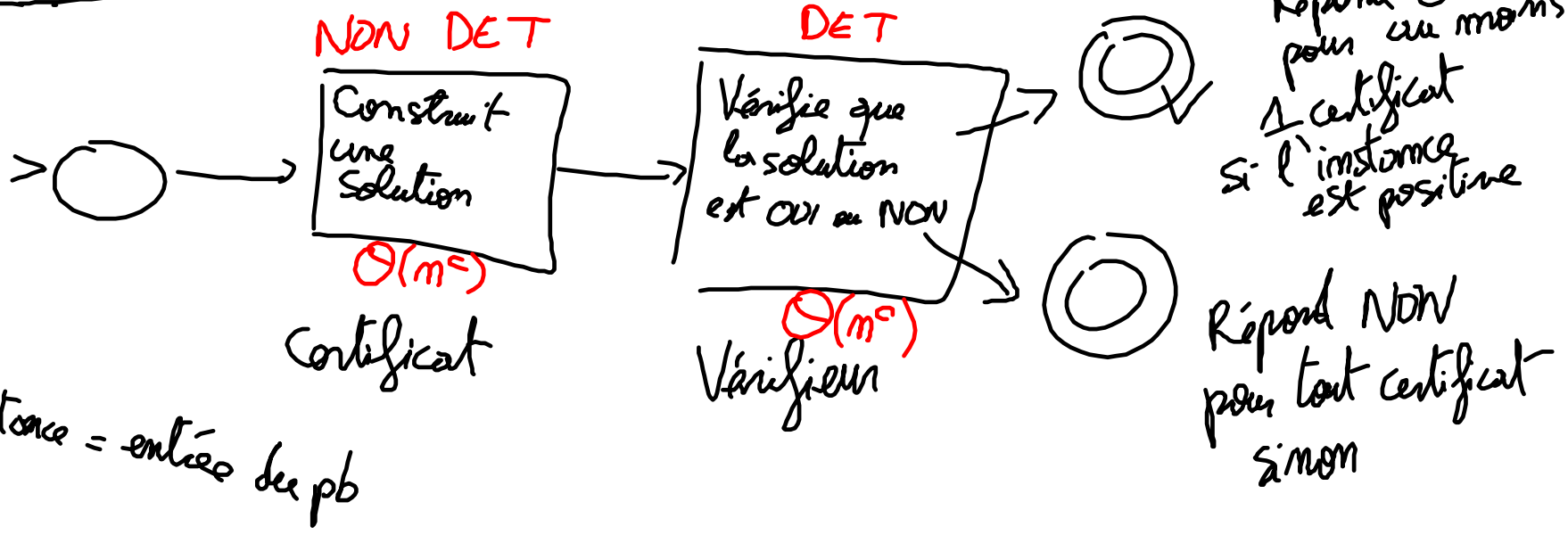
Pb un pb $\in NP$

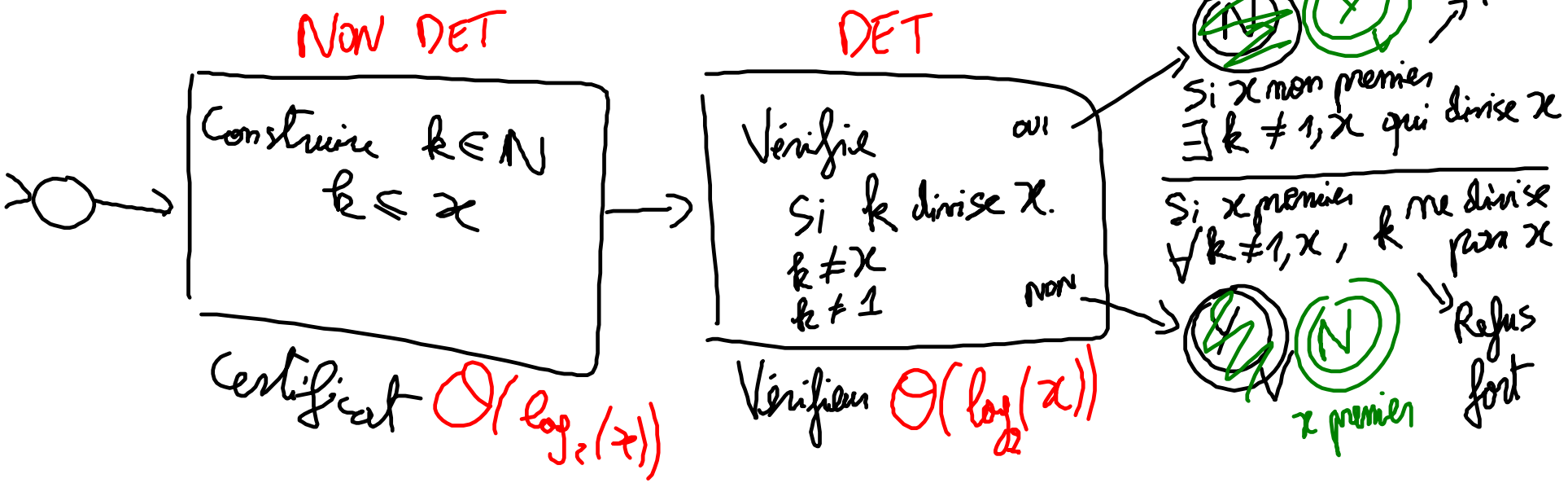


Instance = entrée du pb

NONPRIM $\in$ NP $\Longleftrightarrow$ PRIM $\in$ Co-NP

entrée : $x \in \mathbb{N}$

	x	



PRIM $\in$ P ? → on ne sait pas

L'algo suivant résout PRIM

$\forall\ 2 \leq k \leq \sqrt{x}$

Si k divise x

alors RETURN NON

Sinon

RETURN OUI

Complexité

$\mathcal{O}(\sqrt{x}\ \log_2(x))$

⚠ Taille de l'entrée $\neq x$

$= \log_2(x)$

$\mathcal{O}(\sqrt{2^{\log_2(x)}}\ \log_2(x))$

↳ exponentiel

Chapitre 4 : Réductions et complétude

ENSIIE - Théorie de la complexité

Dimitri Watel (dimitri.watel@ensiie.fr)

2017

Réduction polynomiale et complétude

Idée de base

Une manière de résoudre un problème Π_1 est de le transformer en un autre Π_2 qu'on sait résoudre.

Si la transformation est rapide et que le problème Π_2 peut être résolu rapidement, alors Π_1 peut être résolu rapidement.

Si la transformation est rapide et que le problème Π_1 ne peut être résolu rapidement, alors Π_2 ne peut être résolu rapidement.

Π_1 = Chemin Ham : Soit $G=(V,A)$ orienté, $\exists$? un chemin qui passe par tous les noeuds

Π_2 = Plus long chemin : Soit $G=(V,A)$ orienté, et $k \in \mathbb{N}$ ~~trouver le plus long~~ $\exists$! un chemin dans G de taille $\geq k$?

Π_1 équivalent à Π_2 avec $k = n = |V|$

On peut transformer une instance de Π_1 $G=(V,A)$ en instance de $\Pi_2 = (G=(V,A), n)$ tq résoudre Π_2 permet de résoudre Π_1.

S'il existe un algo efficace pour résoudre Π_2 alors on a un algo efficace pour Π_1

L'inverse n'est pas vrai

$\Rightarrow$ Π_1 est "plus facile" que Π_2

Réduction polynomiale de Karp

Définition (informelle)

Soit deux problèmes de décision Π_1 et Π_2, une réduction polynomiale de Π_1 vers Π_2 transforme en temps polynomial toute instance positive (resp. negative) de Π_1 en une instance positive (resp. negative) de Π_2.

Si Π_1 se réduit à Π_2, alors on dit que Π_2 est *plus difficile* que Π_1.

La transformation du slide précédent est une réduction poly de Karp.

- $\exists$ chemin hamiltonien $\Leftrightarrow$ $\exists$ chemin de taille $\geq n$ (OUI → OUI, NON → NON)
- Durée de la transformation : quasiment instantané / $O(\log_2(n))$ en réalité /.

Réduction polynomiale de Karp

Définition

Soit deux problèmes de décision Π_1 et Π_2, une réduction polynomiale de $\Pi_1 = (\mathcal{L}^1, \mathcal{L}^1_Y, \mathcal{L}^1_N)$ vers $\Pi_2 = (\mathcal{L}^2, \mathcal{L}^2_Y, \mathcal{L}^2_N)$ est un algorithme $\mathcal{R}$ tel que

- $\mathcal{R}$ a une complexité polynomiale en $|I|$
- $\mathcal{R}$ prend en entrée une instance $\mathcal{I}$ de Π_1 et renvoie en sortie une instance $\mathcal{J}$ de Π_2
- $\mathcal{I} \in \mathcal{L}^1_Y \Leftrightarrow \mathcal{J} \in \mathcal{L}^2_Y$

On note $\Pi_1 \preccurlyeq \Pi_2$. $\leftrightarrow \Pi_1$ "plus facile" que Π_2

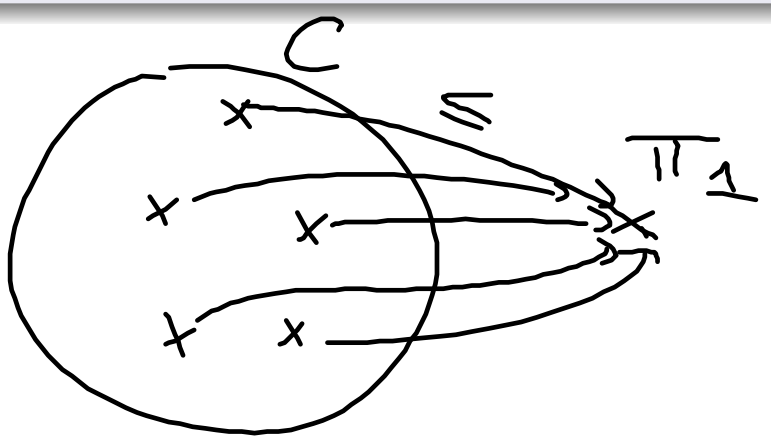
Chemin Hamiltonien / Plus long chemin : $I = (G=(V,A))$ $|I| = n^2$ (écrire G avec sa matrice d'adjacence)

$R(I) = J = (G=(V,A), n)$ R = Compter $|V| = n$ et renvoyer (G, n) $\Theta(n) = \Theta(\sqrt{|I|})$

G Hamiltonien $\Leftrightarrow \exists$ chemin de taille n dans G : $I \in \mathcal{L}^1_Y \Leftrightarrow J \in \mathcal{L}^2_Y$ POLY

Problème $\mathcal{C}$-difficile

Soit $\mathcal{C}$ une classe de complexité (NP, EXPTIME, ...) et Π_1 un problème de décision. On dit que Π_1 est $\mathcal{C}$-Difficile si, pour tout problème Π_2 de $\mathcal{C}$, $\Pi_2 \preccurlyeq \Pi_1$.

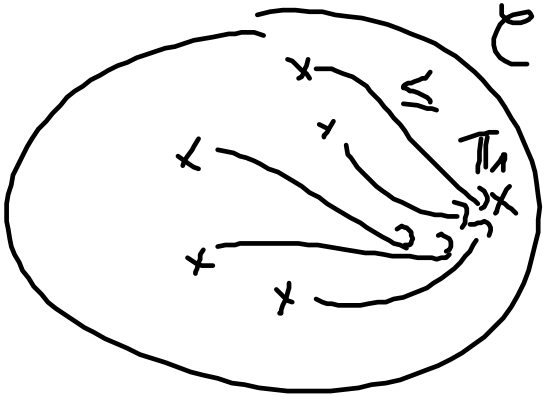


Π_1 est "plus difficile" que $\mathcal{C}$

Complétude

Problème $\mathcal{C}$-complet

Soit $\mathcal{C}$ une classe de complexité (NP, EXPTIME, ...) et Π un problème de décision. On dit que Π est $\mathcal{C}$-Complet si $\Pi \in \mathcal{C}$ et s'il est $\mathcal{C}$-difficile.



Π_1 est un des pb les plus difficiles de $\mathcal{C}$

Résultats importants

Démontrer la difficulté

(1) • Si Π_1 est $\mathcal{C}$-Difficile et si $\Pi_1 \preccurlyeq \Pi_2$ alors Π_2 est $\mathcal{C}$-Difficile.

P et NP

(2) • Si $\Pi_2 \in$ P et $\Pi_1 \preccurlyeq \Pi_2$, alors $\Pi_1 \in$ P. $\longrightarrow$ Si $\Pi_1 \notin P$ $\Pi_2 \notin P$

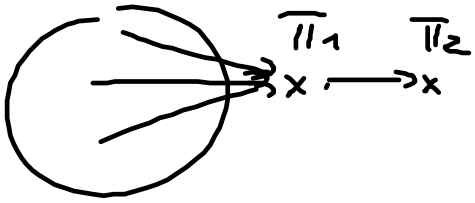
(3) • Si $\Pi_2 \in$ NP et $\Pi_1 \preccurlyeq \Pi_2$, alors $\Pi_1 \in$ NP. Si $\Pi_1 \notin NP$ $\Pi_2 \notin NP$

(4) • P $\neq$ NP $\Leftrightarrow \forall$ problème Π NP-Complet, $\Pi \notin P$.

(5) • 3-SAT est NP-Complet.

(1)+(5) Si $\Pi \geqslant$ 3-SAT et $\Pi \in NP$ alors Π NP-Complet

(1) Mq si Π_1 $\mathcal{C}$-difficile et $\Pi_2 \geq \Pi_1$ alors Π_2 $\mathcal{C}$-difficile



$\forall \Pi \in \mathcal{C}$
$\Pi \leq \Pi_1$
$\Pi_1 \leq \Pi_2$
par transitivité
$\forall \Pi \in \mathcal{C}$
$\Pi \leq \Pi_2$
$\Rightarrow \Pi_2$ $\mathcal{C}$-difficile

Mq $\leq$ est transitif :
Soit Π_1, Π_2, Π_3 / $\Pi_1 \leq \Pi_2$ et $\Pi_2 \leq \Pi_3$ Mq $\Pi_1 \leq \Pi_3$
$\exists$ R, R' algo poly tq
Si I instance de Π_1 R(I) instance de Π_2 tq I OUI $\Leftrightarrow$ R(I) OUI
Si J instance de Π_2 R'(J) instance de Π_3 tq J OUI $\Leftrightarrow$ R'(J) OUI
Posons R'' = R' $\circ$ R Si I instance de Π_1 R''(I) = R'(R(I)) instance de Π_3
I OUI $\Leftrightarrow$ R(I) OUI $\Leftrightarrow$ R'(R(I)) OUI
- R(I) se fait en temps $\Theta(|I|^c)$ pour un $c \in \mathbb{N}$
donc $|R(I)| \leq \Theta(|I|^c)$
R'(J) se fait en temps $\Theta(|J|^{c'})$ pour un $c' \in \mathbb{N}$
donc R'(R(I)) se fait en temps $\Theta(\Theta(|I|^c)^{c'}) = \Theta(|I|^{c \cdot c'})$
POLYNOMIAL

$\Rightarrow \Pi_1 \leq \Pi_3$

Si $\Pi_2 \in P$ (NP) et $\Pi_1 \leq \Pi_2$ alors $\Pi_1 \in P$ (NP)

$\exists$ un algo (mondét) A_2 qui résout Π_2 en temps poly $O(n^c)$ pour un $c \in \mathbb{N}$

$\exists$ une réduction (poly) R de Π_1 vers Π_2

Soit I instance de Π_1

Soit A_1 l'algo qui renvoie $A_2(R(I))$

- R et A_2 polynomiaux $\rightarrow$ A_1 polynomial (on prouve que $R \circ R'$ est poly dans le slide précédent)
- $A_1(I) = $ OUI $\not\Leftrightarrow$ $R(I)$ est une instance positive de Π_2 $\Leftrightarrow$ I est une instance positive de Π_1

(car A_2 résout Π_2 ; car R réduction)

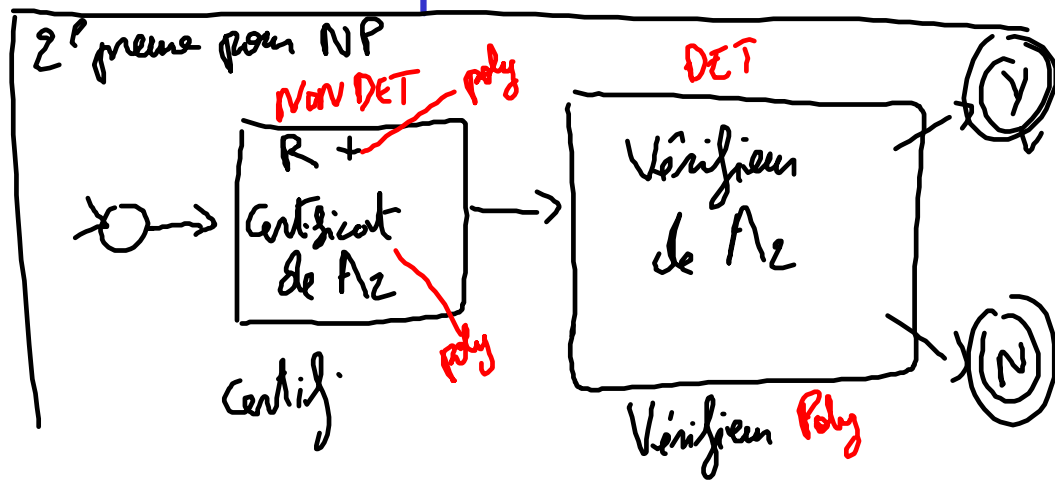
I OUI $\Leftrightarrow$ $R(I)$ OUI $\Leftrightarrow$ $\exists$ un choix mondet / $A_2(R(I))$ répond OUI

$\Leftrightarrow$ $\exists$ un choix mondet / $A_1(I)$ répond OUI

- A_1 mondét (car A_2 est non det)

$\Rightarrow$ A_1 résout Π_1 en temps polynomial

$\Rightarrow$ $\Pi_1 \in P$ (NP)



$P \neq NP \iff \forall \Pi$ NP-complet, $\Pi \notin P$

($\Leftarrow$) • Si $\exists \Pi$ NP-complet, $\Pi \in NP$ et $\Pi \notin P \Rightarrow P \neq NP$

Définition de NP-complet — Hypothèse

• 3SAT NP-Complet

($\not\Rightarrow$) contraposée :

Si $\exists \Pi$ NP-complet / $\Pi \in P$

$\Rightarrow \forall \Pi' \in NP \quad \Pi' \leq \Pi$ (car Π NP-difficile)

or si $\Pi_1 \leq \Pi_2$ et $\Pi_2 \in P$, $\Pi_1 \in P$ (résultat précédent)

alors $\Pi' \in P$ (car $\Pi \in P$)

donc $NP \subseteq P$

or $P \subseteq NP \Rightarrow P = NP$

Réduction polynomiale de Turing

Définition : Réduction polynomiale de Turing

Soient deux problèmes Π_1 et Π_2, une réduction polynomiale de Turing de Π_1 vers Π_2 est un algorithme $\mathcal{R}$ tel que

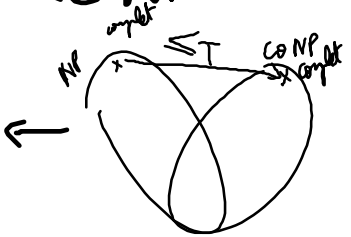
- $\mathcal{R}$ peut appeler un algorithme $\mathcal{R}'$ (imaginaire), appelé *oracle*, résolvant Π_2 en temps constant ;
- $\mathcal{R}$ résoud Π_1 en temps polynomial.

On note $\Pi_1 \preccurlyeq_T \Pi_2$.

Π_1 et Π_2 ne sont pas nécessairement des problèmes de décision.

Si $\Pi \in NP$, $co\text{-}\Pi \in co\,NP$

Tout pb NP complet est NP diff vis à vis de $\leq_T$



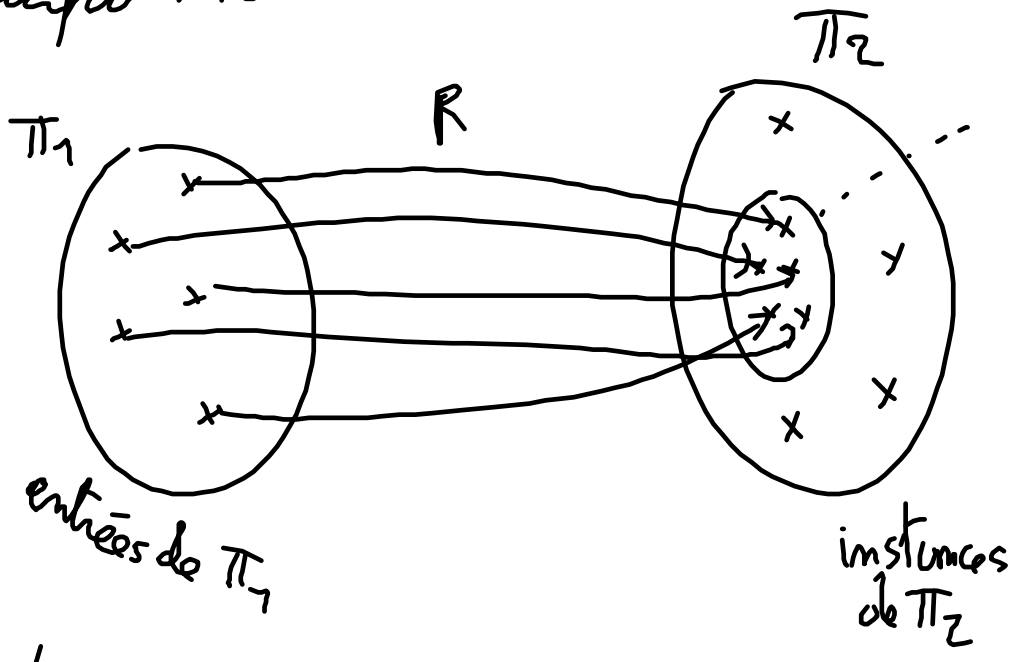
$\Pi \leq_T co\text{-}\Pi$ et $co\text{-}\Pi \leq_T \Pi$

ou $\Pi \not\leq co\,\Pi$ et $co\text{-}\Pi \not\leq \Pi$

Autres réductions

- Réduction exponentielle
- Réduction en espace polynomial
- Réduction probabiliste
- ...

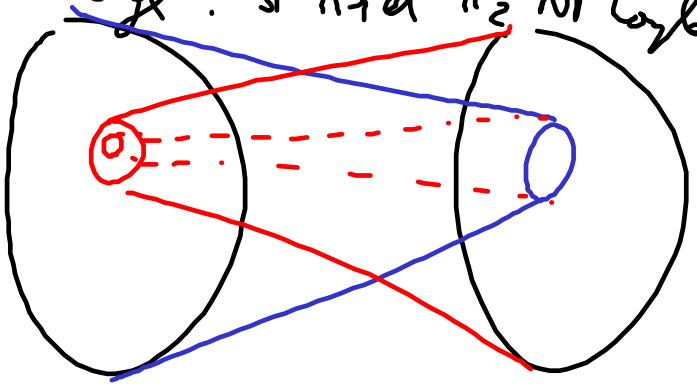
Pourquoi réduction ?



$R(\Pi_1)$ Image de R

"Réduction" car l'image de R ≠ toutes les instances de Π_2

Truc étrange : si Π_1 et Π_2 NP complet : $\Pi_1 \leq \Pi_2$ / $\Pi_2 \leq \Pi_1$

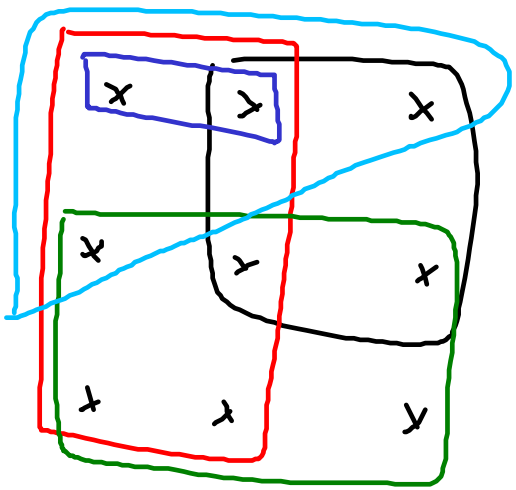


(Set Cover) Soit X un ensemble (de trucs)

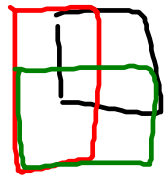
Soit S un sous-ensemble de $\mathcal{P}(X) = 2^X$ = parties de X

Soit $k \in \mathbb{N}$, $\exists ?$ $C \subseteq S$, $|C| \leq k$ / $\bigcup_{s \in C} s = X$

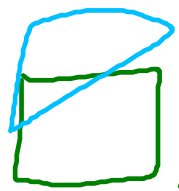
Taille de l'instance $|X| + |S||X| + \log(k)$



$k=3$: OUI

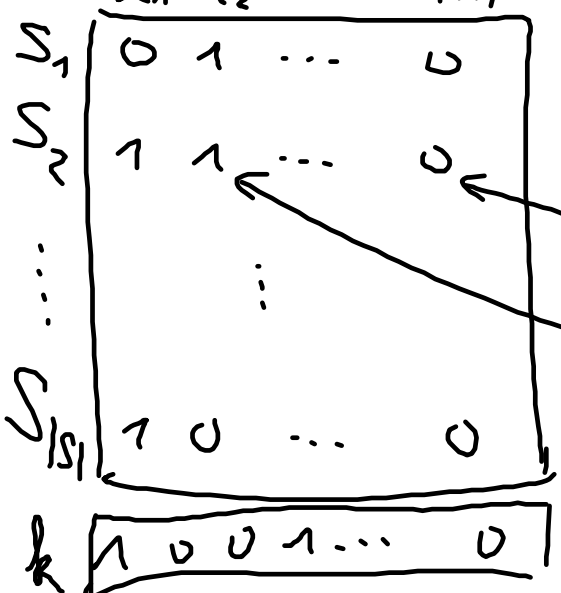


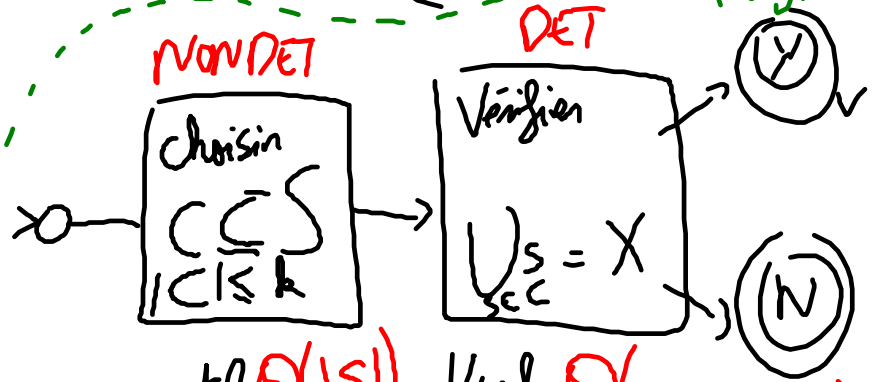
$k=2$: OUI



$k=1$: NON

Encodage $x_1 x_2 \dots x_{|X|}$





certif $\Theta(|S|)$ ou $\Theta(k)$ si $k \leq |S|$

Vérif $\Theta\left(\sum_{s \in C} |s| + |X|\right) = \Theta(|X||C|) = \Theta(|X||S|)$

$\forall s \in S$ ajouter ou non s à C

si $|C| = k$ stop

polynomial

$\Rightarrow$ Set Cover $\in$ NP

3 SAT :

Soit $\varphi = C_1 \wedge C_2 \wedge C_3 \cdots C_m$ une formule

où $C_j = (l^j_1 \vee l^j_2 \vee l^j_3)$

avec $l^j_{1,2,3} \in \{x_1 \overline{x_1}\ x_2 \overline{x_2} \cdots x_n \overline{x_n}\}$

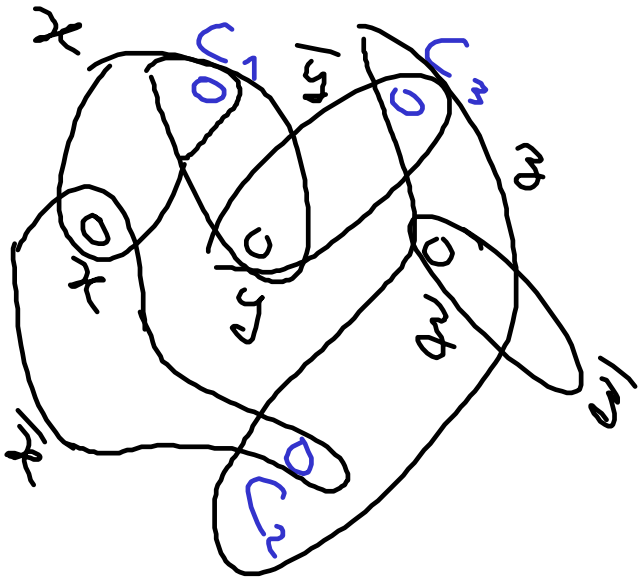
φ est elle satisfiable ?

Taille de l'instance : $n + m$

Mq 3SAT $\leq$ SET COVER

$\varphi_1 = (x \vee y) \wedge (\bar{x} \vee 3) \wedge (\bar{y} \vee 3)$ (C_1, C_2, C_3)

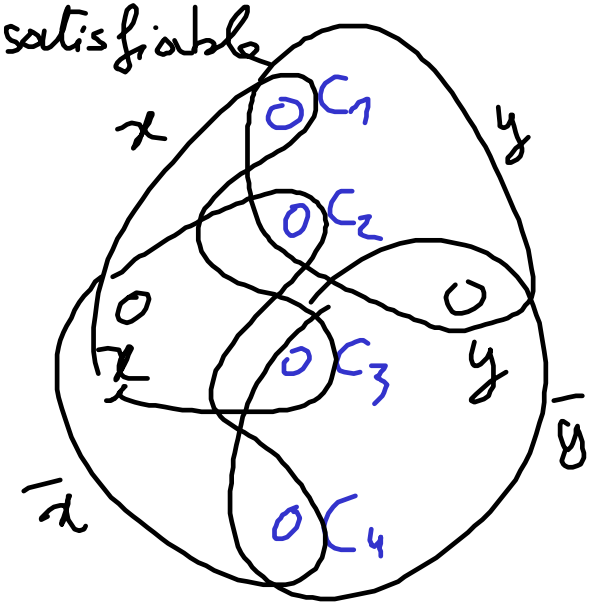
Satisfiable



$k = n = 3$

$\varphi_2 = (x \vee x) \wedge (\bar{x} \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee \bar{y})$ (C_1, C_2, C_3, C_4)

Non satisfiable



$k = 2$