

M1if09 – Calculabilité & complexité

Sylvain Brandel

2021 – 2022

sylvain.brandel@univ-lyon1.fr

COMPLEXITÉ

CLASSES P ET NP

La classe P

- Exemple 1 : voyageur de commerce

Visite de n villes en faisant le moins de km possibles.

Algorithme ?

produire toutes les permutations de villes possibles (sauf la première qui est toujours la même),

pour chaque permutation, calculer le trajet.

→ $(n - 1) !$ permutations possibles.

La classe P

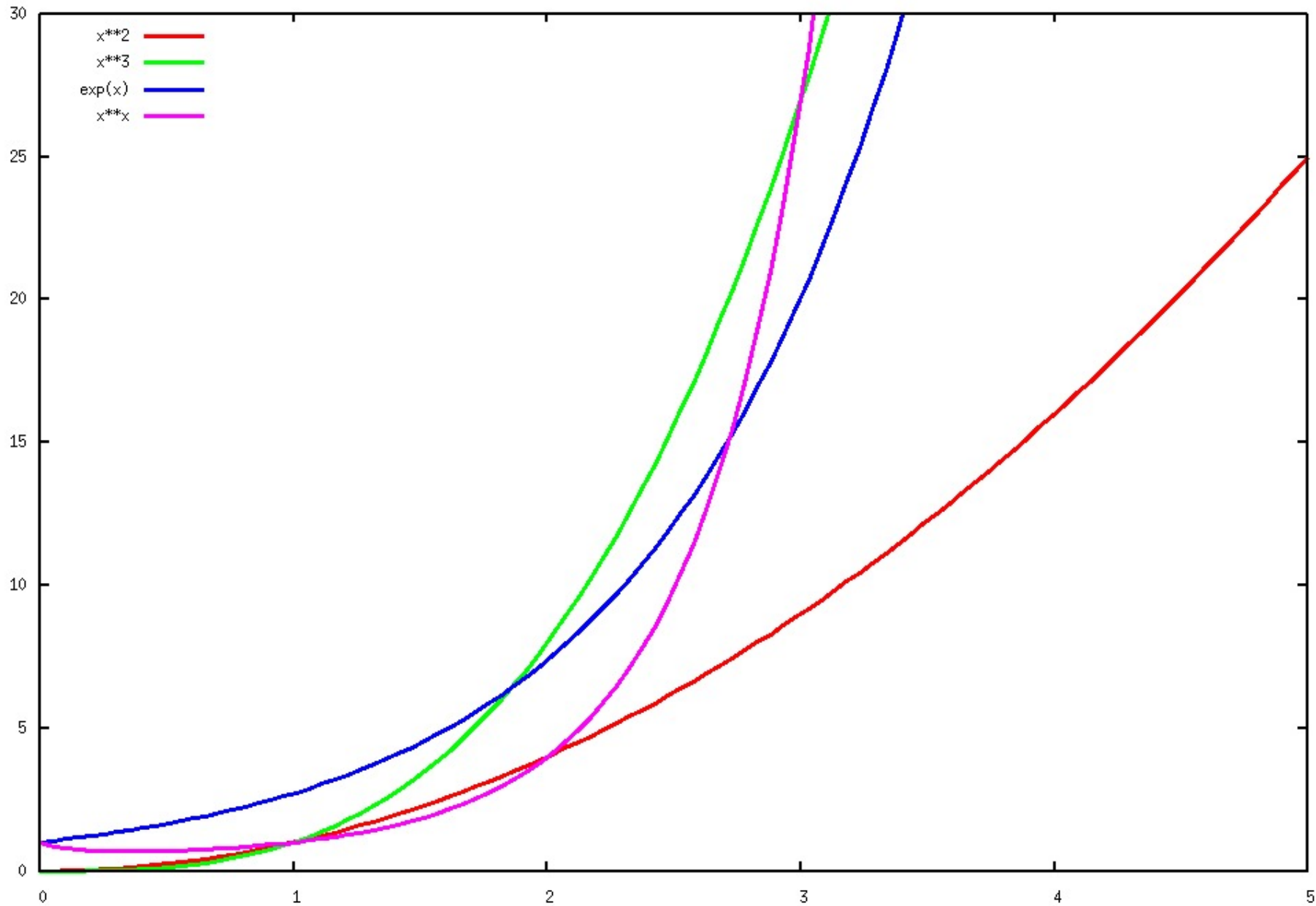
- **Ex. 2 : géométrie tortue**

```
void triangles1(float x, float y, float d, float h) {  
    if (y+d < h) {  
        tracer(x-d, y+d);  
        triangles1(x-d, y+d, d, h);  
        tracer(x+d, y+d);  
        triangles1(x+d, y+d, d, h);  
        tracer(x, y);  
    }  
}
```

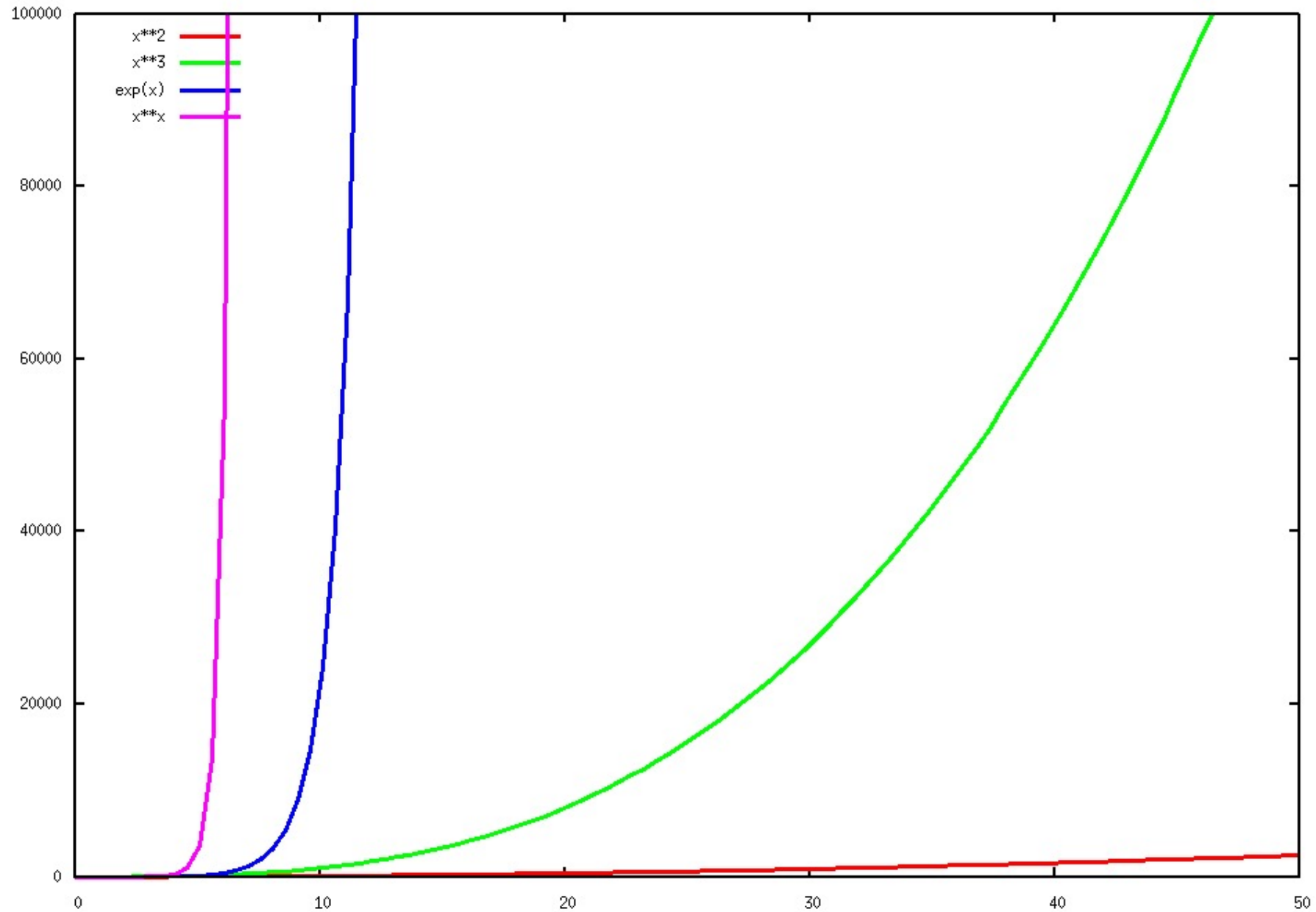
appel avec :

```
placer(x, y); triangles1(x, y, d, h);
```

La classe P



La classe P



Machine de Turing

- Machine de Turing **standard** : quintuplet $M = (K, \Sigma, \Gamma, \delta, q_0)$
 - K : ensemble fini d'états
 - Σ : alphabet d'entrée
 - Γ : alphabet des symboles du ruban
 - δ : **fonction** de transition
fonction partielle $K \times \Gamma \rightarrow K \times \Gamma \times \{\leftarrow, \rightarrow, S\}$,
les symboles \leftarrow et \rightarrow désignent un déplacement élémentaire à gauche ou à droite, S pour Stay
 - $q_0 \in K$ est l'état initial.
- Machine de Turing **non déterministe** : sextuplet $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$
 - Δ : **relation** de transition $\Delta \subset K \times \Gamma \times K \times \Gamma \times \{\leftarrow, \rightarrow, S\}$
 - $F \subset K$: états acceptants
 - Plusieurs sorties différentes pour la même entrée
 - **Accepteur** dont le seul résultat qui nous intéresse est de savoir si la machine s'arrête ou non

La classe P

- Définition

Une machine de Turing **déterministe** $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ est dite **polynomialement bornée** s'il existe un polynôme p tel que pour toute entrée w , il n'y a **pas** de configuration C telle que

$$(q_0, \#w) \vdash_M^{p(|w|+1)} C$$

c'est-à-dire M **s'arrête** toujours, et ce, en au plus $p(|w|+1)$ étapes

Un langage est dit **polynomialement décidable** ssi il existe une machine de Turing polynomialement bornée qui le décide

- La classe des langages polynomialement décidables est notée **P**

Thèse de Church – Turing

- *Un algorithme est une machine de Turing qui s'arrête pour toutes ses entrées*
- *We therefore propose to adapt the Turing machine that halts on all inputs as the precise formal notion corresponding to the intuitive notion of an "algorithm"*
- *Les langages reconnus par une procédure effective sont ceux décidés par une machine de Turing*
- *Every computational process that is intuitively considered to be an algorithm can be converted to a Turing machine*

La classe P

- La thèse de Church-Turing est raffinée en :
Les machines de Turing polynomialement bornées et la classe P correspondent aux notions
 - D'algorithmes **pratiquement exécutables**
 - Et de problèmes réellement solvables
- Théorème
La classe P est stable par complément
- Théorème
Il existe des langages rékursifs non polynomialement décidables

Exemples de problèmes et de complexité

- Exemple 1 : existence d'un chemin

Soient un graphe orienté $G \subset V \times V$ ($V = \{v_1, \dots, v_n\}$)

et deux sommets v_i et $v_j \in V$

Existe-t-il un chemin entre v_i et v_j ?

⇒ Il existe un algorithme en $O(n^3)$:

calcul de la fermeture réflexive – transitive

Exemples de problèmes et de complexité

- Exemple 2 : graphes Eulériens

Soit un graphe G .

Existe-t-il un chemin fermé (cycle) dans G qui utilise chaque **arête** une fois et une seule ?

Un graphe qui contient un tel cycle est dit Eulérien ou unicursal

Le problème du cycle Eulérien $\in P$

Exemples de problèmes et de complexité

- Exemple 3 : graphes Hamiltoniens

Soit un graphe G

Existe-t-il un cycle passant par chaque **sommet** une fois et une seule ?

Un graphe qui contient un tel cycle est dit Hamiltonien

Algorithme :

- Examiner toutes les permutations de nœuds possibles
(→ exponentiel)
- Regarder si le parcours correspondant existe dans G

Le problème des cycles Hamiltoniens n'est pas **connu** comme étant dans P

Classe NP

- Définition

Une machine de Turing **non déterministe** $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$ est dite **polynomialement bornée** s'il existe un polynôme p tel que :

pour toute entrée w , il **n'y a pas** de configuration C telle que

$$(q_0, \#w) \vdash_M^{p(|w|+1)} C$$

c'est-à-dire M **s'arrête** toujours, et ce, en au plus $p(|w|+1)$ étapes.

- NP est la classe des langages décidés par une machine de Turing non déterministe polynomialement bornée

NP pour *Non déterministe Polynomial*

Classe NP

- Rappel : Logique d'ordre 0 – calcul des propositions :
 - Variables booléennes
 - Connecteurs (opérations) : \wedge , \vee , \Rightarrow , \Leftrightarrow et \neg
 - Un **littéral** est une expression de la forme p ou $\neg p$ pour une variable propositionnelle p
 - Une **clause** est une disjonction de variables propositionnelles ou de leur négation :
$$E = x_1 \vee x_2 \vee \dots \vee x_n \text{ (chaque } x_i \text{ est un littéral)}$$
 - **Forme normale conjonctive**
$$F = E_1 \wedge E_2 \wedge \dots \wedge E_n \quad \text{(chaque } E_i \text{ est une clause)}$$
$$= \bigwedge_{i=1}^n \left(\bigvee_{j=1}^{P_i} x_{ji} \right) \quad \text{avec } x_{ji} = a_{ji} \text{ ou } \neg a_{ji} \quad \text{(atomes, littéraux positifs ou négatifs = clauses)}$$
 - Assignment booléenne ou **interprétation** :
 - fonction : $X \rightarrow \{T, \perp\}$ X : ensemble des variables
 T : vrai, \perp : faux
 - **Satisfiabilité** : \exists (au moins) une interprétation rendant la formule vraie

Classe NP

- Énoncé du problème SAT

Soit une forme normale conjonctive F
 F est-elle satisfiable ?

- Proposition

Le problème SAT est dans NP

Classe NP

- Algorithme brutal pour le problème SAT
 - Faire une table de vérité
 - exponentiel en fonction du nombre de variables.
 - Pour chaque assignation, tester la FNC (linéaire)
 - exponentiel
- 2-SAT (au plus 2 littéraux par clause) : $(a \vee b) \wedge (c \vee d) \wedge e \wedge \dots$
 - 2-SAT $\in P$ (algorithme P connu, même si table de vérité de taille exponentielle)
- 3-SAT (exactement 3 littéraux par clause) : $(a \vee b \vee c) \wedge (d \vee e \vee f) \wedge \dots$
 - 3-SAT $\in NP$ (pas d'algorithme P connu, algorithme NP connu)
- SAT (FNC avec clauses avec nombre quelconque de littéraux)
 - SAT $\in NP$

Classe NP

- Voyageur de commerce (faible)

Soient un entier $n \geq 2$, une matrice de distance d_{ij} et un entier $B \geq 0$.

(B est le **budget** du voyageur de commerce)

Le problème consiste à trouver une permutation π sur $\{1, 2, \dots, n\}$ telle que $C(\pi) \leq B$, où :

$$C(\pi) = d_{\pi(1) \pi(2)} + d_{\pi(2) \pi(3)} + \dots + d_{\pi(n-1) \pi(n)} + d_{\pi(n) \pi(1)}$$

- Proposition

Le problème du voyageur de commerce est dans NP

Classe NP

- De manière semblable, des algorithmes de type :
 - Générer de manière non déterministe une situation
 - Tester de manière déterministe cette situation

Peuvent résoudre des problèmes précédents avec des machines de Turing ND polynomialement bornées

- Si le test de la situation se fait avec une machine de Turing (déterministe) polynomialement bornée
- Et si la taille de la situation est bornée polynomialement en fonction des entrées

Classe NP

- 2-SAT \in P, 3-SAT \in NP
 \Rightarrow SAT \in NP
 ce qui ne veut pas dire que SAT \notin P...
- P \subseteq NP (clair)
- NP \subseteq P ? Cela voudrait dire :
 - 1) Qu'il existe une MT polynomialement bornée équivalente à une MT ND polynomialement bornée
 - 2) Que les problèmes SAT, voyageur de commerce, cycle de Hamilton, etc. seraient dans P
- En fait, **on ne sait pas** si NP \subseteq P (et donc P = NP).

Classe NP

- Définition (borne exponentielle)

Une MT $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ est dite **exponentiellement bornée** s'il existe un **polynôme** p tel que :

pour toute entrée w , il n'y a pas de configuration C telle que

$$(q_0, \#w) \vdash_M 2^{p(|w|)+1} C$$

Cette machine **s'arrête** toujours, et ce en au plus $2^{p(|w|)+1}$ étapes

- On note EXP la classe des langages qui peuvent être décidés par une MT exponentiellement bornée

- Théorème

Si $L \in NP$, alors $L \in EXP$

Autrement dit : $NP \subseteq EXP$

Classe NP

Certificat

- MT non dét. polynom. bornée : résolution de problèmes d'existence
 - Production d'une situation
 - Test
 - Réussite → problème d'existence résolu
 - Échec à tous → problème d'existence résolu (réponse négative)
- Certificat (témoin) :
 - Mot synthétisant une solution et passant le test avec succès
 - Tous les problèmes NP ont des certificats
 - Seuls les problèmes NP ont des certificats
 - En pratique
 - Longueur des certificats polynomiale par rapport aux entrées
 - Test des certificats en temps polynomial

Classe NP

Certificat

- Définition

Soient Σ un alphabet et ";" un symbole $\notin \Sigma$

Soit L' un langage tel que $L' \subseteq \Sigma^* ; \Sigma^*$

On dit que L' est **polynomialement équilibré** s'il existe un polynôme p tel que
si $x ; y \in L'$ alors $|y| \leq p(|x|)$

- Théorème

Soit $L \subseteq \Sigma^* ; \Sigma^*$ un langage et Σ un alphabet pour lequel ";" $\notin \Sigma$ et $|\Sigma| \geq 2$.

$L \in NP$ ssi il existe un langage polynomialement équilibré $L' \subseteq \Sigma^* ; \Sigma^*$ tel que :

$$L' \in P \text{ et } L = \{x \mid \exists y \in \Sigma^* : x ; y \in L'\}$$

- $L' = \{x ; y \mid y \text{ est un certificat pour } x\}$

- Si $x \in L$, alors il y a au moins un certificat

- Si L' existe, alors une MT non dét. décide L en testant tous les certificats, en utilisant une MT dét. décidant L' , donc $L \in NP$
- Si $L \in NP$, alors il existe une MT non dét. polynomialement bornée décidant L

- Exemple

- L : SAT ; L' : une interprétation validant chaque instance positive

- L : 3-COL ; L' : un coloriage pour chaque graphe 3-coloriable

- L : ensemble des nombre composites ; L' : la paire d'entiers composant chaque nombre