

IG4 - Polytech Montpellier

Vincent Berry & Jérôme Fortin

COMPLEXITÉ

Séance 2

Classes de problèmes & réductions



POLYTECH[®]
MONTELLIER

COURS 2 : LES DEUX GRANDES CLASSES DE PROBLÈMES

Plan d'attaque :

I. Intro : problèmes faciles vs problèmes difficiles

II. Problèmes faciles : la classe P

III. Problèmes difficiles : la classe NP

INTRODUCTION

- Pour certains problèmes, plusieurs algorithmes ont été proposés, parfois de complexité assez différente :
- certains algorithmes ont une **complexité exponentielle**
- d'autres ont une **complexité polynomiale**

- Exemple de Fibonacci :

$$\text{Fib}(0) = 1$$

$$\text{Fib}(1) = 1$$

$$\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2)$$

PLAN

I. Intro : problèmes faciles vs problèmes difficiles

II. Temps polynomial : la classe P

III. La classe NP

INTRODUCTION

Pour certains problèmes nous ne trouvons pas d'algorithme polynomial, est-ce dû à notre manque d'astuce ?

Exemple : un graphe G contient-il un *chemin hamiltonien* ?

(déf°: un chemin hamiltonien est un chemin qui passe une fois et une seule par chaque sommet)

On ne trouve pas d'algorithme polynomial...

Certains problèmes sont-ils intrinsèquement «difficiles» que d'autres ?

- ❖ pour certains problèmes, on ne trouve pas d'gorithme polynomial



...une définition ! (dichotomie sur la nature des problèmes)

Problème facile* =

problème qui admet un algorithme polynomial pour le résoudre

* «tractable» in English in the text

- Rmq : pourquoi «polynomial» et pas $\mathcal{O}(n^2)$?

Arguments en faveur de cette définition :

- ★ en pratique, la majeure partie des algorithmes polynomiaux ont un faible exposant
- ★ si on passe d'un modèle de calcul connu et opérationnel à un autre (M. Turing, RAM, ordinateur moderne) ou de puissance de machine, on garde l'aspect polynomial

PROBLÈMES DE DÉCISION



problème de la vraie vie à traiter
→ formalisation / modélisation



→ un **problème Q** est une relation entre un ensemble **I** d'**instances** (*données possibles*) et un ensemble **S** de «**solutions**».

- *Exemple : Plus-Court-Chemin (PCC)*
- INSTANCE
- SOLUTION

$$G = (S, A)$$

$$d, f \in S$$

séquence de sommets de G

PROBLÈMES DE DÉCISION



- Pour simplifier on se limite aux **problèmes de décision** :
- Ex : **k**-Vertex-Cover (**k**-PCC)

Question : «existe-t-il **OUI** ou **NON** une couverture du graphe utilisant moins de **k** sommets ?»

- INSTANCE

$$G = (S, A)$$

un entier k

- SOLUTION

{ oui , non }

1 , 0

PROBLÈMES DE DÉCISION

- Ce passage aux pbs de décision n'est **pas vraiment restrictif** :
 - si on sait résoudre le pb d'optimisation en temps polyn. -> on sait résoudre le pb de décision en temps polynomial aussi :
 1. résoudre pb optimisation -> sol° s
 2. comparer *valeur(s)* à k -> répondre au pb de décision
 - si le pb de décision est difficile -> le pb d'optimisation est difficile aussi



pour la suite de ce cours : on ne considère que des pbs de décision

Exercices

- Formulez les problèmes suivants sous la forme de problèmes de décision :
 - Voyageur de commerce (TSP) :
 - Clique maximum :

LA CLASSE P

- Un algorithme résout un pb Q en $O(T(n))$
si pour toute instance i de taille n , il fournit la solution
en temps au plus $O(T(n))$
- Un problème Q peut être résolu en temps polynomial
s'il existe un algo A qui le résout en $O(n^k)$ où k est une
constante.
- la classe P est l'ensemble des problèmes de décision
que l'on peut résoudre en temps polynomial

APPLICATION

Nous considérons ici une restriction du problème SAT nommée **HORN-SAT** : une instance est une formule booléenne B représentée sous la forme d'une conjonction de clauses disjonctives ayant au plus un littéral positif dans chaque clause, par ex :

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_2 \vee x_3) \wedge (x_4)$$

La question à ce problème est de savoir s'il existe une affectation Aff de valeurs aux variables x_i t.q. $\text{Aff}(B) = V$ RAI

Les formules Horn interviennent en intelligence artificielle pour développer des systèmes experts ou formaliser des bases de connaissances.

Question 1 : Que signifie que **HORN – SAT** ∈ P ?

- On peut trouver en temps polynomial une affectation Aff de valeurs aux variables tel que $\text{Aff}(B) = \text{VRAI}$
- On peut trouver une affectation V de valeurs à un nombre polynomial de variables t.q. $\text{Aff}(I) = \text{VRAI}$
- On peut savoir en temps polynomial s'il existe ou non une affectation A de valeurs aux variables t.q. $\text{Aff}(B) = \text{VRAI}$
- la taille d'une instance de HORN-SAT est polynomiale en le nombre de variables

PLAN

III. Problèmes difficiles : la classe NP

III.1. Mise en situation

III.2. Validation en temps polynomial

III.3. Les classes NP et NPC

III.4. Lien entre les classes de problèmes

III.1 MISE EN SITUATION



Vous pouvez y aller
Charles. Nous comptons
sur vous !

augmentation / démission

«Nous allons nous lancer sur le marché à forte valeur ajoutée des *schtrapalouxes*. On a besoin d'une bonne **méthode** pour déterminer si **oui** ou **non** un nouveau composant peut **remplacer** un **composant** traditionnel de ces machines.

pb de décision

Puisque vous êtes notre ingénieur informaticien, vous aurez en charge de trouver un programme **efficace**.

classe P

- Vous filez illico au dépt. des *schtrapalouxes*, qui vous confirme que ces machines contiennent des composants, chacun étant un **assemblage de n schmurtzes**, produisant une certaine **sortie** suivant une **donnée** reçue sur plusieurs capteurs.
- Vous formalisez le pb de remplacement du composant traditionnel par un autre, puis cherchez ardemment un **algorithme pour déterminer un assemblage de schmurtzes rendant les mêmes services que le composant à remplacer**.
- Trois semaines après, vous ne parvenez toujours pas à produire le moindre algorithme polynomial...
 - Vous avez bien un algorithme pour savoir si un composant peut en remplacer un autre, mais il nécessite d'**examiner tous les ordres possibles** des **n schmurtzes** ... c-à-d ?

Question pour vous

- ... c-a-d des années étant donné que $n > 1000$!

III.1 MISE EN SITUATION

Vous commencez à craindre pour la prochaine réunion avec le boss



III.1 MISE EN SITUATION

L'idéal serait de pouvoir annoncer :



Ceci implique de prouver que le problème est par nature «intraitable», et de montrer qu'aucun algorithme efficace ne peut exister pour le résoudre.

III.1 MISE EN SITUATION

- Malheureusement, prouver qu'un problème est de façon inhérente intraitable, peut être aussi difficile que trouver un algorithme efficace pour le résoudre !
- Maisla **théorie de la NP-complétude** fournit des techniques simples pour montrer qu'un **pb est «aussi difficile»** que nombre de problèmes célèbres qui ont résisté aux experts depuis des dizaines d'années.



III.1 MISE EN SITUATION

Armé de cette techniques, vous pourrez montrer que le problème de remplacement d'un composant de *schtrapalouxe* est NP-complet, et sereinement annoncer :



III.1 MISE EN SITUATION

- Maintenant que vous avez sauvé votre tête, vous pouvez utiliser votre temps pour
 - tester un algorithme **exponentiel** dans le pire des cas mais nécessitant de **faibles temps calculs** pour la très grande majorité des **instances** considérées.
 - ajouter des contraintes au pbm pour trouver un algorithme **efficace** qui répond juste dans 99% des cas constraints.
 - trouver un algo **efficace** mais approximatif, c-à-d tolérant une légère perte d'efficacité dont l'ampleur peut toutefois être prouvée faible, dans le pire des cas.

La preuve de difficulté n'est donc qu'une étape sur la route conduisant à des programmes efficaces



III.2 VALIDATION EN TEMPS POLYNOMIAL

Avant de résoudre le pbm, posons-nous la question de reconnaître / valider une solution

pb de la classe P
[Dijkstra]

CHEMIN

Donnée : $\langle G, u, v, k \rangle$

Question : Existe-t-il un chemin de u à v dans G de $\lg < k$?

- Supposons que l'on nous donne une solution potentielle $S=[s_1, s_2, \dots, s_l]$ pour CHEMIN.
- On s'intéresse ici à la validation de S :
 - valider S signifie vérifier que S est bien un chemin joignant u à v et que S est de longueur $< k$
 - si S est valide, S peut être vu comme un «certificat» de la réponse «oui» au problème.

Exercice : si l'on veut un algorithme **de validation linéaire** pour CHEMIN, quelle structure de données et quel algorithme pour représenter G ?

III.2 VALIDATION EN TEMPS POLYNOMIAL

Avant de résoudre le pbm, posons-nous la question de reconnaître / valider une solution

CHEMIN

Donnée : $\langle G, u, v, k \rangle$

Question : Existe-t-il un chemin de u à v dans G de $\lg < k$?

- Supposons que l'on nous donne une solution potentielle $S = [s_1, s_2, \dots, s_i]$ pour CHEMIN.

- valider S pour CHEMIN

prend un temps polynomial

D'acc, mais **résoudre** CHEMIN prend aussi un temps polynomial...

[Dijkstra]



III.2 VALIDATION EN TEMPS POLYNOMIAL

CYCLE HAMILTONIEN

Donnée : $\langle G \rangle$

Question : Existe-t-il un cycle passant une et une seule fois par chaque sommet de G ?

pas d'gorithme efficace connu

- résolution par énumération : $O(n!)$ (permutations)
- ... mais facile de vérifier qu'une liste des sommets est un cycle hamiltonien :

Algo de validation $\in P$

- $G=(S,A)$
- $[s_1,s_2,\dots,s_i]$ séquence de sommets

Exercice : choisissez une structure de données et donnez un algorithme de validation linéaire pour CYCLE HAMILTONIEN

III.2 VALIDATION EN TEMPS POLYNOMIAL



- Algorithme de validation :

c'est un algorithme prenant deux entrées :

- ▶ x une «**instance**» du pbm
- ▶ y un **certificat** (souvent une solution potentielle)

- On dit aussi que l'algo A valide x si $A(x,y) = \text{oui}$

Exemple : $A(<\!G\!>, <\!s_1, \dots, s_n\!>)$ valide G si s_1, \dots, s_n est une permutation des sommets de G qui est un cycle hamiltonien

111.3 LES CLASSES NP ET NPC



La classe NP :

un pb Q est dans la classe NP s'il existe un algorithme de **validation** en **temps polynomial** pour Q .

Exemples :

CYCLE HAMILTONIEN est dans NP

SAT est dans NP

ARRET PROGRAMME n'est pas dans NP



*NP ne veut pas dire Non-Polynomial,
mais «Non-Déterministe en tps Polynomial» (autre def° de NP)*

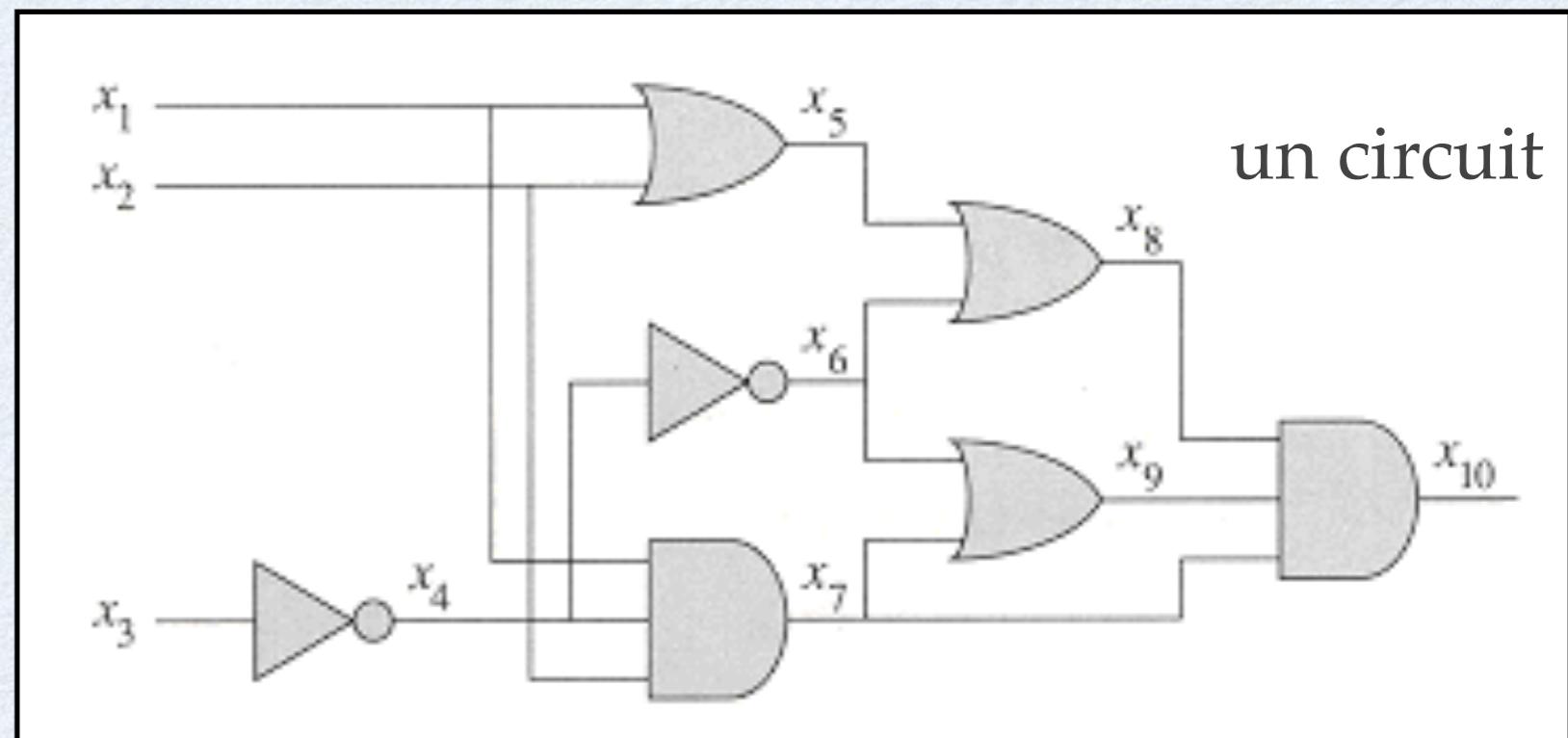
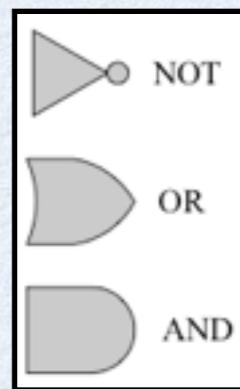
CIRCUIT SAT = un problème célèbre de la classe NP



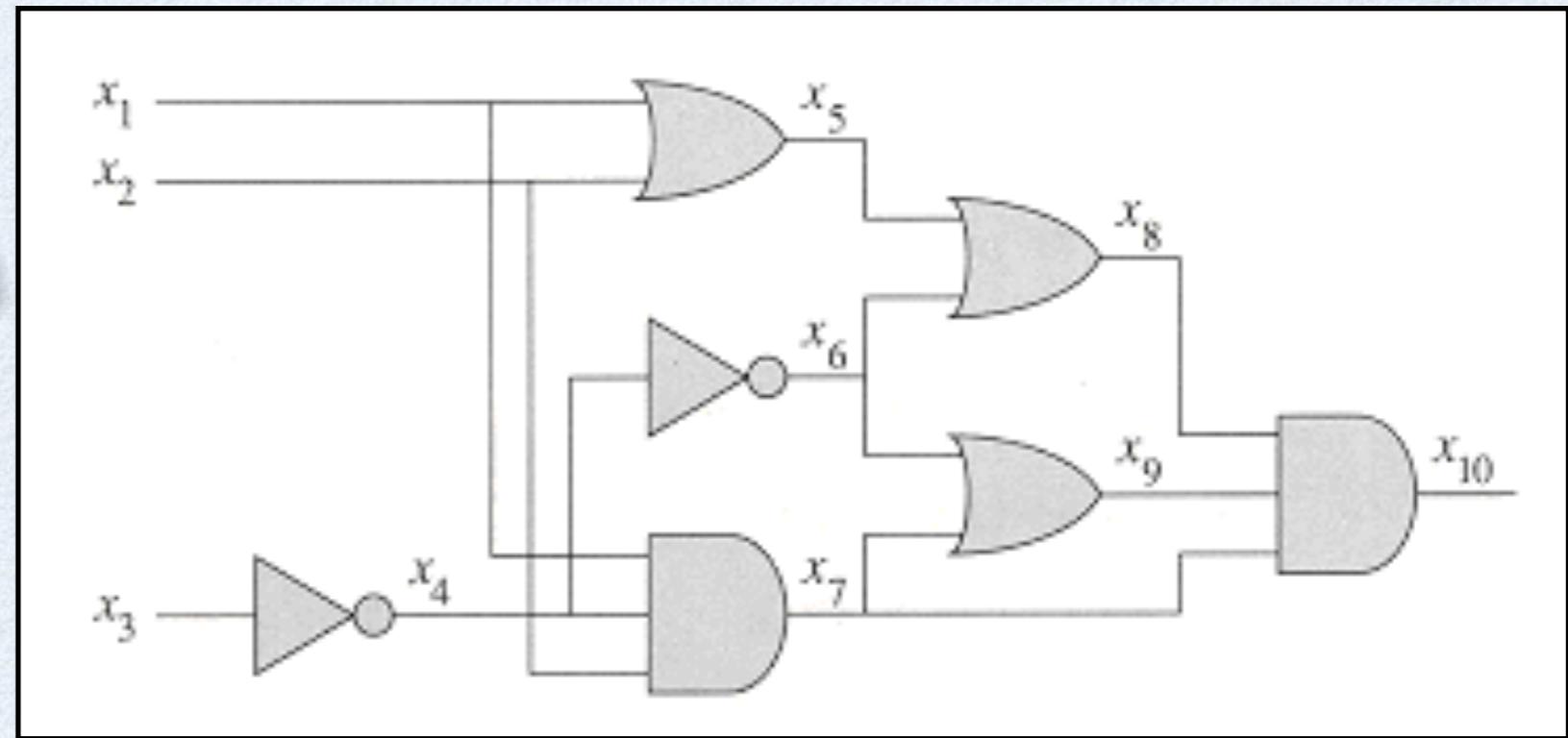
Donnée : <circuit de portes logiques ayant un seul point de sortie>

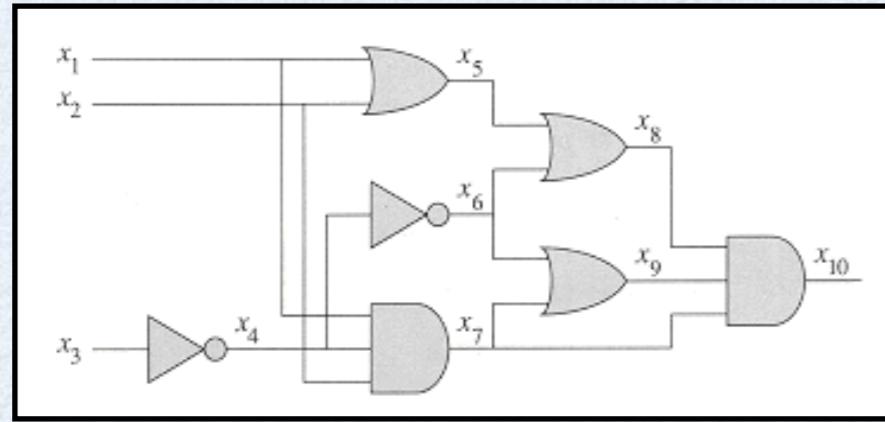
Question : Existe-t-il une affectation de valeurs aux entrées de façon à ce que le circuit renvoie «vrai» ?

portes logiques



Exercice 1 : prouvez que ce circuit est une instance positive de CIRCUIT SAT





Exercice 2 : prouvez que CIRCUIT SAT est dans NP

III.3 LES CLASSES NP ET NPC

Alors, dans la classe NP,
il y a un plein de
problèmes ?



Donc NP contient des
problèmes de difficultés
très différentes ... Y a-t-il
parmi eux des pbs
vraiment difficiles ?

Oui, y compris ceux
de la classe P !

Oui : les pbs NP-complets
! Ce sont les plus difficiles



- Montrez que tout problème p de P est dans NP

III.3 LES CLASSES NP ET NPC



Réduction d'un problème à un autre :

$$\begin{array}{rcl} f: Q & \longrightarrow & Q' \\ x & \longrightarrow & f(x) \end{array}$$

avec les mêmes réponses :

Q répond oui pour x ssi Q' répond
oui pour $f(x)$

f transforme une instance de Q en une instance de Q'

Notation :

$$Q \leq Q'$$

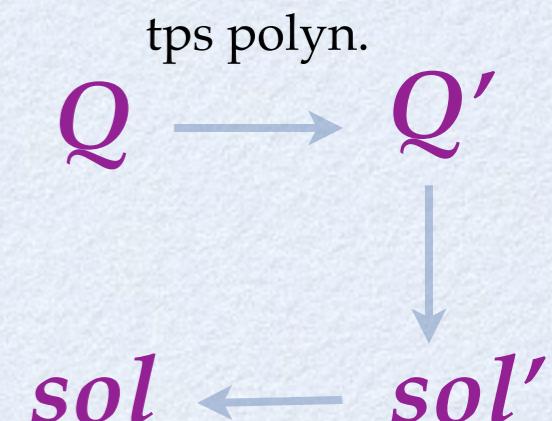
sens de la transformation

Idée : si on sait résoudre Q' alors on sait résoudre Q

III.3 LES CLASSES NP ET NPC

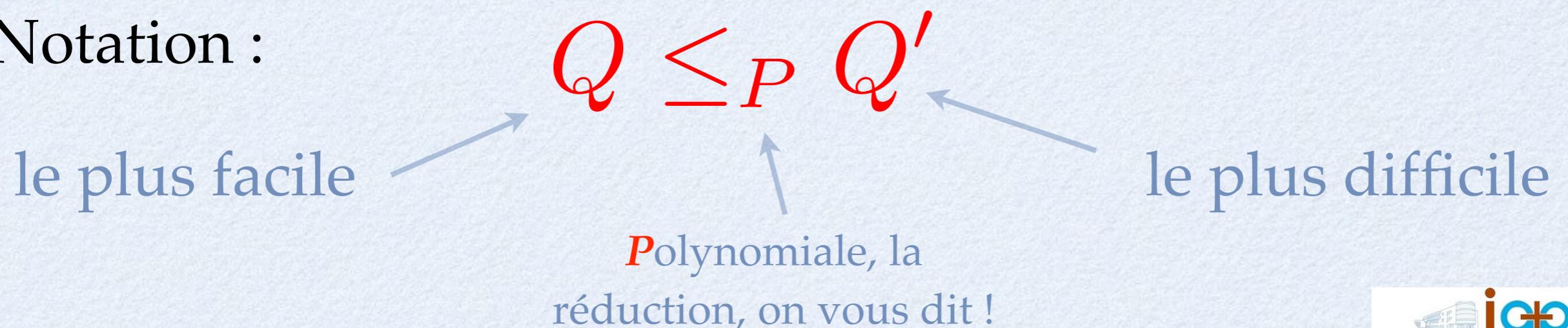
Réduction en temps polynomial :

Idée : un pb Q peut être ramené à un pb Q' si une instance de Q peut être «**facilement transformée**» en une instance de Q' , dont la solution fournira une solution à l'instance de Q .



Autrement dit, Q pas plus difficile que Q' .

Notation :



III.3 LES CLASSES NP ET NPC

Réduction en temps polynomial :

Exemple :

CLIQUE : $\langle G = (S, A), k \text{ entier} \rangle$ G a-t-il une clique de taille $\geq k$

STABLE : $\langle G = (S, A), k \text{ entier} \rangle$ G a-t-il un stable de taille $\geq k$

(un stable est un ensemble de sommets 2 à 2 non adjacents)

CLIQUE \leq_P STABLE

Exercice 1 : trouver la transformation
f() réduisant CLIQUE en STABLE

Exercice 2 : montrer que f()
s'effectue en temps polynomial

CLIQUE :

$\langle G = (S, A), k \text{ entier} \rangle$

G a-t-il une clique de taille $\geq k$

STABLE :

$\langle G = (S, A), k \text{ entier} \rangle$

G a-t-il un stable de taille $\geq k$

(un stable est un ensemble de sommets 2 à 2 non adjacents)

Exercice 1 : trouver la transformation f() réduisant CLIQUE en STABLE

Exercice 2 : montrer que f()
s'effectue en temps polynomial

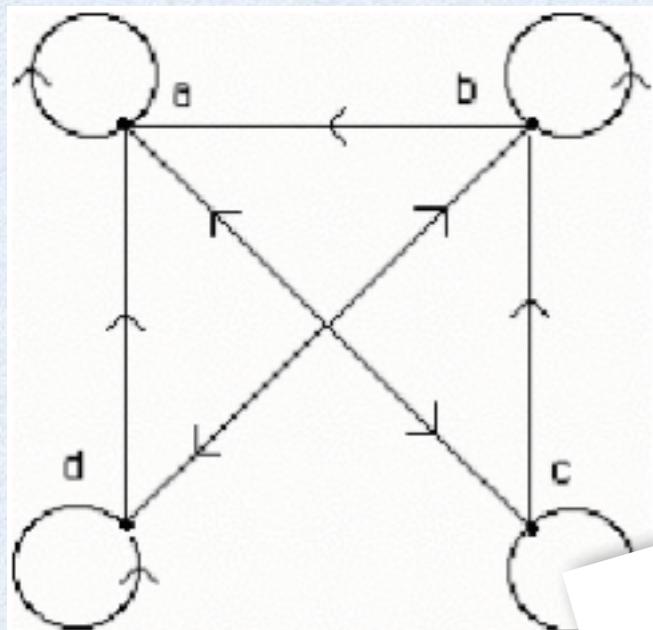
- CLIQUE** : $\langle G = (S, A), k \text{ entier} \rangle$ G a-t-il une clique de taille $\geq k$
- STABLE** : $\langle G = (S, A), k \text{ entier} \rangle$ G a-t-il un stable de taille $\geq k$
(un stable est un ensemble de sommets 2 à 2 non adjacents)

Exercice 2 : montrer que f()
s'effectue en temps polynomial

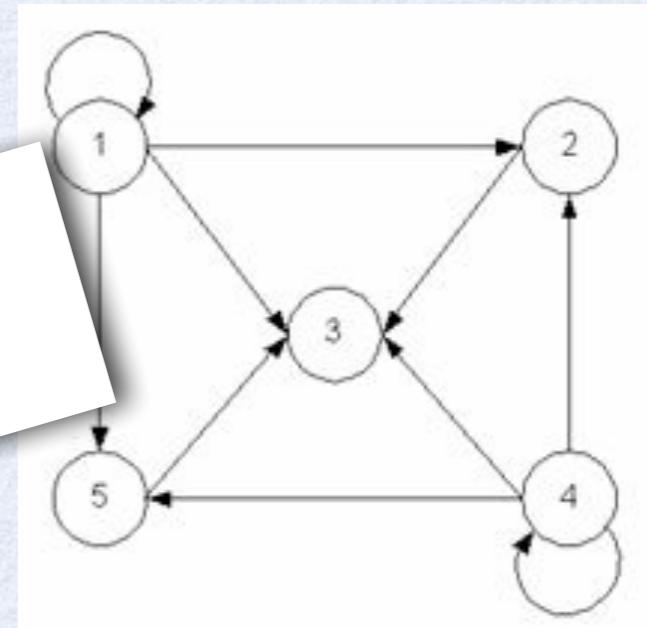
III.3 LES CLASSES NP ET NPC

Propriétés :

- si $Q_1 \leq_P Q_2$ alors $Q_2 \in P \Rightarrow Q_1 \in P$
- la relation \leq_P est reflexive et transitive



Exercice : pourquoi la première propriété est-elle vraie ?



Montrez que si $Q_1 \leq_P Q_2$ alors $Q_2 \in P \Rightarrow Q_1 \in P$

111.3 LES CLASSES NP ET NPC

Définitions :



NP-difficulté :

un pbm Q est NP-difficile si $\forall Q' \in NP, Q' \leq_P Q$



NP-complétude :

un pbm Q est NP-complet si $Q \in NP$ et Q est NP-difficile



La classe NPC :

NPC est le nom de la classe des problèmes NP-complets

III.3 LES CLASSES NP ET NPC

NP-difficulté :

un pbm Q est NP-difficile si $\forall Q' \in NP, Q' \leq_P Q$

NP-complétude :

un pbm Q est NP-complet si $Q \in NP$ et Q est NP-difficile



Les pbs NP-complets ont
l'air vraiment coriaces...

il en existe au moins ?

III.3 LES CLASSES NP ET NPC

Liste de problèmes NP-Complets :

CIRCUIT SAT

Voyageur de commerce

Circuit Hamiltonien

Coloriage d'un graphe avec k couleurs ($k > 3$)

Ordonnancement Unitaire (n tâches T_i de durée unitaire, un graphe de précédence, m machines identiques, un nombre entier B .

Question: Existe-t-il un ordonnancement de durée au plus B ?)

Partition (un ensemble A de n valeurs entières, **Question:** existe-t-il $B \subseteq A$ tel que $\sum b \in B = \sum a \in A \setminus B$?)

...

Preuves complètes parfois un peu complexes, mais sauriez-vous faire la partie montrant qu'ils sont dans NP ?

Exercice



III.3 LES CLASSES NP ET NPC

- En pratique, pour montrer qu'un pb est **NP-complet**, on ne doit heureusement pas trouver une réduction depuis *chaque* pb de la classe NP :

Lemme : Si Q est un pb t.q. $Q' \leq_P Q$ pour un certain pb $Q' \in NPC$ alors Q est NP-difficile. De plus, si $Q \in NP$ alors $Q \in NPC$.

- Donc, en réduisant un pb de NPC à un pb Q qui nous intéresse, on réduit implicitement tout pb de NPC à Q !
- Preuve :
 - Q' est NP-complet \rightarrow tout pb $Q'' \in NP$ est t.q. $Q'' \leq_P Q'$.
 - Par hyp, $Q' \leq_P Q$, et donc par transitivité de \leq_P on sait $Q'' \leq_P Q$.
 - Ceci montre par déf que Q est NP-difficile
 - Si en plus il est NP, alors par déf, il est NP-complet

Ceci n'est pas un exercice !

Exercice

CYCLE-HAMILTONIEN : { $\langle G = (S, A) \rangle$: un graphe quelconque, G a-t-il un cycle hamiltonien, c'est à dire un cycle qui passe par tous les sommets une fois et une seule ?}.

- Ce problème est NP-complet.
- Utilisez cette information pour montrer que TSP est NP-complet.
- Rappel : **VOYAGEUR DE COMMERCE (TSP)** : { $\langle G = (S, A) \rangle$: n villes et une distance $d(i;j)$ entre toute paire de villes i et j . Existe-t-il un chemin de longueur totale $\leq k$ qui visite chaque ville ? }

Pour faire ça,
Donnée

Exercice - suite

un graphe $G = (S, A)$ dans lequel nous voulons tester l'existence d'un cycle hamiltoinen,

To do

créer une instance D de TSP

- ▶ Quelles villes choisir pour le TSP ?
- ▶ Comment fixer la distance $d(i;j)$ entre deux villes i et j ?

Montrons
que

G admet un cycle hamiltonien
 \Leftrightarrow
D admet une tournée de coût $\leq n$

\Rightarrow

\Leftarrow

Exercice - suite

Exercice



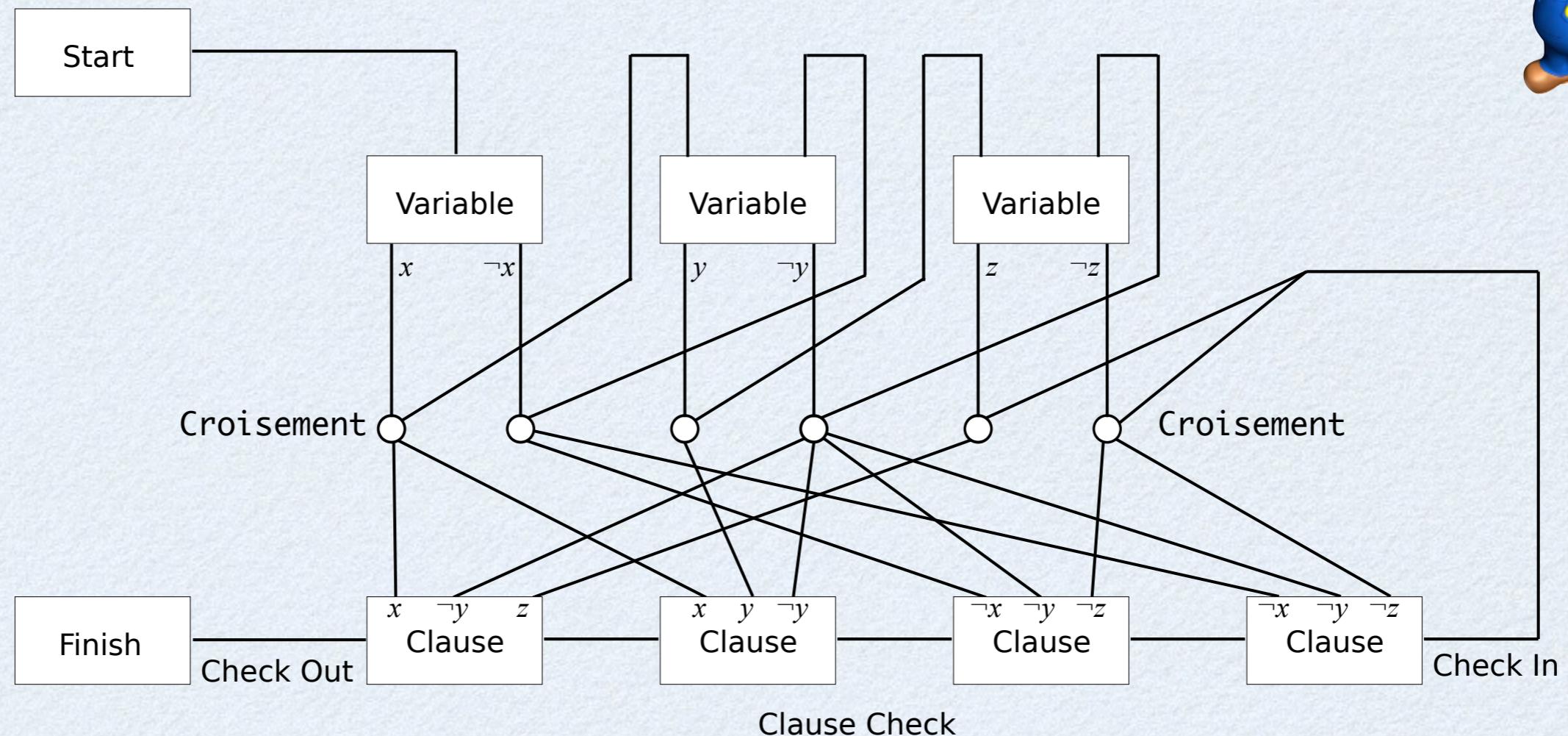
Super Mario Bros est NP-difficile !

«Pour un jeu de tableaux, quelle est le meilleur trajet que l'on puisse faire (speed run) ?»

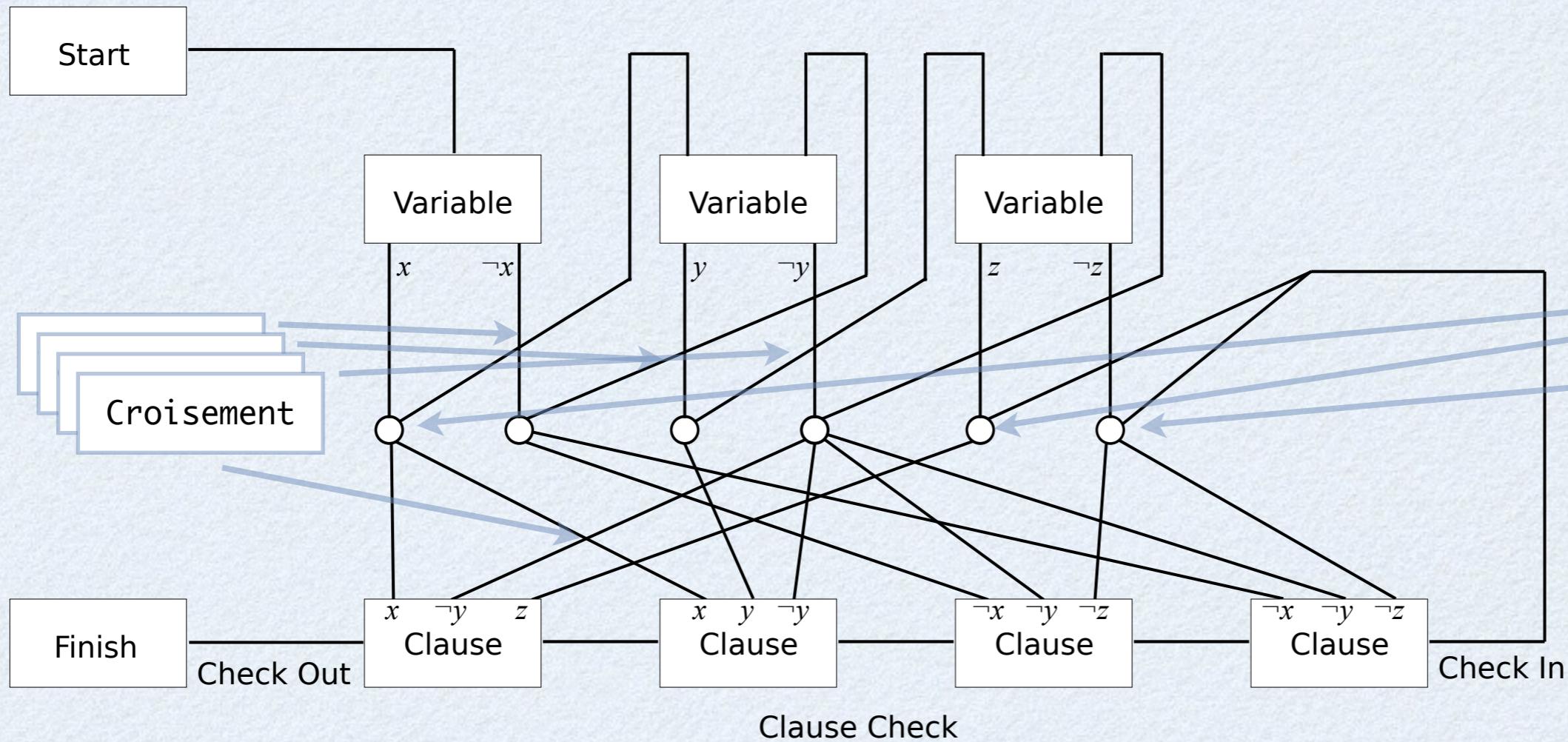
Problème de décision : étant donné un *tableau* (stage) existe-t-il une solution pour joindre l'arrivée depuis le départ ?

Motivation : s'il est difficile de répondre à cette question, alors il est encore plus difficile de répondre à l'optimalité d'un trajet.

- Réduction depuis le problème **3-SAT** en utilisant ce schéma de parcours et des gadgets :



- Mario commence à l'écran Start, et passe ensuite dans les écrans correspondant aux «Variables».
- Chaque écran «Variable» force le joueur à faire un choix exclusif entre “true” (x) et “faux” ($\neg x$) pour une variable de la formule.
- Chacun de ces choix permet au joueur de suivre un chemin jusqu’aux écrans des «Clauses» contenant ce littéral (x ou $\neg x$).
- Ces chemins peuvent se croiser mais les écrans «croisement» empêchent le joueur de changer de chemin (passer d'un écran variable directement à un autre de même type)



- En visitant un écran Clause, le joueur «débloque» la clause, ce qui permettra plus tard le chemin de «Check in» à «Check out»
- En arrivant d'un écran «Variable» dans un écran «Clause» le joueur ne peut que remonter ensuite vers l'écran «croisement» dont il vient
- Après avoir traversé **tous** les écrans «Variable» et débloqué des Clauses, il peut suivre le long chemin de vérification «check» pour atteindre l'écran de fin («Finish»).
- Le joueur peut traverser tout le chemin de vérification si et seulement si chaque clause a été déverrouillée par un littéral (au moins).

D'après ce schéma, il suffit de concevoir les écrans Start, Variable, Clause, Finish, et Croisement pour montrer la **NP-difficulté** de **Super Mario Bros**

Spécifications du jeu :

Exercice - suite

- Le personnage change d'état s'il prend un champignon :

- état **Mario** : il fait un étage de «haut»



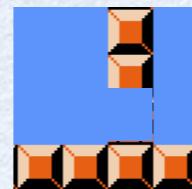
- état **Super Mario** : il faut deux étages de haut



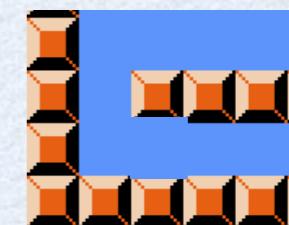
- Les actions possibles pour (Super) Mario :

- avancer**,

- ramper** sous un obstacle fin (quand SuperMario), c-à-d passer dans un trou d'un de haut, et un de large comme



mais pas comme



- sauter** :

- pour franchir un obstacle (ou un trou)

- en sautant sous une brique **Mario** peut en faire sortir un objet caché, et **SuperMario** peut la faire disparaître

- en sautant sur une tortue rouge, il la propulse ce qui a pour effet de détruire les briques contre lesquelles elle rebondit



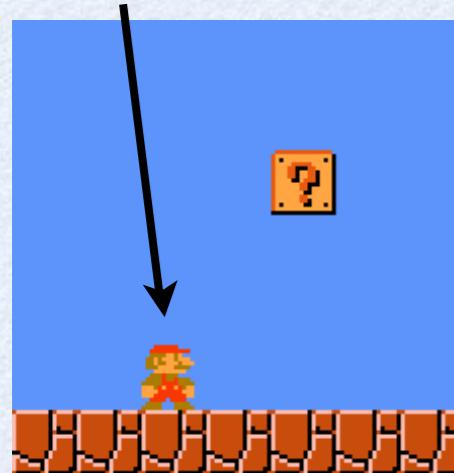
Exercice - suite



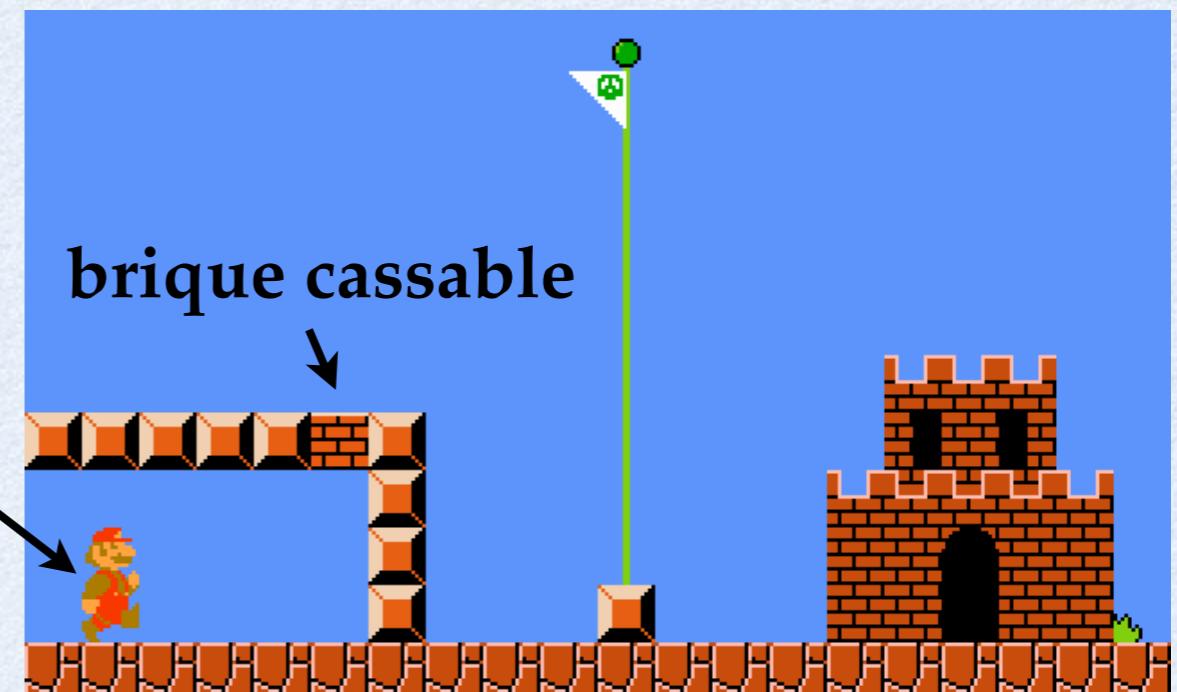
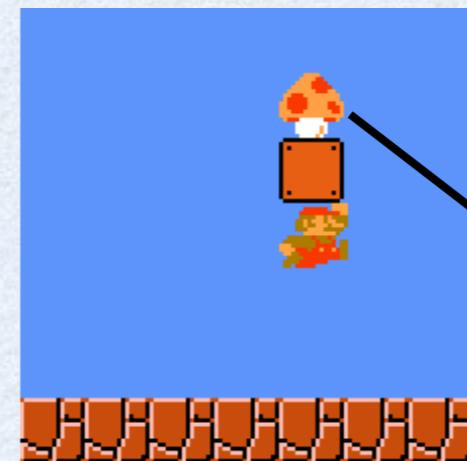
- Les écrans Start et Finish :

initialement :

état **Mario**



Start : la brique donne
accès à un champignon



état **SuperMario**

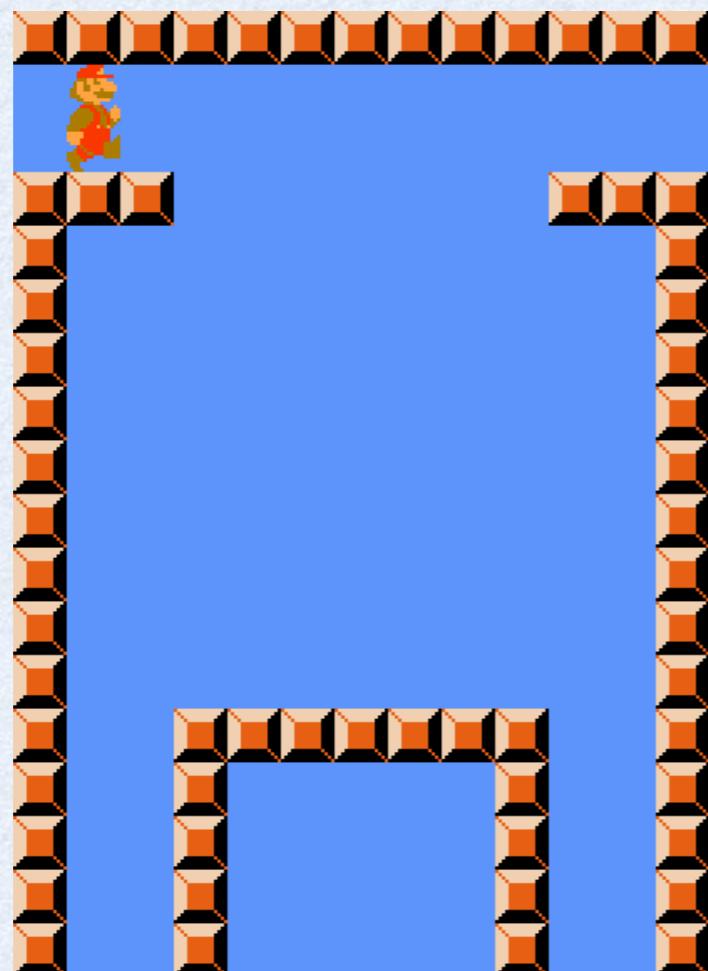
Finish

- Pour pouvoir finir le tableau, le personnage doit être **SuperMario** (**Mario** pas assez grand pour sauter un mur de 4 de haut) -> le joueur est forcé de prendre le champignon dans l'écran Start

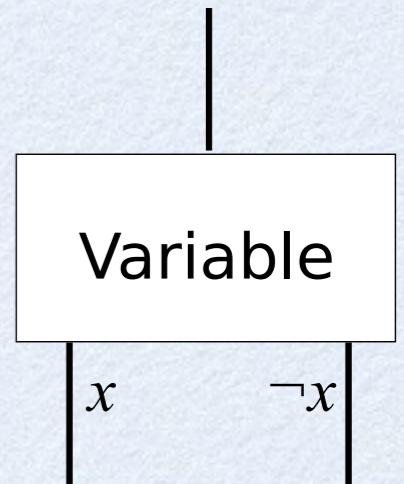
Exercice - suite



- Chaque écran de «Variable» est conçu ainsi:



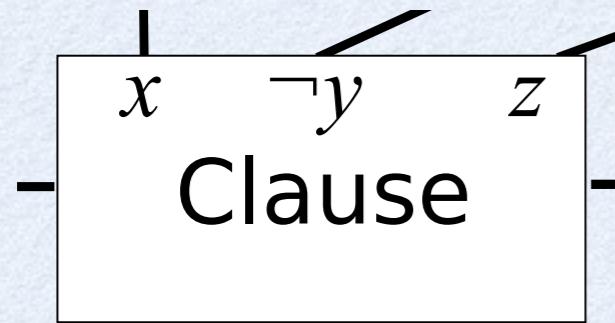
écran
«Variable»



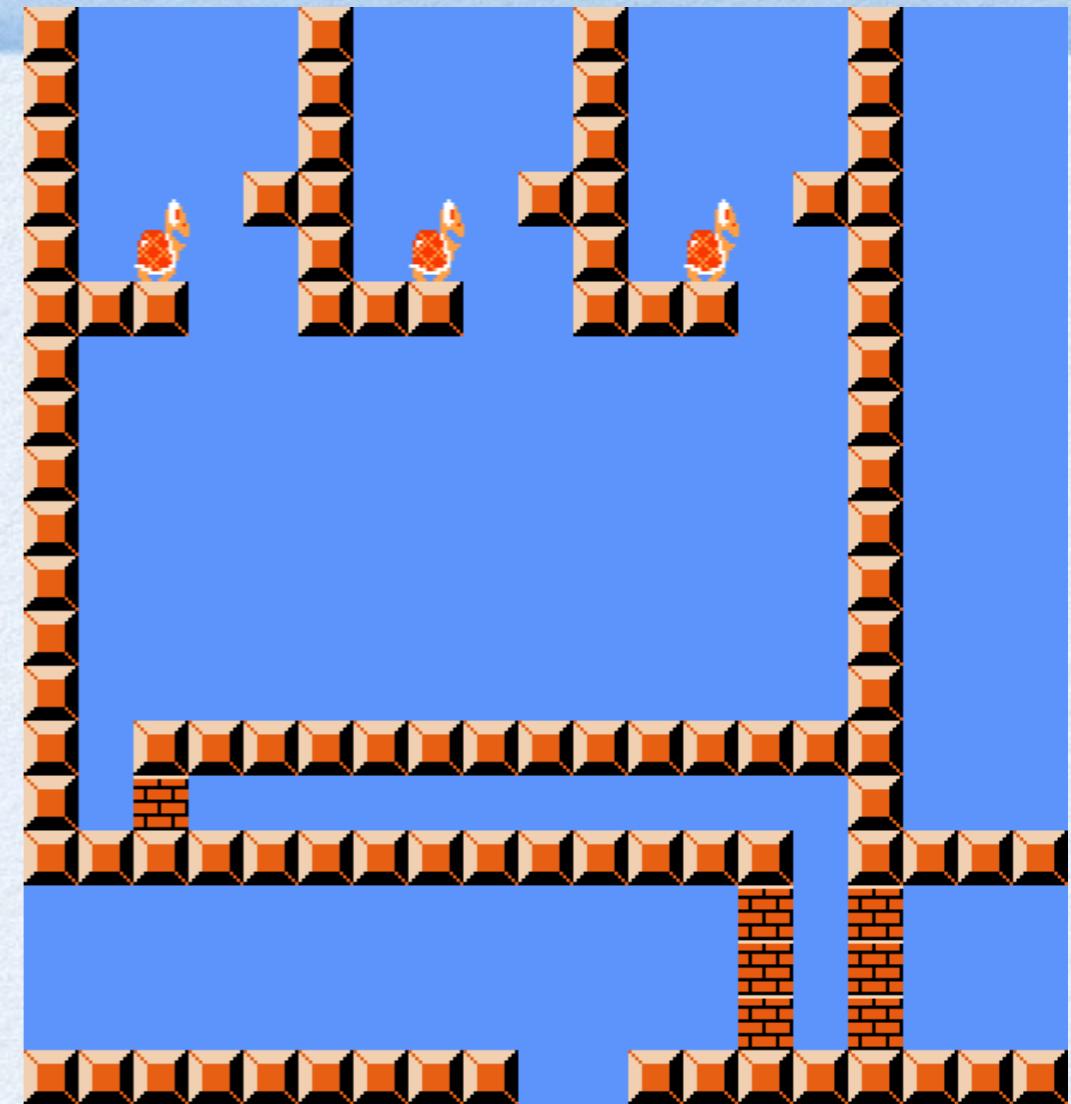
- Expliquez comment cela permet de coder le choix pour la variable entre “true” (x) et “faux” ($\neg x$)

Exercice - suite

- Chaque écran de «Clause» est conçu ainsi:



écran
«Clause»

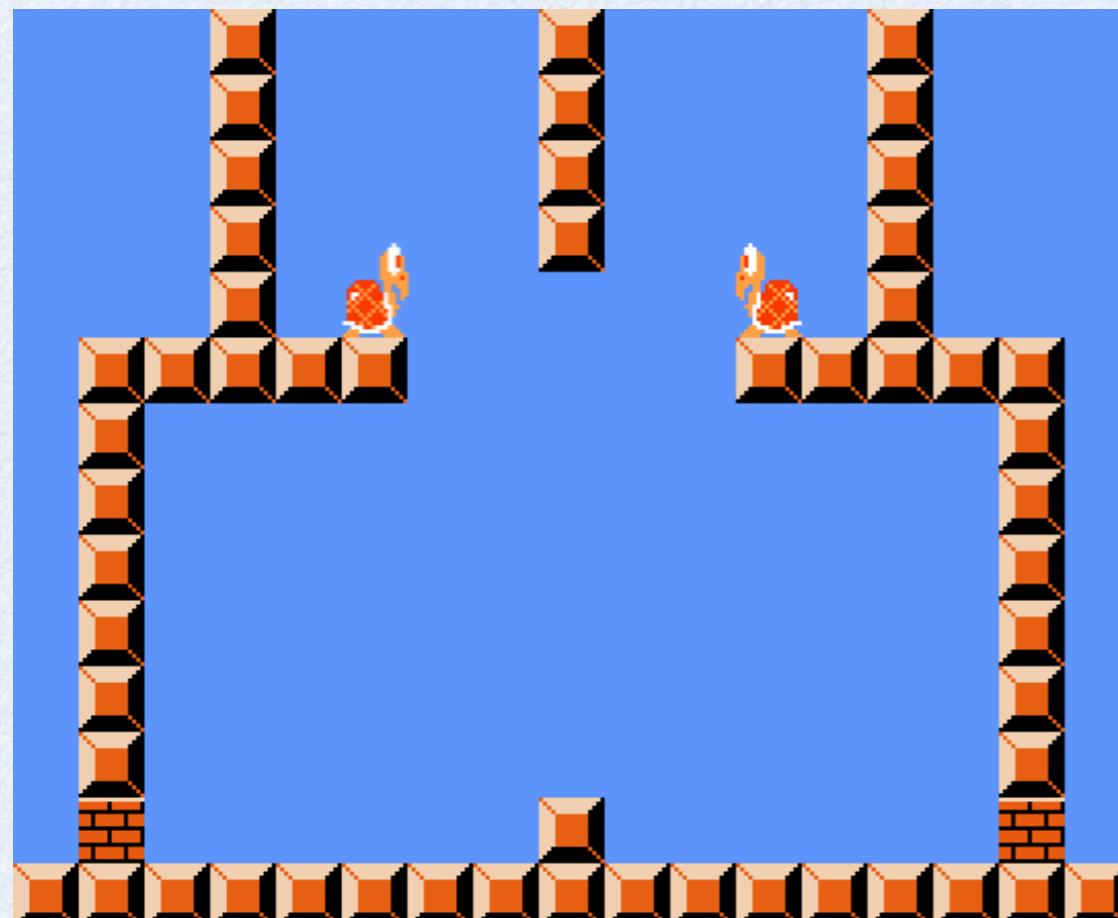


- Expliquez pourquoi il y a trois entrées par le haut
- Expliquez comment le personnage peut «débloquer» le passage droite-gauche en bas de l'écran, et ce indépendamment de l'entrée du haut par lequel le personnage arrive.
- Quelle sortie pour le personnage quand il vient du haut ? quand il vient du bas ?

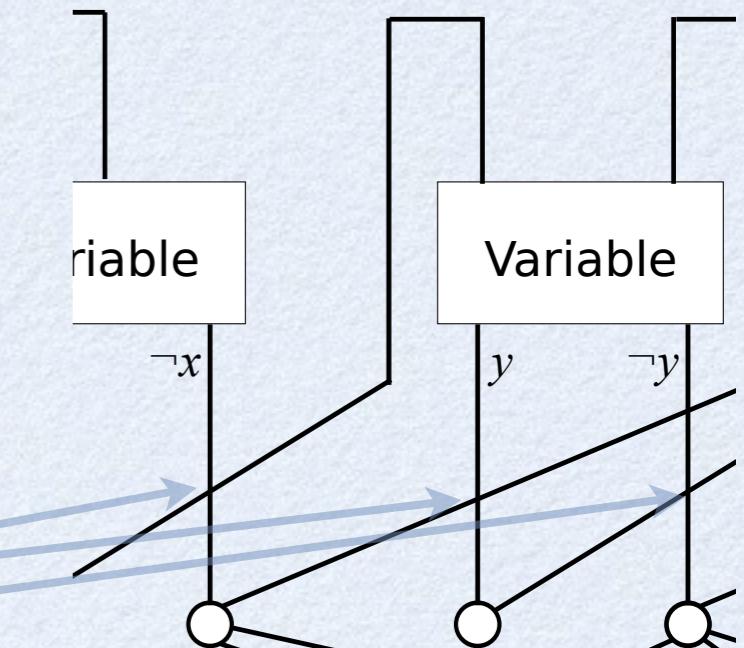
Exercice - suite



- Chaque écran de «Croisement» a 4 entrées :



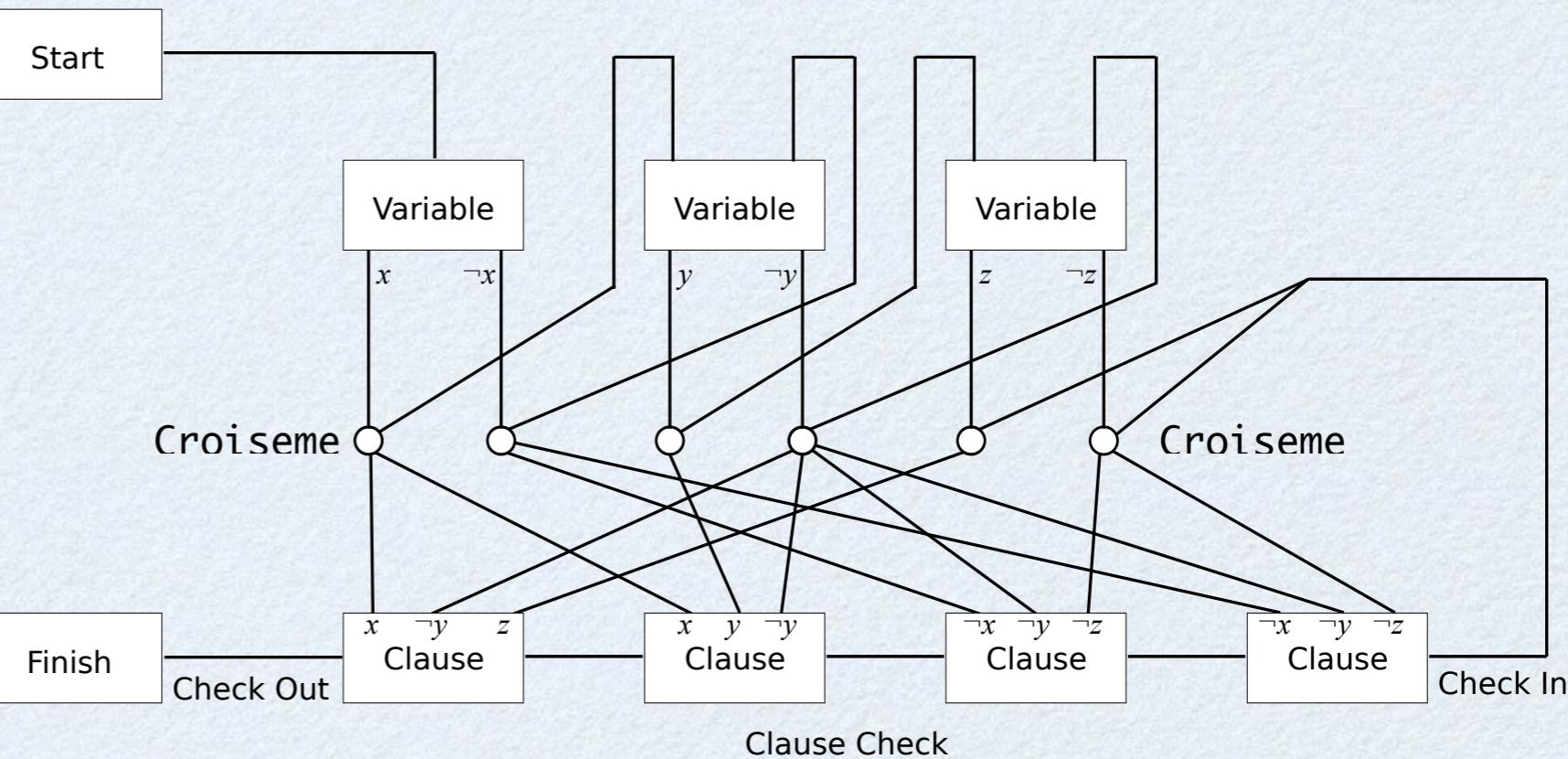
écran
«Croisement»



NB : quand une brique en bas est brisée, SuperMario peut quitter le tableau par le bas en **rampant** sous le mur restant car il ne fait que un carré de large

- S'il arrive par l'entrée en haut à gauche et tape dans la carapace, quelle sortie libère celle-ci en tombant ?
- Expliquez pourquoi en revenant il ne peut ressortir que par la même entrée

NP-DIFFICULTÉ



- Montrez comment l'enchaînement des écrans du tableau oblige à passer par chaque écran de variable.
- Que peut-on en conclure pour un chemin de Start à Finish ?
- A quelle condition le chemin de «Check in» à «Check out sera-t-il ouvert ?
- Que peut-on en déduire pour un chemin de Start à Finish ?
- Cela montre que tout chemin correct pour résoudre SuperMario Bros satisfait la formule de 3-SAT donnée en entrée
- Inversement, montrez comment une instantiation des variables satisfaisant la formule permet d'obtenir un chemin menant de Start à Finish dans le jeu SuperMario Bros.
- Qu'en déduire pour le problème SuperMario Bros ?

ET MÊME PLUS !

- Supposons qu'il y ait une solution et considérons un chemin solution de l'écran Start à l'écran Finish.
- Ce chemin a besoin de passer au plus une fois par chaque tortue, donc la solution est de taille polynomiale en la taille de la donnée (nb d'écrans proportionnel au nombre de tortues). Vérifier une solution prend donc un temps polynomial, donc le **SuperMario Bros** est **NP**.
- Mais nous savons aussi qu'il est **NP-difficile** (par réduction de 3-SAT)....

Question : qu'est-ce que cela implique ?

pas comme Sonic, pffff !



III.3 LES CLASSES NP ET NPC

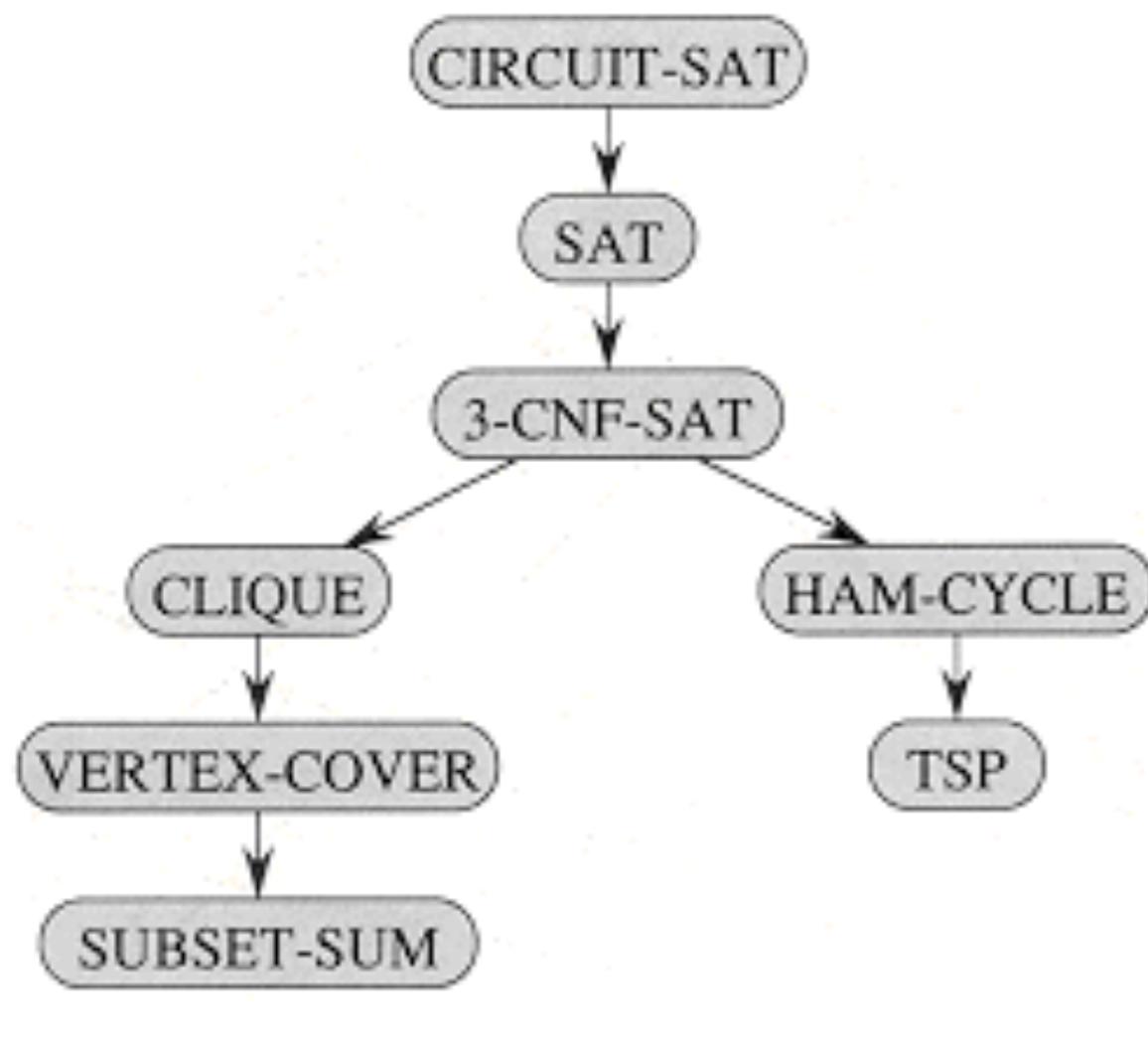
Pour résumer :

Pour montrer qu'un pb Q est NP-complet on doit :

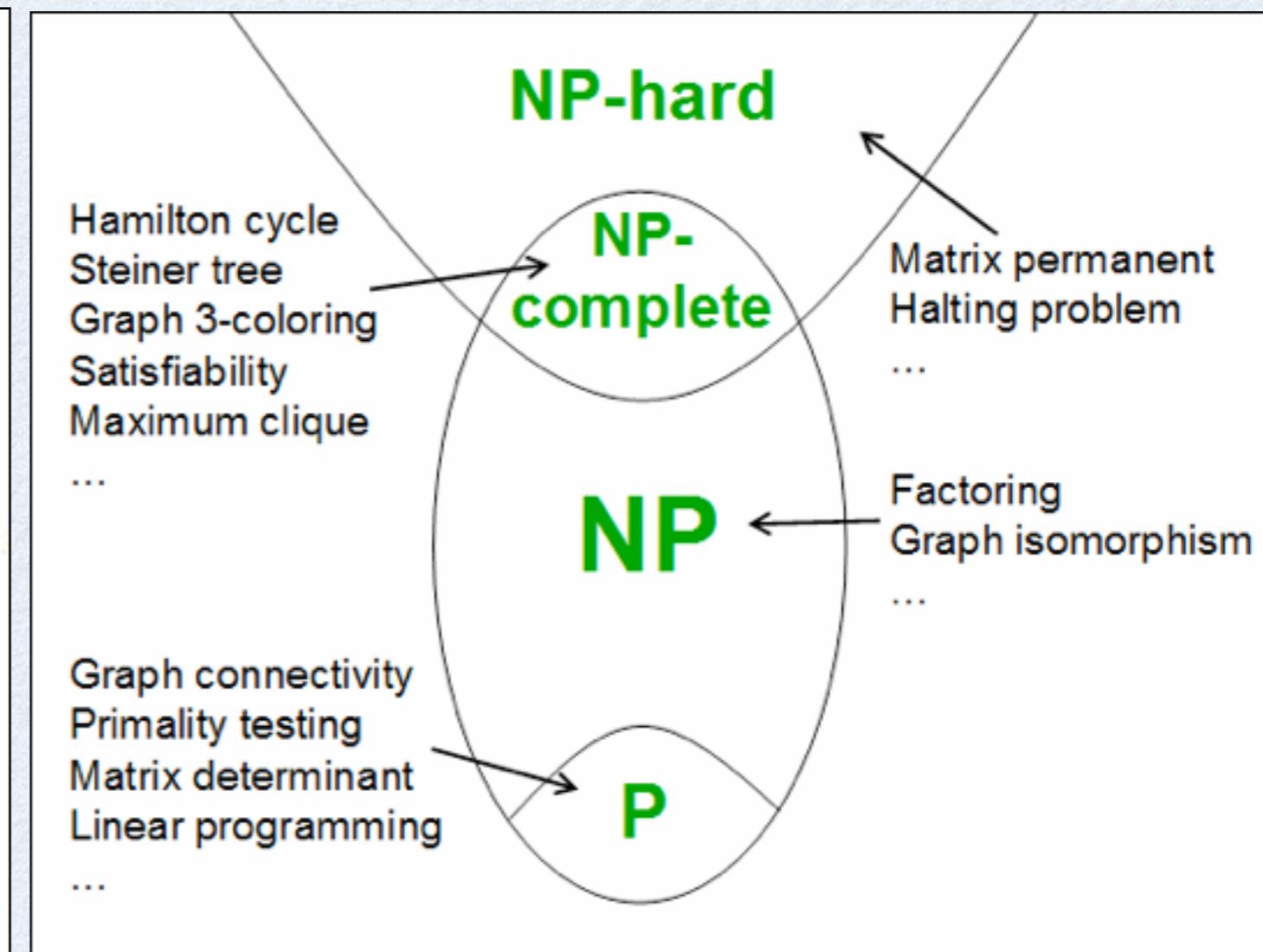
1. prouver que Q est dans NP, c-à-d qu'il existe un algorithme de validation polynomial pour une instance x et un certificat y
2. choisir un pb Q' de NPC
3. décrire un algorithme **polynomial** capable de calculer une transformation f faisant correspondre à toute instance x' de Q' en une instance x de Q
4. montrer que x est une instance positive pour Q
si et seulement si x' est une instance positive pour Q'



Réductions connues entre problèmes NP-complets :



Classes de problèmes et exemples :



Rappel : Ceci concerne des problèmes de décision (réponse oui / non)



Rmq à plusieurs millions de \$:

- la gloire éternelle attend la personne qui
 1. prouve qu'il n'existe pas d'algorithme polynomial pour un pb de NPC
 2. ou trouve un algorithme polynomial pour un pb de NPC