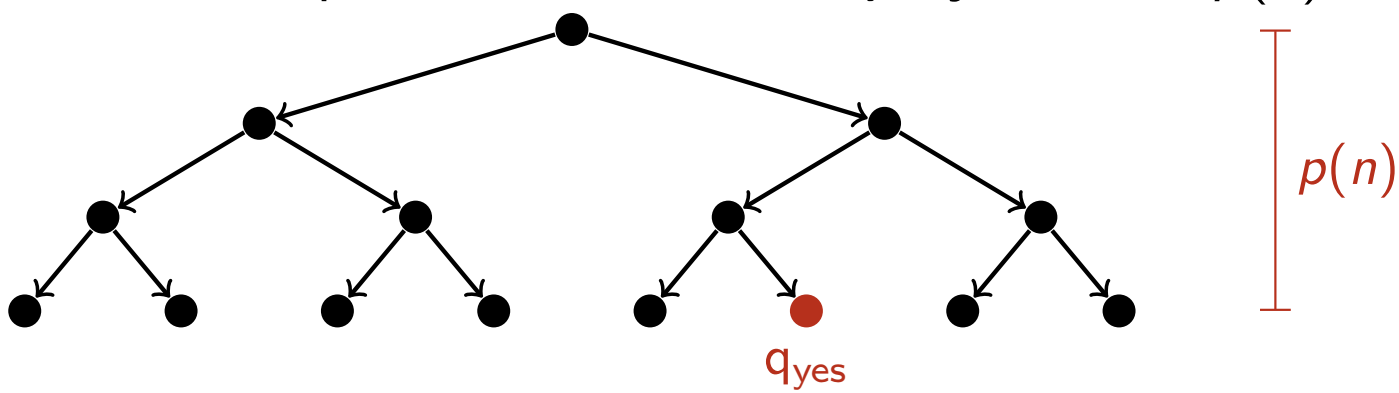
SAT est NP-complet

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

Preuve

- Appartient : SAT est **dans NP**
- Difficile : SAT est **NP-difficile**
  - Mq : pour tout langage $L \in NP$, on a $L \leq_p SAT$
  - L est décidé par une MT non-dét. **polynomiale** $p(n)$



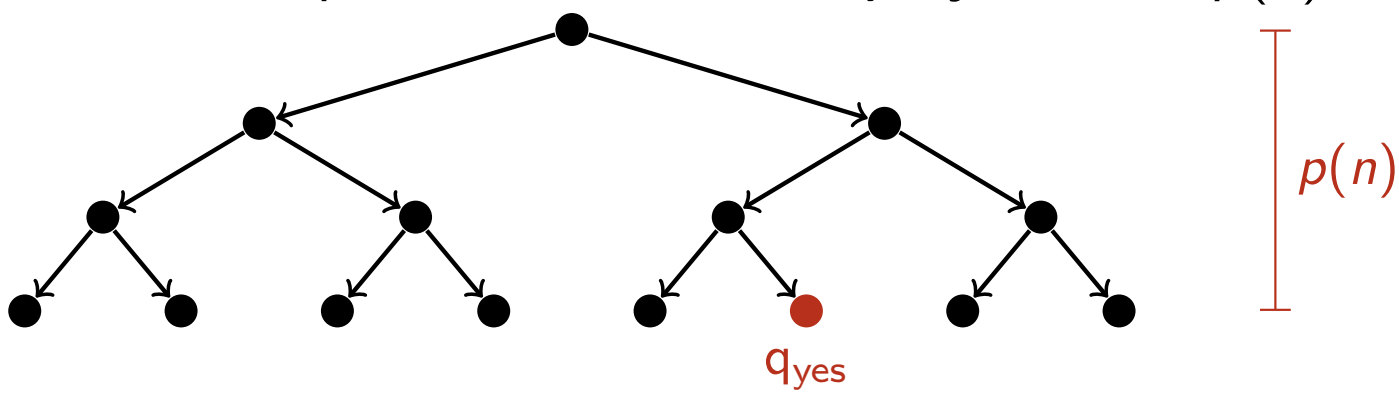
SAT est NP-complet

Théorème (Cook 1971, Levin 1973)

Le problème SAT est NP-complet

Preuve

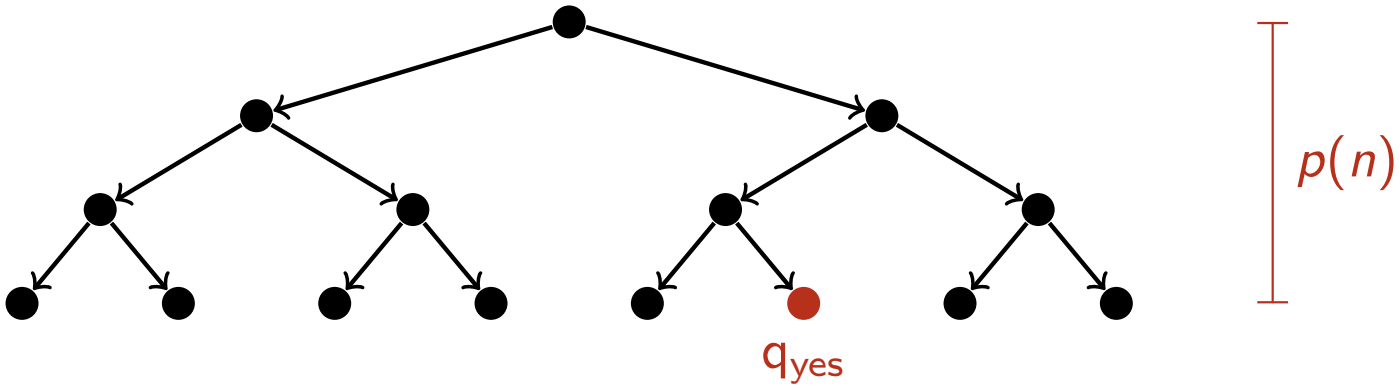
- Appartient : SAT est **dans NP**
- Difficile : SAT est **NP-difficile**
  - Mq : pour tout langage $L \in NP$, on a $L \leq_p SAT$
  - L est décidé par une MT non-dét. **polynomiale** $p(n)$

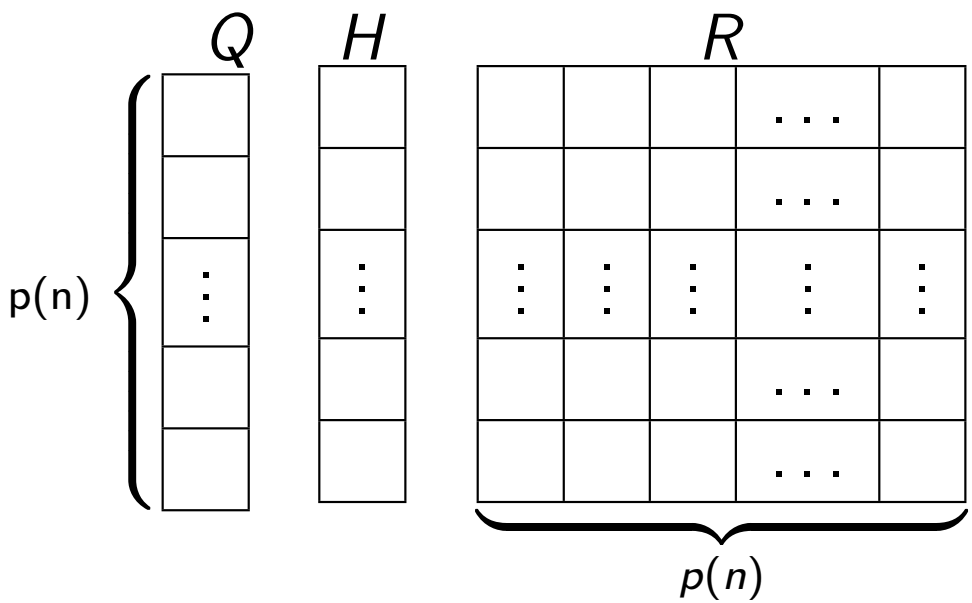


  - formule propositionnelle ϕ de **taille** $\mathcal{O}(p^c(n))$ **satisfiable**
    $\iff$ M accepte
    $\iff$ M fait $p(n)$ **choix non-dét. puis accepte**

Tableau

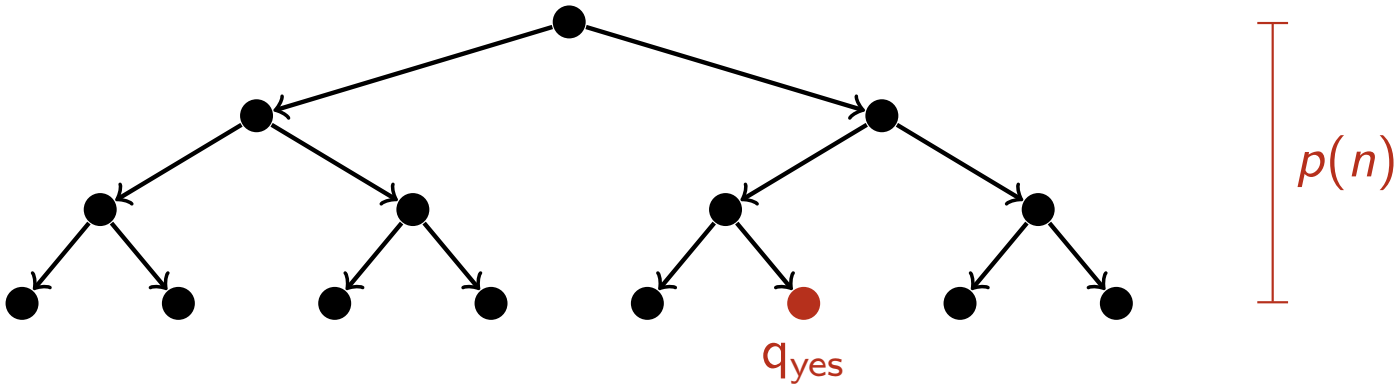
Séquence de choix non-déterministes valides

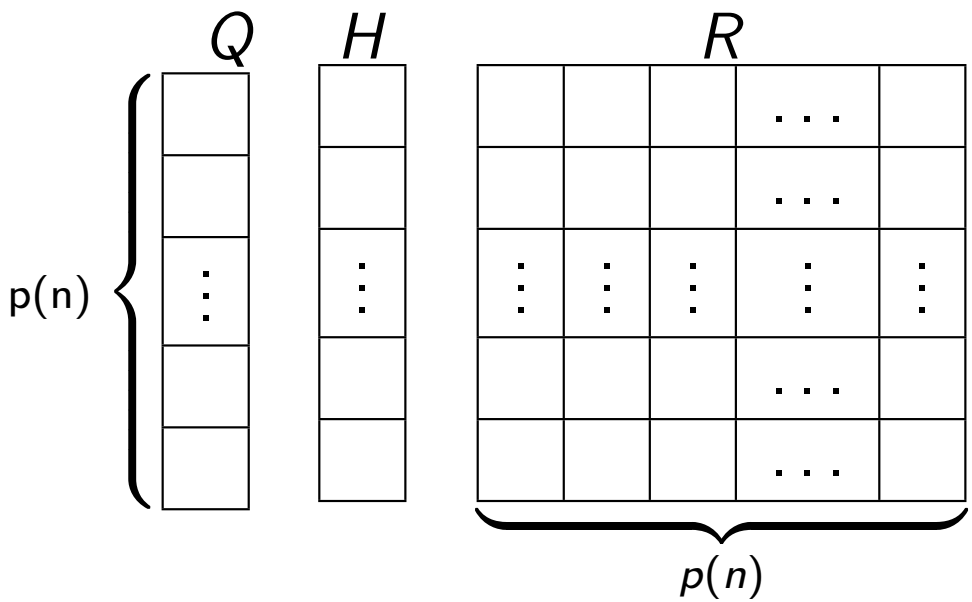




Tableau

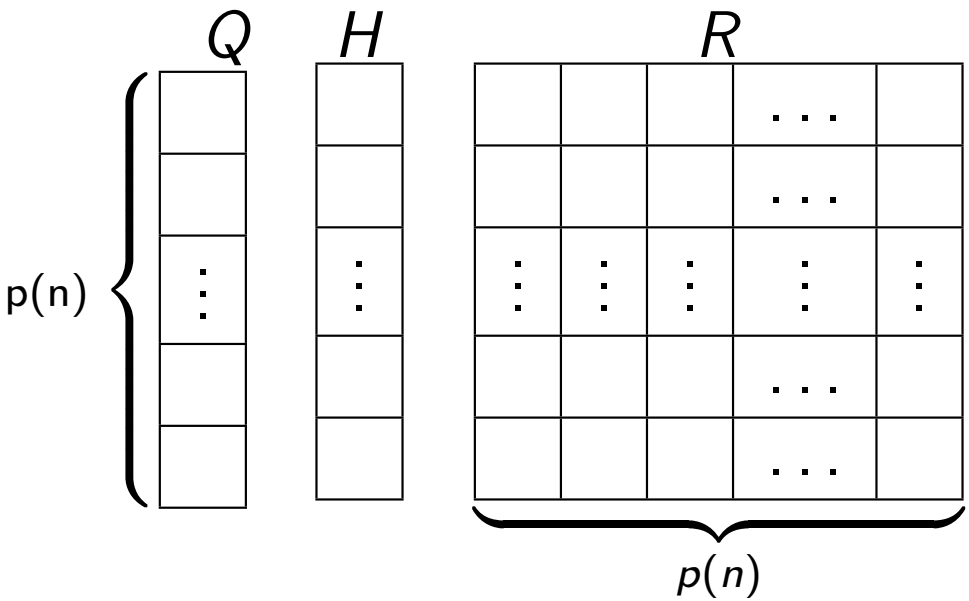
Séquence de choix non-déterministes valides





Objectif : encoder le tableau en **logique propositionnelle**

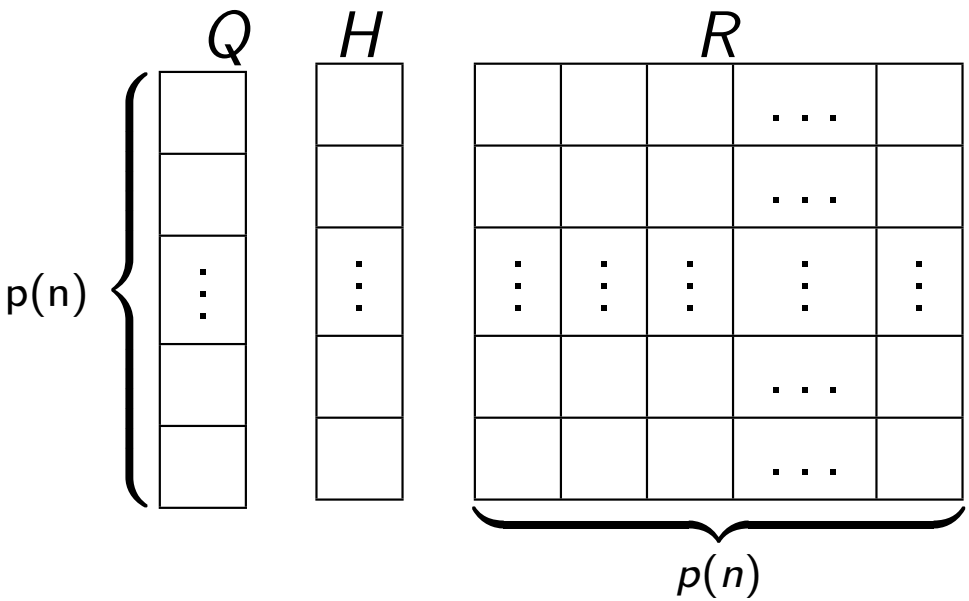
Encodage du tableau : variables



Variables propositionnelles : avec $1 \leq i \leq p(n)$

- $q_{i,k}$ avec $k \in Q$
- $h_{i,l}$ avec $1 \leq l \leq p(n)$
- $r_{i,m,s}$ avec $1 \leq m \leq p(n)$ et $s \in \Sigma$

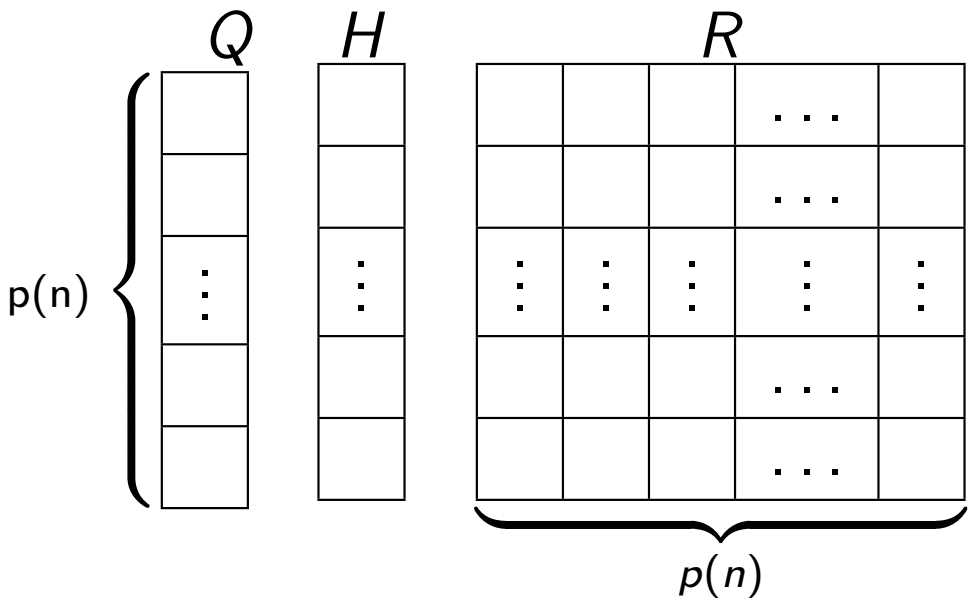
Encodage du tableau : formules



$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

ϕ **sat.** $\Leftrightarrow$ M **accepte** w

Encodage du tableau : formules



$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

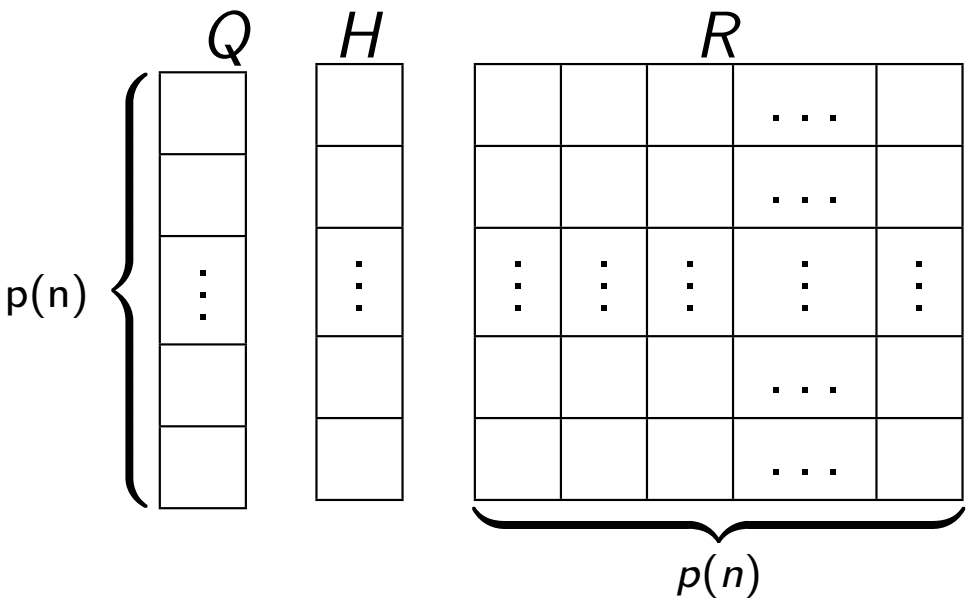
ϕ **sat.** $\Leftrightarrow$ M **accepte** w

ϕ_1 : chaque case contient **1 et 1 seule valeur**

$$\bigwedge_{\substack{1 \leq i \leq p(n) \\ 1 \leq m \leq p(n)}} \left[\left(\bigvee_{s \in \Sigma} r_{i,m,s} \right) \wedge \bigwedge_{s_1 \neq s_2 \in \Sigma} \neg (r_{i,m,s_1} \wedge r_{i,m,s_2}) \right]$$

de même pour les autres variables : $q_{i,k}$ et $h_{i,l}$

Encodage du tableau : formules



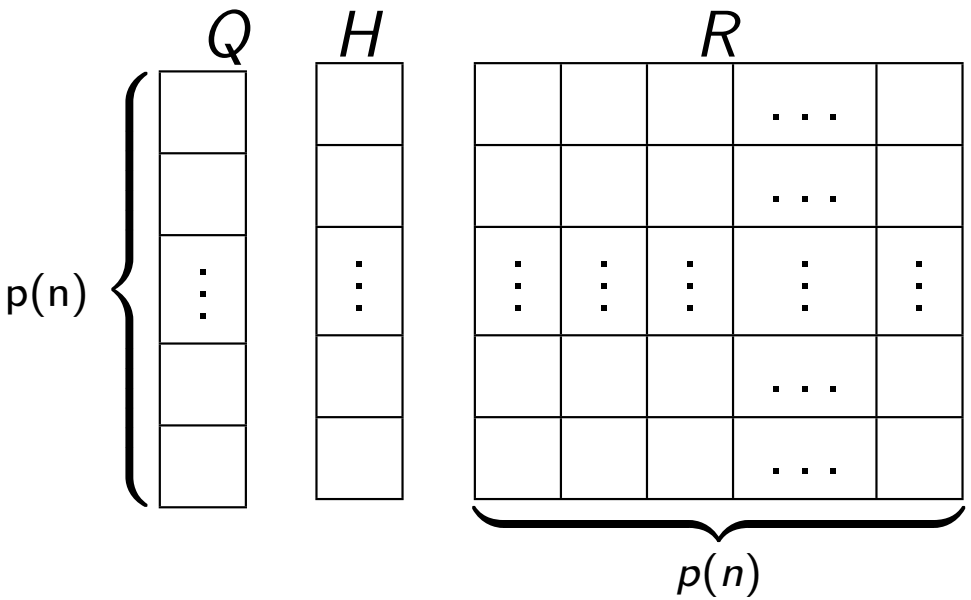
$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

ϕ **sat.** $\Leftrightarrow$ M **accepte** w

ϕ_I : 1ère ligne = configuration initiale (q_0, w)

$$q_{1,q_0} \wedge h_{1,1} \wedge \left[\bigwedge_{1 \leq m \leq |w|} r_{1,m,w_m} \wedge \bigwedge_{|w| < m \leq p(n)} r_{1,m,\square} \right]$$

Encodage du tableau : formules



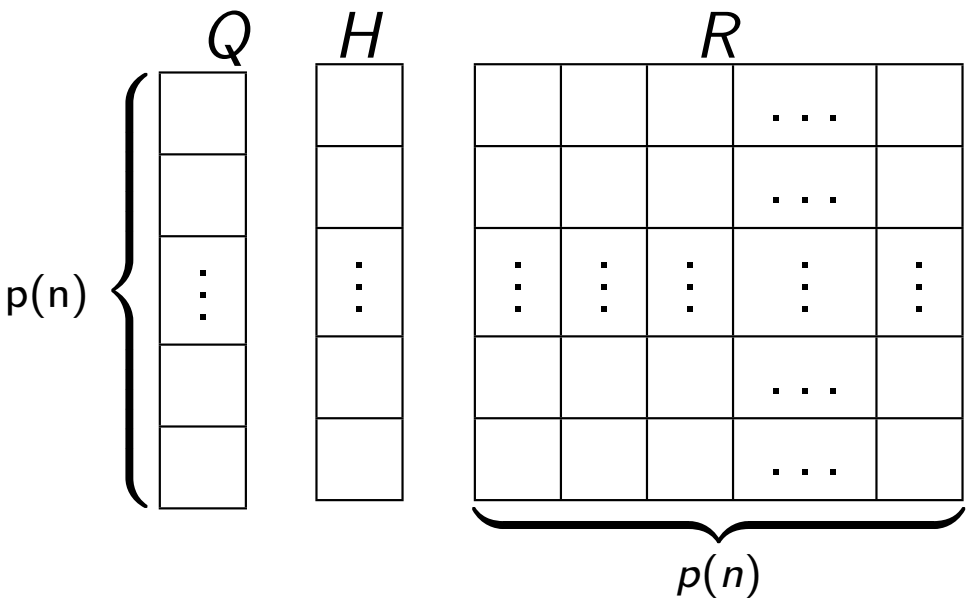
$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

ϕ **sat.** $\Leftrightarrow$ M **accepte** w

ϕ_F : la dernière configuration est acceptante

$$\bigvee_{q_{yes} \in F} q_{p(n),q_{yes}}$$

Encodage du tableau : formules



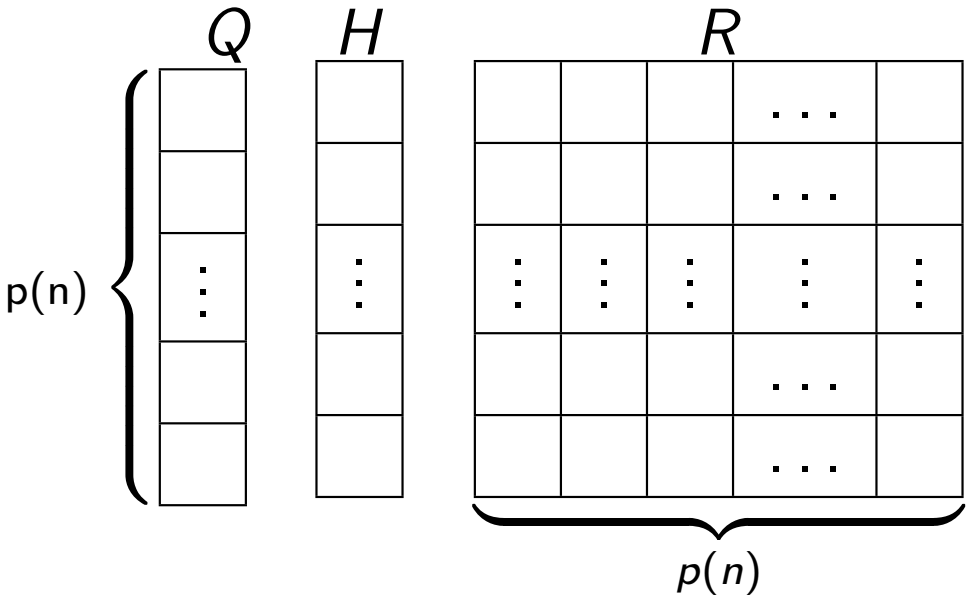
$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

ϕ **sat.** $\Leftrightarrow$ M **accepte** w

ϕ_T : respect des transitions de T

$$\bigwedge_{\substack{1 \leq i < p(n) \\ 1 \leq m \leq p(n) \\ s \in \Sigma}} (r_{i,m,s} \wedge \neg h_{i,m} \implies r_{i+1,m,s}) \quad \wedge \quad \bigwedge_{1 \leq i < p(n)} \bigvee_{t \in T} \phi_{t,i}$$

Encodage du tableau : formules



$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

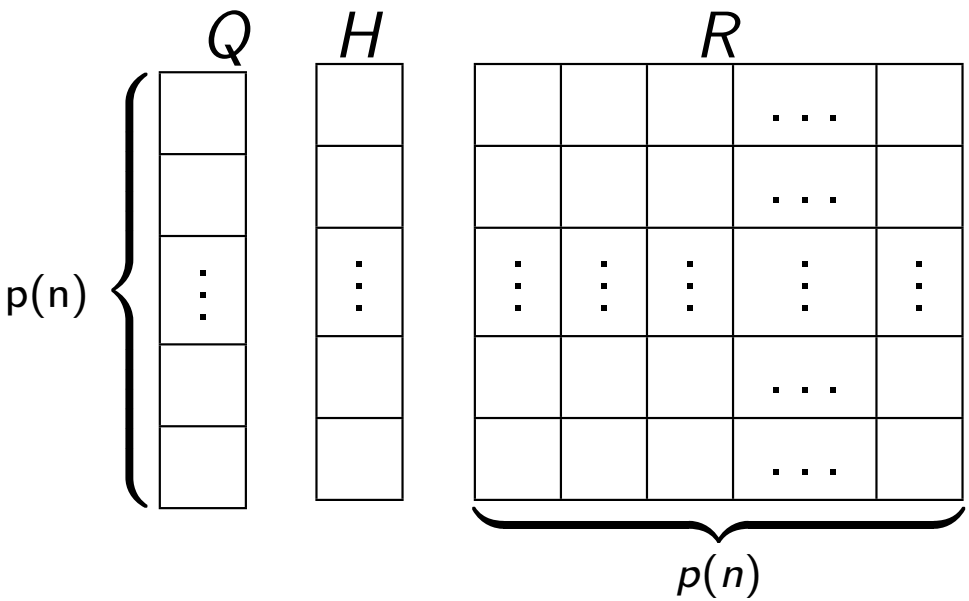
ϕ **sat.** $\Leftrightarrow$ M **accepte** w

$\phi_{t,i}$: encodage de la transition $q_1, \alpha \to q_2, \beta, \delta$ pour la ligne i

$$q_{i,q_1} \wedge q_{i+1,q_2} \wedge \bigwedge_{1 \leq m \leq p(n)} (r_{i,m,\alpha} \wedge h_{i,m} \implies r_{i+1,m,\beta} \wedge h_{i+1,m+\delta})$$

avec $\delta \in \{-1, 0, 1\}$

Encodage du tableau : formules

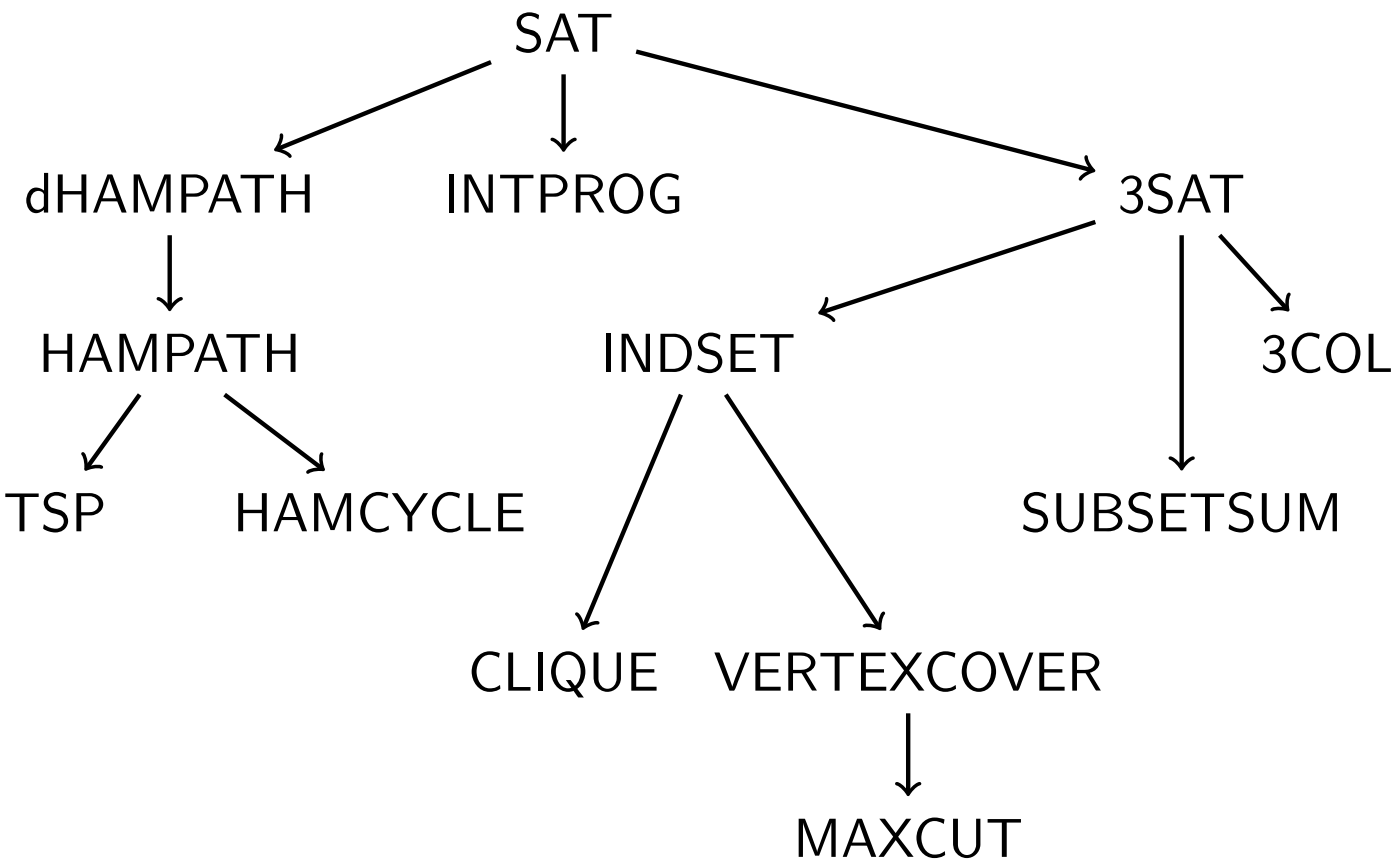


$\phi = \phi_1 \wedge \phi_I \wedge \phi_F \wedge \phi_T$

ϕ **sat.** $\Leftrightarrow$ M **accepte** w

- ϕ de taille $\mathcal{O}(|\langle M \rangle|.p^c(n))$ pour une constante $c \in \mathbb{N}$
- **Calculée en temps polynomial**

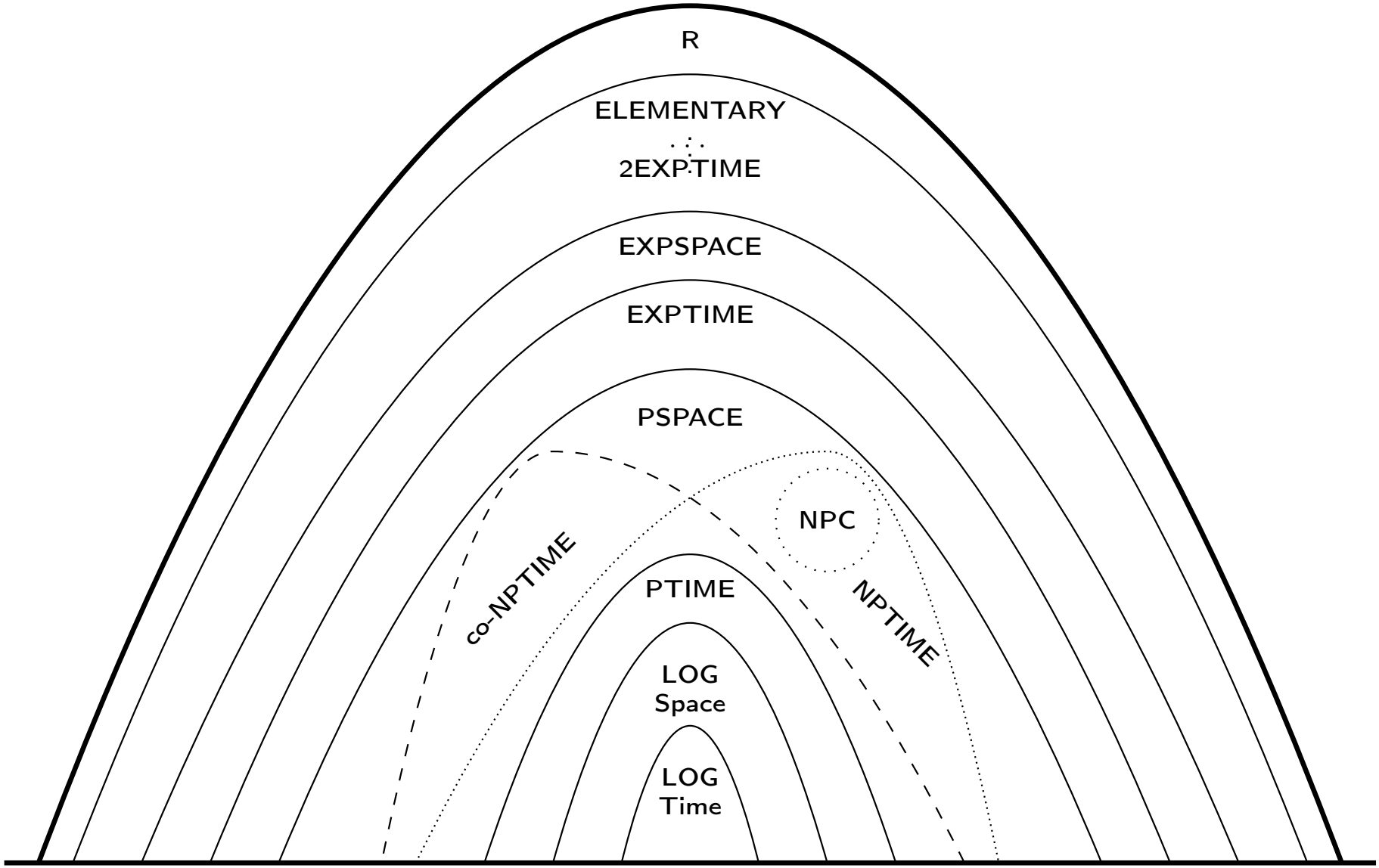
Ex. langages **NP**-complets (Karp)



cf. TD + `https://www.csc.kth.se/~viggo/wwwcompendium/`

La classe coNP

Classes de complexité



coNP

Soit $L \subseteq \{0,1\}^*$.

$\overline{L} = \{0,1\}^* \setminus L$ est le **complémentaire** de L.

coNP

Soit $L \subseteq \{0,1\}^*$.

$\overline{L} = \{0,1\}^* \setminus L$ est le **complémentaire** de L.

Exemple : $\mathit{UNSAT} = \{\langle\phi\rangle \mid \phi$ n'est pas satisfiable$\}$ est le complémentaire de SAT

coNP

Soit $L \subseteq \{0,1\}^*$.

$\overline{L} = \{0,1\}^* \setminus L$ est le **complémentaire** de L.

Exemple : $UNSAT = \{\langle\phi\rangle \mid \phi$ n'est pas satisfiable$\}$ est le complémentaire de SAT

Définition

coNP $= \{L \subseteq \{0,1\}^* \mid \overline{L} \in$ NP$\}$

coNP

Soit $L \subseteq \{0,1\}^*$.

$\overline{L} = \{0,1\}^* \setminus L$ est le **complémentaire** de L.

Exemple : $UNSAT = \{\langle\phi\rangle \mid \phi$ n'est pas satisfiable$\}$ est le complémentaire de SAT

Définition

coNP $= \{L \subseteq \{0,1\}^* \mid \overline{L} \in$ NP$\}$

Remarque : **coNP n'est pas** le complémentaire de **NP**

P, NP et coNP

Lemme

$P \subseteq NP \cap coNP$

Preuve. Exercice

P, NP et coNP

Lemme
$P \subseteq NP \cap coNP$

Preuve. Exercice

Lemme
Si $P = NP$, *alors* $NP = coNP$

Preuve
P = coP (exercice) et P = NP (hypothèse).

SAT, NP et coNP

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **satisfiable** ?

UNSAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **insatisfiable** ?

SAT, NP et coNP

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **satisfiable** ?

UNSAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **insatisfiable** ?

- UNSAT est coNP, car SAT est NP

SAT, NP et coNP

SAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **satisfiable** ?

UNSAT

ENTRÉE : une formule propositionnelle ϕ

QUESTION : ϕ est-elle **insatisfiable** ?

- UNSAT est coNP, car SAT est NP
- SAT coNP ? Que serait un certificat pour UNSAT ?

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

- PRIME FACTORISATION est dans NP

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

- PRIME FACTORISATION est dans NP
  certificat= p, vérification en **temps polynomial**
  $\rightarrow$ p premier (AKS 2002) et $p < k$ et p divise n

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

- PRIME FACTORISATION est dans NP
  certificat= p, vérification en **temps polynomial**
  $\rightarrow$ p premier (AKS 2002) et $p < k$ et p divise n
- PRIME FACTORISATION est dans coNP

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

- PRIME FACTORISATION est dans NP
  certificat= p, vérification en **temps polynomial**
  $\rightarrow$ p premier (AKS 2002) et $p < k$ et p divise n
- PRIME FACTORISATION est dans coNP
  certificat= factorisation de n : $f_1, \ldots, f_m$, vérification en **temps polynomial**
  $\rightarrow$ f_i est premier (AKS 2002) et $\prod f_i = n$ et $f_i \geq k$

FACTOR, NP et coNP

PRIME FACTORISATION

ENTRÉE : deux entiers $n, k \in \mathbb{N}$
QUESTION : n admet-il un facteur premier $< k$?

Ex : $(6, 3)$ ✓, mais $(49, 4)$ $\times$

- PRIME FACTORISATION est dans NP
  certificat= p, vérification en **temps polynomial**
  $\rightarrow$ p premier (AKS 2002) et $p < k$ et p divise n
- PRIME FACTORISATION est dans coNP
  certificat= factorisation de n : $f_1, \ldots, f_m$, vérification en **temps polynomial**
  $\rightarrow$ f_i est premier (AKS 2002) et $\prod f_i = n$ et $f_i \geq k$
- Cryptographie basée sur PRIME FACTORISATION (RSA)

Classes de complexité