#Version : calculabilite.tex 13 nov 24 18:34:18 compiled November 13, 2024 6:34pm +01:00

Calculabilité & Complexité Algorithmique

Nicolas Bedon

Université de Rouen

SAT : un problème NP-complet

Expressions booléennes :

- variables $x_1, \ldots, x_n$ à valeur dans $\mathbb{B}$;
- connecteurs propositionnels $\neg, \wedge, \vee$.

Forme normale conjonctive (FNC)

- ψ est de la forme $\psi \equiv \wedge_{i \in I} \vee_{j \in J} \psi_{i,j}$
- chaque $\psi_{i,j}$ est de la forme
  - x_k
  - $\neg x_k$

Chaque expression booléenne admet une FNC

Satisfaisabilité de ψ :

Existence d'une valuation de $x_1, \ldots, x_n$ qui rende ψ vraie.

Theorem

La satisfaisabilité SAT des expressions booléennes est un problème NP-complet.

SAT : un problème NP-complet : preuve

$SAT(\psi) \in NP$ car :

- Vérifier qu'une valuation pour $x_1, \ldots, x_n$ permette de satisfaire ψ est linéaire en la taille de ψ ;
- Pour trouver une valuation, il faut en choisir une de manière non-déterministe et vérifier qu'elle satisfait ψ.

Réduction

- une réduction, en temps polynomial, de tout problème NP vers *SAT*
- idée : coder les exécutions acceptantes de toute MT non déterministe $M \in NTIME(p(n))$ par une expression booléenne ψ_M
  - le temps d'exécution $p(n) > 0$ de M est un polynôme

$$T = \{0, \ldots, p(n)\}$$

  - la taille nécessaire de bande de M est donc au plus de $p(n)$ cases

$$E = \{0, \ldots, p(n) - 1\}$$

SAT : un problème NP-complet : preuve de complétude

Variables de ψ_M

Pour tout $a \in \Gamma$, $i \in T$, $j \in E$,

$c_{a,i,j}$: au temps i, la lettre de la bande à la position j est un a ;

$p_{i,j}$: au temps i, la tête de lecture est à la position j ;

$e_{q,i}$: au temps i, M est dans l'état q.

Nombre de variables : polynomial !

$$|\Gamma||T||E| + |T||E| + |Q||T| =$$
$$|\Gamma|(p(n)+1)p(n) + (p(n)+1)p(n) + |Q|(p(n)+1)$$

SAT : un problème NP-complet : preuve de complétude

Encodage des propriétés de M

- À tout moment, chaque case de la bande contient exactement un caractère
  - au plus un

$$\bigwedge_{\substack{i\in T\\ j\in E\\ a\in\Gamma}} \left(c_{a,i,j} \rightarrow \bigwedge_{\substack{b\in\Gamma\\ b\neq a}} \neg c_{b,i,j}\right)$$

  - au moins un

$$\bigwedge_{\substack{i\in T\\ j\in E}} \bigvee_{a\in\Gamma} c_{a,i,j}$$

- À tout moment la tête est exactement sur une position de la bande
  - au plus une

$$\bigwedge_{\substack{i\in T\\ j\in E}} \left(p_{i,j} \rightarrow \bigwedge_{\substack{j'\in E\\ j'\neq j}} \neg p_{i,j'}\right)$$

  - au moins une

$$\bigwedge_{i\in T} \bigvee_{j\in E} p_{i,j}$$

SAT : un problème NP-complet : preuve de complétude - Encodage des propriétés de M

- À tout moment M est exactement dans un état
  - au plus un

$$\bigwedge_{\substack{i \in T \\ q \in Q}} (e_{q,i} \rightarrow \bigwedge_{\substack{q' \in Q \\ q' \neq q}} \neg e_{q',i})$$

  - au moins un

$$\bigwedge_{i \in T} \bigvee_{q \in Q} e_{q,i}$$

- Au temps 0, M est dans son état initial q_0

$$e_{q_0,0}$$

- Si on atteint un état final on y reste jusqu'à écoulement complet du temps

$$\bigwedge_{i \in T} \bigvee_{q \in F} e_{q,i} \rightarrow \bigwedge_{\substack{j \in T \\ i < j}} e_{q,j}$$

- Au temps $p(n)$, M est dans un état final

$$\bigvee_{q \in F} e_{q,p(n)}$$

SAT : un problème NP-complet : preuve de complétude - Encodage des propriétés de M

- Au temps 0, la tête est en début de bande

$$p_{0,0}$$

- Au temps 0, le contenu de la bande est $a_0 a_1 \dots a_k \# \dots \#$

$$(\wedge_{r\in\{0,\dots,k\}} c_{a_r,0,r}) \wedge (\wedge_{r\in E-\{0,\dots,k\}} c_{\#,0,r})$$

- Entre les temps t et $t+1$, seul le caractère sous la tête au temps t peut avoir changé

$$\wedge_{\substack{t\in T-\{p(n)\},j\in E,a\in\Gamma \\ j'\in T,j'\neq j}} (p_{t,j} \wedge c_{a,t,j'}) \rightarrow c_{a,t+1,j'}$$

- Et son changement est en cohérence avec δ

$$\wedge_{\substack{t\in T-\{p(n)\},j\in E,a\in\Gamma \\ b\in\Gamma,q\in Q}} (c_{a,t,j} \wedge p_{t,j} \wedge e_{q,t}) \rightarrow$$

$$\vee_{(q,a,q',b,D)\in\delta} (c_{b,t+1,j} \wedge p_{t+1,j+1} \wedge e_{q',t+1})$$

$$\vee_{\substack{(q,a,q',b,G)\in\delta \\ j\neq 0}} (c_{b,t+1,j} \wedge p_{t+1,j-1} \wedge e_{q',t+1})$$

SAT : un problème NP-complet : preuve de complétude - Encodage des propriétés de M

- ψ_M est la conjonction de toutes ces formules ;
- ψ_M a une taille polynomiale $p'(|M|)$;
- ψ_M se calcule en temps polynomial $p''(|M|)$;
- le nombre de solutions de ψ_M est égal au nombre de chemins acceptants de M ;
- chaque solution de ψ_M donne une sortie de M.

donc tout problème $\in NP$ donné par une MT non déterministe M se réduit en temps polynomial à SAT.

SAT : un problème NP-complet

- SAT est *NP*-complet
- Que se passe t-il quand on contraint les formules ?
  - C-SAT : satisfaisabilité des formules en FNC
    - C-SAT est *NP*-complet
  - 3-SAT : satisfaisabilité des formules en FNC dans lesquelles chaque clause a exactement 3 littéraux
    - littéral : disjonction d'atomes ou de négations d'atomes
    - Exemple : $(x \vee \neg x \vee y) \wedge (\neg y \vee z \vee \neg x) \in$ 3-SAT
    - 3-SAT est *NP*-complet

C-SAT : un problème NP-complet

C-SAT(ψ) $\in$ *NP* car :

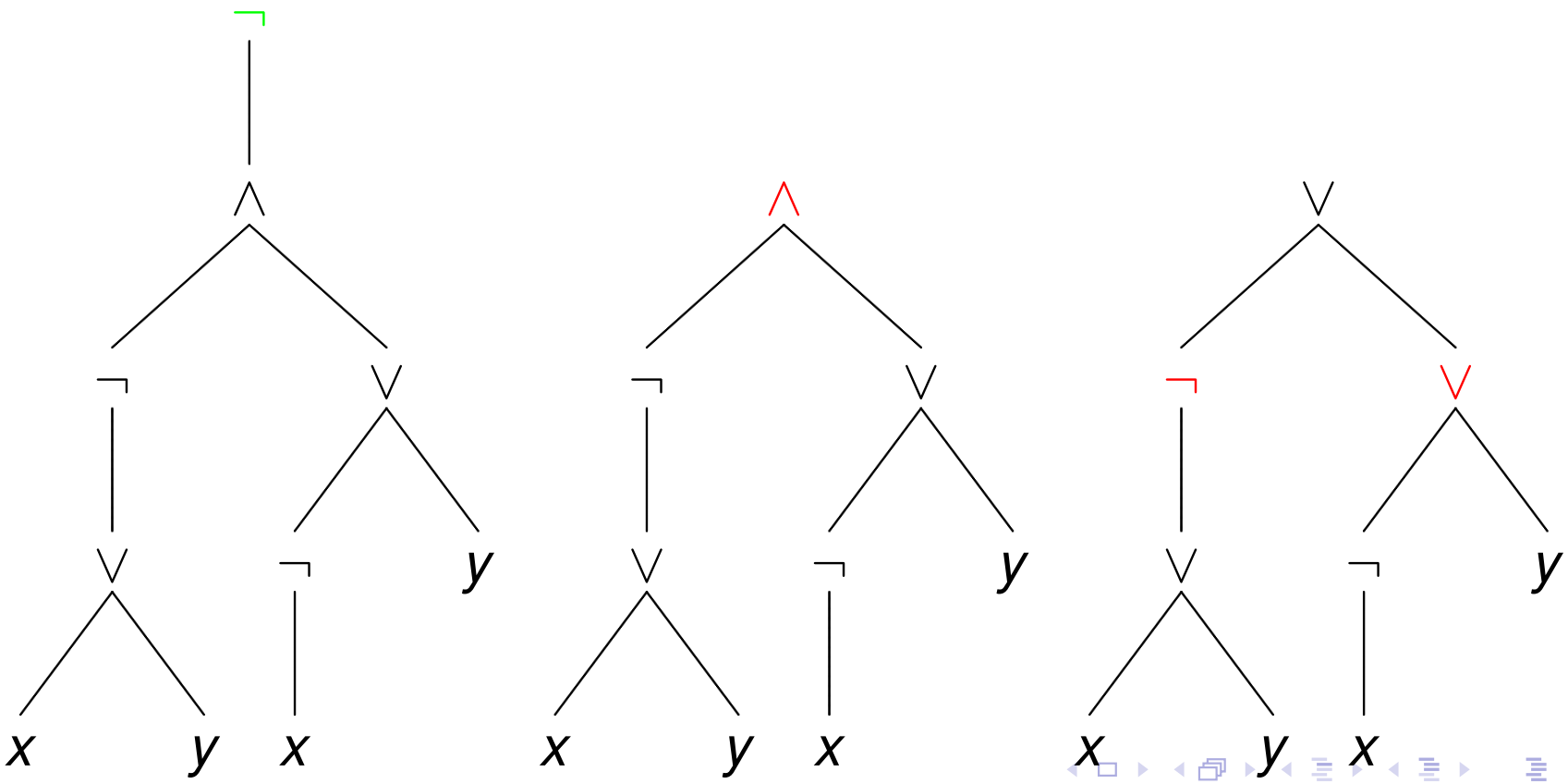
- C-SAT est un cas particulier de SAT

Réduction

- Une réduction, en temps polynomial, de SAT vers C-SAT
- Comme tout problème de NP se réduit à SAT en temps polynomial, tout problème de NP se réduit à C-SAT en temps polynomial
- Schéma : ψ une formule
  1. ψ' : descendre les négations de ψ sur les atomes
  2. calculer une formule ψ'' dont les solutions impliquent celles de ψ'
- Les deux étapes en temps polynomial sur la taille de ψ

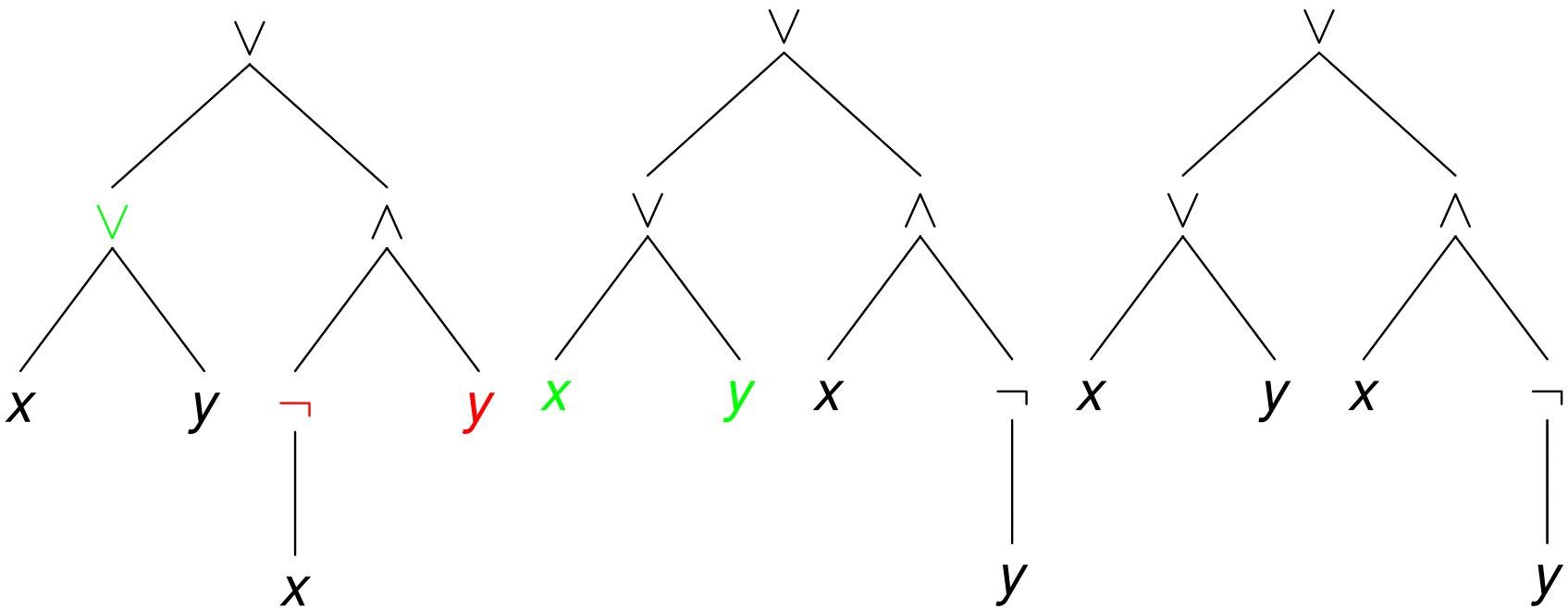
C-SAT : un problème NP-complet - Preuve de complétude - Descente des négations

- Parcours de l'arbre syntaxique de la formule
- On mémorise les négations
  - Nœud rouge : négation à mettre en œuvre
  - Nœud vert : pas de négation à mettre en œuvre
- On applique les simplifications et transformations



C-SAT : un problème NP-complet - Preuve de complétude - Descente des négations

- Parcours de l'arbre syntaxique de la formule
- On mémorise les négations
  - Nœud rouge : négation à mettre en œuvre
  - Nœud vert : pas de négation à mettre en œuvre
- On applique les simplifications et transformations



C-SAT : un problème NP-complet - Preuve de complétude - Étape 2

Construction d'un ensemble de clauses $C_1, \ldots, C_n$ tel que

- Si $S \models \phi'$ alors il existe $S' \supseteq S$ tel que $S' \models \wedge_{i=1}^{n} C_i$
- Si $S' \models \wedge_{i=1}^{n} C_i$ il existe $S \subseteq S'$ tel que $S \models \phi'$

Par induction sur ϕ'

- $C_x = \{x\}$, $C_{\neg x} = \{\neg x\}$
- HR : $C_{\phi'_i} = \{C_{\phi'_i,1}, \ldots, C_{\phi'_i,n_i}\}$
- $C_{\phi'_1 \wedge \phi'_2} = \{C_{\phi'_1,1}, \ldots, C_{\phi'_1,n_1}, C_{\phi'_2,1}, \ldots, C_{\phi'_2,n_2}\}$
  - On suppose qu'il n'y a pas de variable inutile dans les clauses
- $C_{\phi'_1 \vee \phi'_2} = \{C_{\phi'_1,1} \vee y, \ldots, C_{\phi'_1,n_1} \vee y, C_{\phi'_2,1} \vee \neg y, \ldots, C_{\phi'_2,n_2} \vee \neg y\}$
  - y est une nouvelle variable (quelconque)
  - Si $y = 1$ est dans S' pour $S' \models \wedge_{\substack{i \in \{1,2\} \\ j \in \{1,\ldots,n_j\}}} C_{\phi'_i,j}$, alors $S' = \{y = 1\} \cup S''$ avec $S'' \models \wedge_{i=1}^{n_2} C_{\phi'_2,i}$ et on applique l'HR
  - Réciproquement, si $S \models \phi'_1 \vee \phi'_2$, alors wlog. $S \models \phi'_1$ et par HR il existe $S' \models \wedge_{j=1}^{n_i} C_{\phi'_i,j}$, et $S' \cup \{y = 0\} \models \wedge_{\substack{i \in \{1,2\} \\ j \in \{1,\ldots,n_j\}}} C_{\phi'_i,j}$

Une solution de $\wedge_{i=1}^{n} C_i$ donne une solution pour ϕ' en temps linéaire.

3-SAT : un problème NP-complet

3-SAT(ψ) $\in$ *NP* car :

- 3-SAT est un cas particulier de C-SAT

Réduction

- Toute formule en CNF se transforme en une formule CNF à 3 littéraux max. par clause :
  - $\psi = \wedge_{i=1}^{n} C_i$
  - $C_i = l_1 \vee \cdots \vee l_k$ avec $k > 3$ se transforme en

$$(l_1 \vee l_2 \vee y_1) \wedge (l_3 \vee \neg y_1 \vee y_2) \wedge \ldots \wedge (l_{k-2} \vee \neg y_{k-4} \vee y_{k-3}) \wedge (l_{k-1} \vee l_k \vee \neg y_{k-3})$$

  - les y_j sont de nouvelles variables
  - si C_i est vraie alors on trouve facilement une valuation des y_j pour rendre C_i' vraie
    - au moins un des l_j est vrai : on prend j l'indice du plus petit, j' celui du plus grand
    - on fixe tous les y_s, $s \leq j-2$, à vrai
    - on fixe tous les y_s, $j'-1 \leq s$, à faux
    - si $j \neq j'$, on fixe tous les y_s, $j-1 \leq s \leq j'-2$ ou tous vrais, ou tous faux (peu importe)

Voir diapositive suivante

3-SAT : un problème NP-complet

Valuation des y_i pour rendre c'_i vraie

Au moins un des l_j est vrai : on prend

- j l'indice du plus petit,
- j' celui du plus grand

Choix de la valuation des y_i :

$$
\begin{aligned}
&(l_1 \vee l_2 \vee y_1)\\
\wedge&(l_3 \vee \neg y_1 \vee y_2)\\
&\dots\\
\wedge&(l_{j-1} \vee \neg y_{j-3} \vee y_{j-2})\\
\wedge&(l_j \vee \neg y_{j-2} \vee y_{j-1})\\
\wedge&(l_{j+1} \vee \neg y_{j-1} \vee y_j)\\
&\dots\\
\wedge&(l_{a-1} \vee \neg y_{a-3} \vee y_{a-2})\\
\wedge&(l_a \vee \neg y_{a-2} \vee y_{a-1})\\
\wedge&(l_{a+1} \vee \neg y_{a-1} \vee y_a)\\
&\dots\\
\wedge&(l_{b-1} \vee \neg y_{b-3} \vee y_{b-2})\\
\wedge&(l_b \vee \neg y_{b-2} \vee y_{b-1})\\
\wedge&(l_{b+1} \vee \neg y_{b-1} \vee y_b)\\
&\dots\\
\wedge&(l_{j'-1} \vee \neg y_{j'-3} \vee y_{j'-2})\\
\wedge&(l_{j'} \vee \neg y_{j'-2} \vee y_{j'-1})\\
\wedge&(l_{j'+1} \vee \neg y_{j'-1} \vee y_{j'})\\
&\dots\\
\wedge&(l_{k-2} \vee \neg y_{k-4} \vee y_{k-3})\\
\wedge&(l_{k-1} \vee l_k \vee \neg y_{k-3})
\end{aligned}
$$

3-SAT : un problème NP-complet

Valuation des y_i pour rendre c'_i vraie

Au moins un des l_j est vrai : on prend

- j l'indice du plus petit,
- j' celui du plus grand

Choix de la valuation des y_i :

1. on fixe tous les y_s, $s \leq j-2$, à vrai

$$
\begin{aligned}
&(l_1 \vee l_2 \vee y_1)\\
\wedge&(l_3 \vee \neg y_1 \vee y_2)\\
&\dots\\
\wedge&(l_{j-1} \vee \neg y_{j-3} \vee y_{j-2})\\
\wedge&(l_j \vee \neg y_{j-2} \vee y_{j-1})\\
\wedge&(l_{j+1} \vee \neg y_{j-1} \vee y_j)\\
&\dots\\
\wedge&(l_{a-1} \vee \neg y_{a-3} \vee y_{a-2})\\
\wedge&(l_a \vee \neg y_{a-2} \vee y_{a-1})\\
\wedge&(l_{a+1} \vee \neg y_{a-1} \vee y_a)\\
&\dots\\
\wedge&(l_{b-1} \vee \neg y_{b-3} \vee y_{b-2})\\
\wedge&(l_b \vee \neg y_{b-2} \vee y_{b-1})\\
\wedge&(l_{b+1} \vee \neg y_{b-1} \vee y_b)\\
&\dots\\
\wedge&(l_{j'-1} \vee \neg y_{j'-3} \vee y_{j'-2})\\
\wedge&(l_{j'} \vee \neg y_{j'-2} \vee y_{j'-1})\\
\wedge&(l_{j'+1} \vee \neg y_{j'-1} \vee y_{j'})\\
&\dots\\
\wedge&(l_{k-2} \vee \neg y_{k-4} \vee y_{k-3})\\
\wedge&(l_{k-1} \vee l_k \vee \neg y_{k-3})
\end{aligned}
$$

3-SAT : un problème NP-complet

Valuation des y_i pour rendre c'_i vraie

Au moins un des l_j est vrai : on prend

- j l'indice du plus petit,
- j' celui du plus grand

Choix de la valuation des y_i :

1. on fixe tous les y_s, $s \leq j-2$, à vrai
2. on fixe tous les y_s, $j'-1 \leq s$, à faux

$$
\begin{aligned}
&(l_1 \vee l_2 \vee y_1)\\
&\wedge(l_3 \vee \neg y_1 \vee y_2)\\
&\quad\ldots\\
&\wedge(l_{j-1} \vee \neg y_{j-3} \vee y_{j-2})\\
&\wedge(l_j \vee \neg y_{j-2} \vee y_{j-1})\\
&\wedge(l_{j+1} \vee \neg y_{j-1} \vee y_j)\\
&\quad\ldots\\
&\wedge(l_{a-1} \vee \neg y_{a-3} \vee y_{a-2})\\
&\wedge(l_a \vee \neg y_{a-2} \vee y_{a-1})\\
&\wedge(l_{a+1} \vee \neg y_{a-1} \vee y_a)\\
&\quad\ldots\\
&\wedge(l_{b-1} \vee \neg y_{b-3} \vee y_{b-2})\\
&\wedge(l_b \vee \neg y_{b-2} \vee y_{b-1})\\
&\wedge(l_{b+1} \vee \neg y_{b-1} \vee y_b)\\
&\quad\ldots\\
&\wedge(l_{j'-1} \vee \neg y_{j'-3} \vee y_{j'-2})\\
&\wedge(l_{j'} \vee \neg y_{j'-2} \vee y_{j'-1})\\
&\wedge(l_{j'+1} \vee \neg y_{j'-1} \vee y_{j'})\\
&\quad\ldots\\
&\wedge(l_{k-2} \vee \neg y_{k-4} \vee y_{k-3})\\
&\wedge(l_{k-1} \vee l_k \vee \neg y_{k-3})
\end{aligned}
$$

3-SAT : un problème NP-complet

Valuation des y_i pour rendre c'_i vraie

Au moins un des l_j est vrai : on prend

- j l'indice du plus petit,
- j' celui du plus grand

Choix de la valuation des y_i :

1. on fixe tous les y_s, $s \leq j-2$, à vrai
2. on fixe tous les y_s, $j'-1 \leq s$, à faux
3. si $j \neq j'$, on fixe tous les y_s, $j-1 \leq s \leq j'-2$
   - ou tous vrai,
   - ou tous faux

   (peu importe, ici vrai)

$$
\begin{aligned}
&(l_1 \vee l_2 \vee y_1)\\
\wedge&(l_3 \vee \neg y_1 \vee y_2)\\
&\ldots\\
\wedge&(l_{j-1} \vee \neg y_{j-3} \vee y_{j-2})\\
\wedge&(l_j \vee \neg y_{j-2} \vee y_{j-1})\\
\wedge&(l_{j+1} \vee \neg y_{j-1} \vee y_j)\\
&\ldots\\
\wedge&(l_{a-1} \vee \neg y_{a-3} \vee y_{a-2})\\
\wedge&(l_a \vee \neg y_{a-2} \vee y_{a-1})\\
\wedge&(l_{a+1} \vee \neg y_{a-1} \vee y_a)\\
&\ldots\\
\wedge&(l_{b-1} \vee \neg y_{b-3} \vee y_{b-2})\\
\wedge&(l_b \vee \neg y_{b-2} \vee y_{b-1})\\
\wedge&(l_{b+1} \vee \neg y_{b-1} \vee y_b)\\
&\ldots\\
\wedge&(l_{j'-1} \vee \neg y_{j'-3} \vee y_{j'-2})\\
\wedge&(l_{j'} \vee \neg y_{j'-2} \vee y_{j'-1})\\
\wedge&(l_{j'+1} \vee \neg y_{j'-1} \vee y_{j'})\\
&\ldots\\
\wedge&(l_{k-2} \vee \neg y_{k-4} \vee y_{k-3})\\
\wedge&(l_{k-1} \vee l_k \vee \neg y_{k-3})
\end{aligned}
$$

3-SAT : un problème NP-complet

3-SAT(ψ) $\in$ *NP* car :

- 3-SAT est un cas particulier de C-SAT

Réduction

- Toute formule en CNF se transforme en une formule CNF à 3 littéraux max. par clause :
  - $\psi = \wedge_{i=1}^{n} C_i$
  - $C_i = l_1 \vee \cdots \vee l_k$ avec $k > 3$ se transforme en

$$(l_1 \vee l_2 \vee y_1) \wedge (l_3 \vee \neg y_1 \vee y_2) \wedge \ldots \wedge (l_{k-2} \vee \neg y_{k-4} \vee y_{k-3}) \wedge (l_{k-1} \vee l_k \vee \neg y_{k-3})$$

  - les y_j sont de nouvelles variables
  - si C_i est vraie alors on trouve facilement une valuation des y_j pour rendre C_i' vraie
    - au moins un des l_j est vrai : on prend j l'indice du plus petit, j' celui du plus grand
    - on fixe tous les y_s, $s \leq j-2$, à vrai
    - on fixe tous les y_s, $j'-1 \leq s$, à faux
    - si $j \neq j'$, on fixe tous les y_s, $j-1 \leq s \leq j'-2$ ou tous vrais, ou tous faux (peu importe)

    Voir diapositive précédente
  - réciproquement si C_i' est vraie alors tous les l_j ne peuvent être faux. Par l'absurde, supp. tous les l_j faux ; dans chaque triplet formant C_i', un des éléments au moins est vrai et ça n'est pas l_j, donc c'est nécessairement la variable y_s ou la négation $\neg y_r$. Si c'est $y_s = \texttt{true}$ alors nécessairement dans un des triplets quelque part à droite un des l_t sera vrai, contradiction. Si c'est $\neg y_r = \texttt{true}$, même raisonnement mais en regardant les autres triplets vers la gauche. La valuation pour rendre C_i vraie est immédiate.
- Cette transformation est en temps polynomial

Utilité de 3-SAT

- Montrer que des problèmes sont NP-complets

Vertex-cover : un problème NP-complet

Vertex-cover

- $G = (V, E)$ un graphe fini (non orienté)
- Vertex-cover de taille k : sous-ensemble $A \subseteq V$ tel que toute arête aie au moins une extrémité dans A
- k-Vertex-cover problem : existence de A de taille au plus k.

k-Vertex-cover $\in NP$ car :

- si A est donné, on vérifie en temps polynomial que c'est un Vertex-cover

Réduction

- Réduction de 3-SAT à Vertex-cover en temps polynomial
  - $\psi = \wedge_{i=1}^{n} \vee_{j=1}^{3} l_{i,j}$
  - Les nœuds du graphe sont les littéraux $l_{i,j}$
  - Tous les nœuds (deux à deux différents) du même littéral sont reliés
  - Tout littéral est relié à ses négations
- Vertex-cover de taille $2n \iff$ Solution de ψ

Vertex-cover : un problème NP-complet - Complétude

- Réduction de 3-SAT à Vertex-cover en temps polynomial
  - $\psi = \wedge_{i=1}^{n} \vee_{j=1}^{3} l_{i,j}$
  - Les nœuds du graphe sont les littéraux $l_{i,j}$
  - Tous les nœuds (deux à deux différents) du même littéral sont reliés
  - Tout littéral est relié à ses négations
- Vertex-cover de taille $2n \iff$ Solution de ψ
  - $\Rightarrow$
    - Exactement un nœud par clause n'est pas dans A
    - On fixe la valeur de vérité de ce littéral à VRAI
  - $\Leftarrow$
    - Au moins un littéral $l_{i,j}$ par clause i est vrai
    - On met tous les littéraux de la clause dans A, sauf $l_{i,j}$

Vertex-cover : un problème NP-complet - Exemple

Circuit Hamiltonien : un problème NP-complet

Circuit Hamiltonien : chemin

- passant exactement une fois par chaque nœud
- revenant à son point de départ

Deux versions : graphes dirigés (H_d) ou non (H_{nd})

$H_d \in NP$ car :

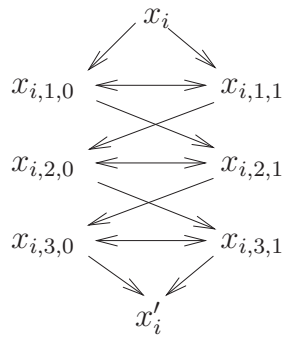
- Si c est donné, on vérifie en temps polynomial qu'il s'agit d'un circuit Hamiltonien

Réduction

- Réduction de 3-SAT à H_d
  À partir de $\psi = \wedge_{i=1}^{n} C_i$ on construit un graphe dirigé

Circuit Hamiltonien : un problème NP-complet - Complétude

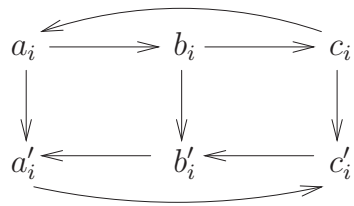
Pour chaque variable x_i un sous-graphe $G(x_i)$ de la forme



avec un « étage » de plus que d'apparitions de x_i ou $\neg x_i$ dans la formule
Un circuit Hamiltonien

- commence en haut
- termine en bas
- parcours chaque étage entièrement
- parcours tous les étages du haut vers le bas

Pour chaque clause C_i un sous-graphe $G(C_i)$ de la forme



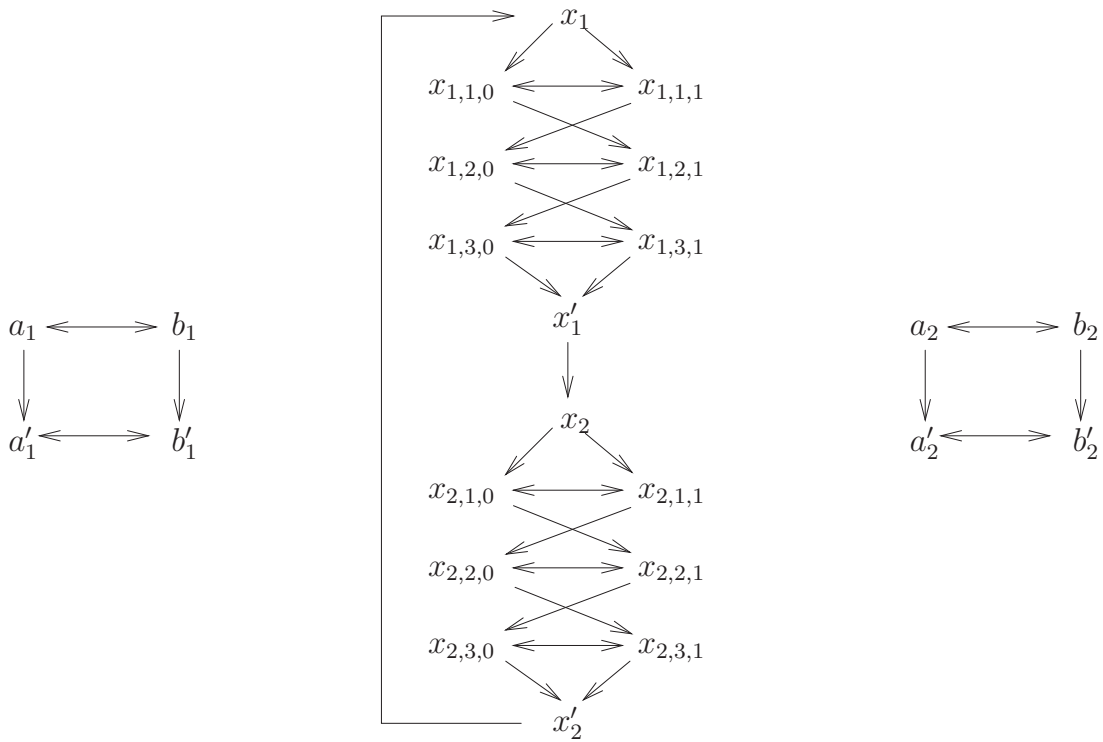
Un chemin hamiltonien rentrant en x sort en x'

Construction du graphe général (a un chemin Hamiltonien ssi ψ est satisfaisable)

- Connexion des sous-graphes des variables $x_1, x_2, \ldots, x_n$
  - en connectant x'_i avec $x_{i+1 \mod n}$
- Connexion des sous-graphes $C_i = \vee_{j=1}^{3} l_{i,j}$ avec les sous-graphes des variables
  - si $l_{i,1} = x_j$, trouver un nœud $x_{j,k,1}$ non utilisé et ajouter $x_{j,k,1} \to a_i$ et $a'_i \to x_{j,k+1,0}$
  - si $l_{i,1} = \neg x_j$, trouver un nœud $x_{j,k,0}$ non utilisé et ajouter $x_{j,k,0} \to a_i$ et $a'_i \to x_{j,k+1,1}$
  - pour $l_{i,2}$ et $l_{i,3}$, pareil en remplaçant a par respectivement b et c

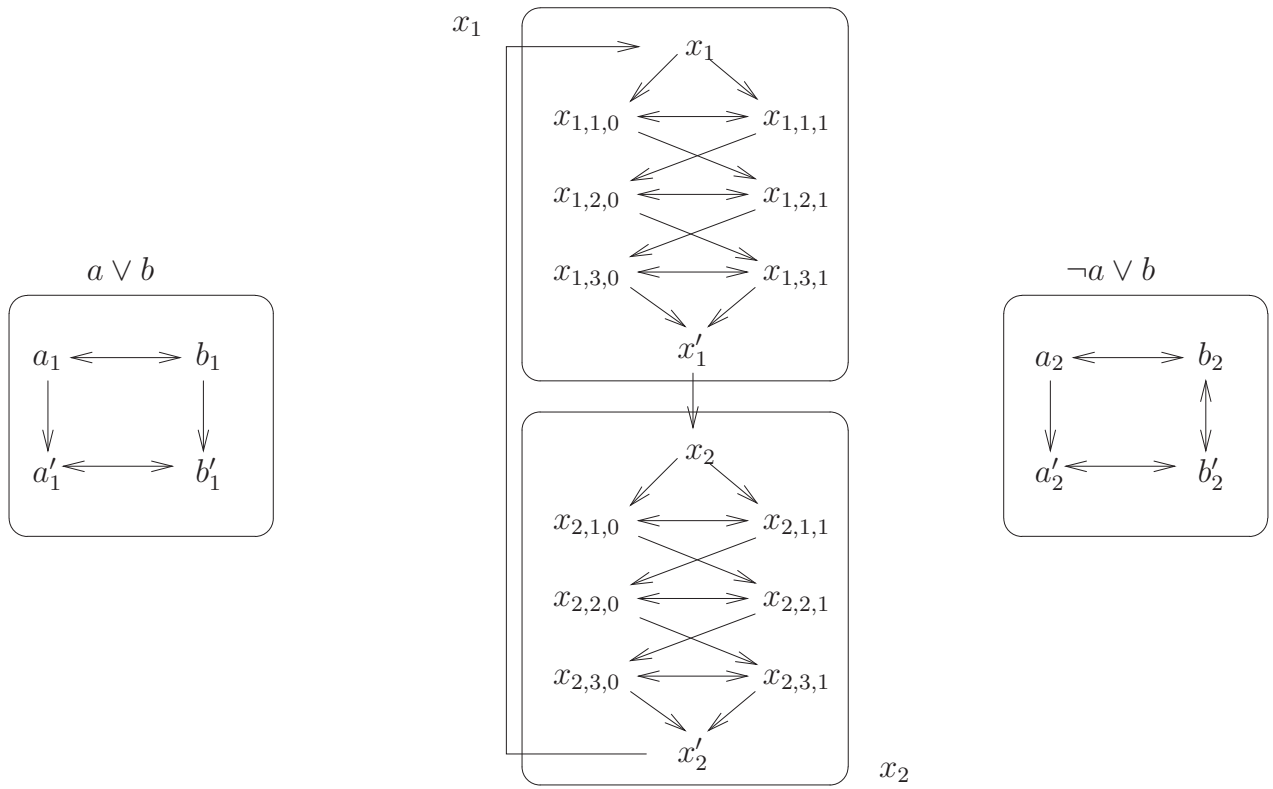
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



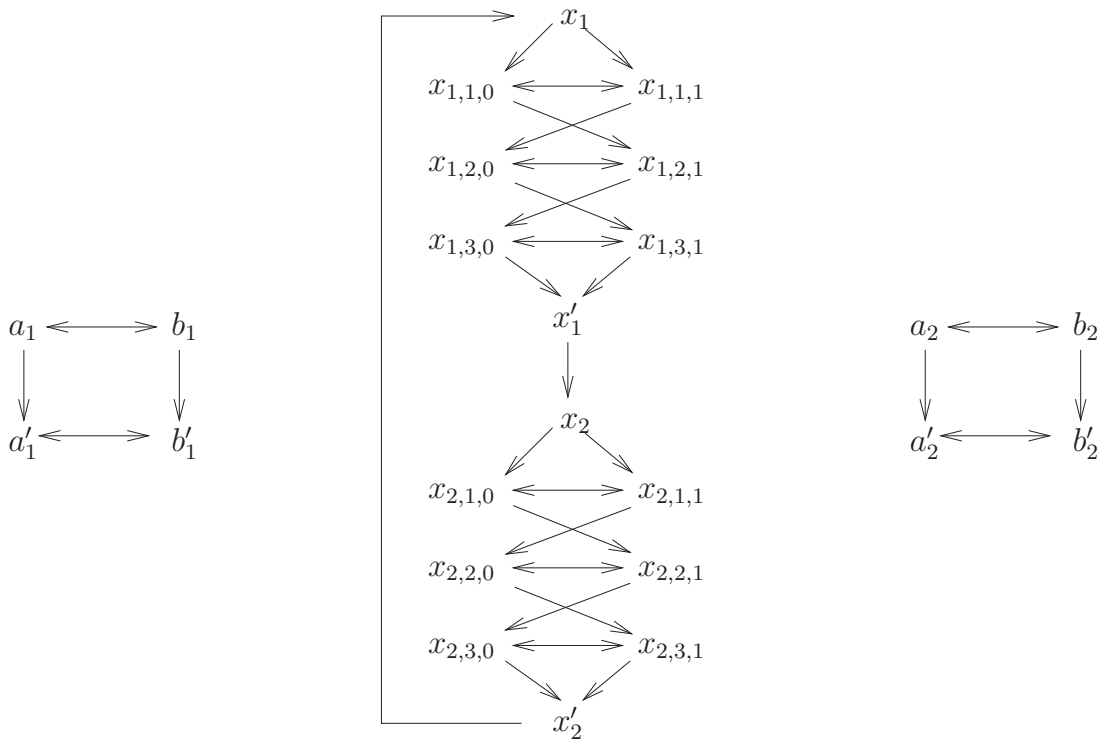
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



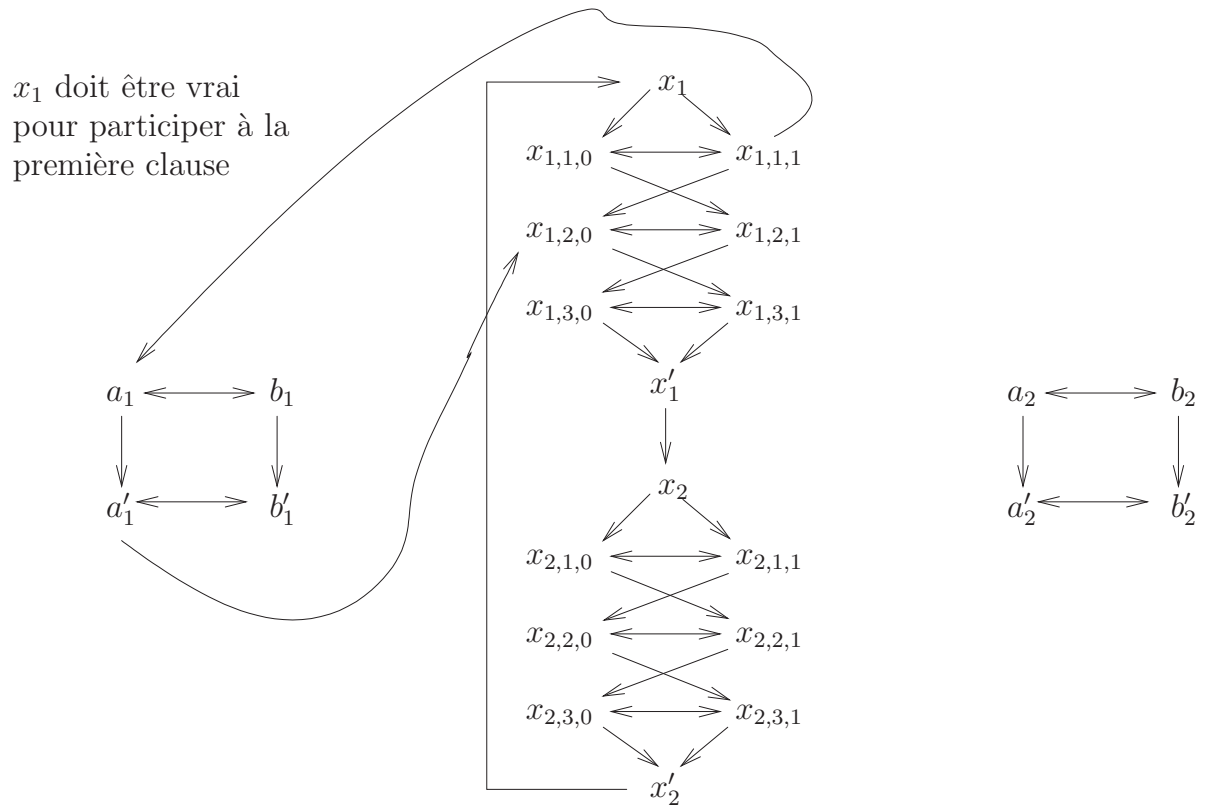
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



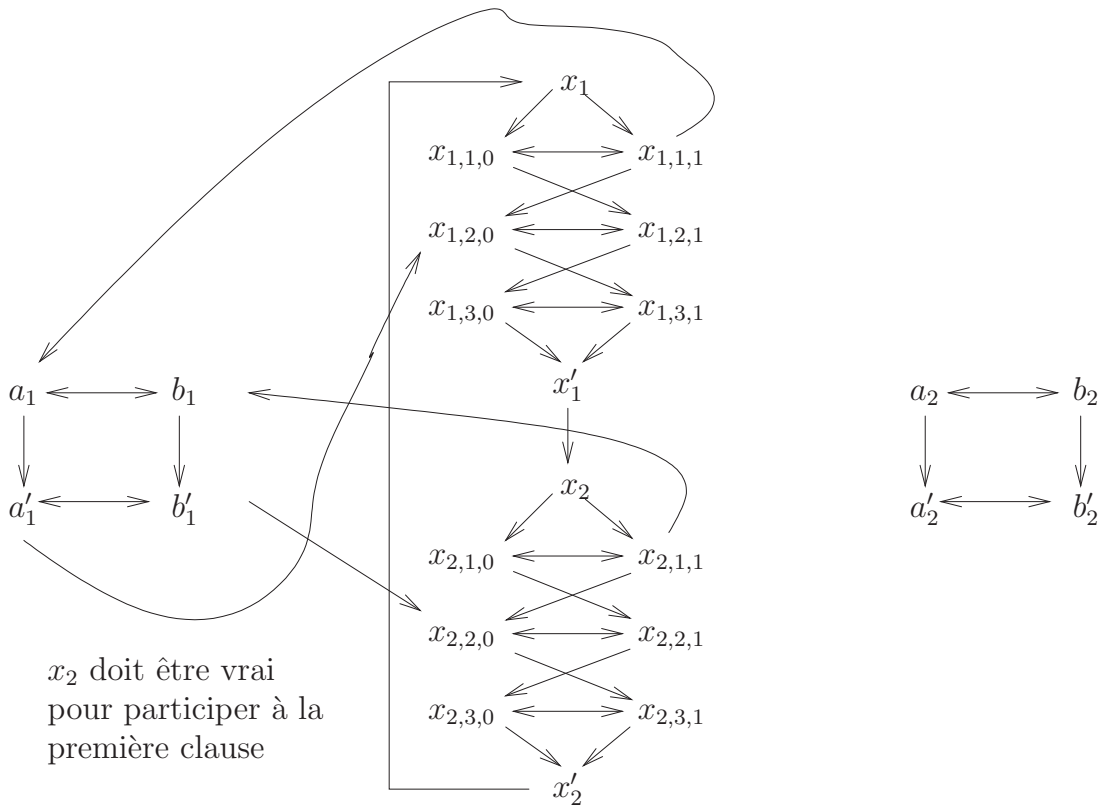
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



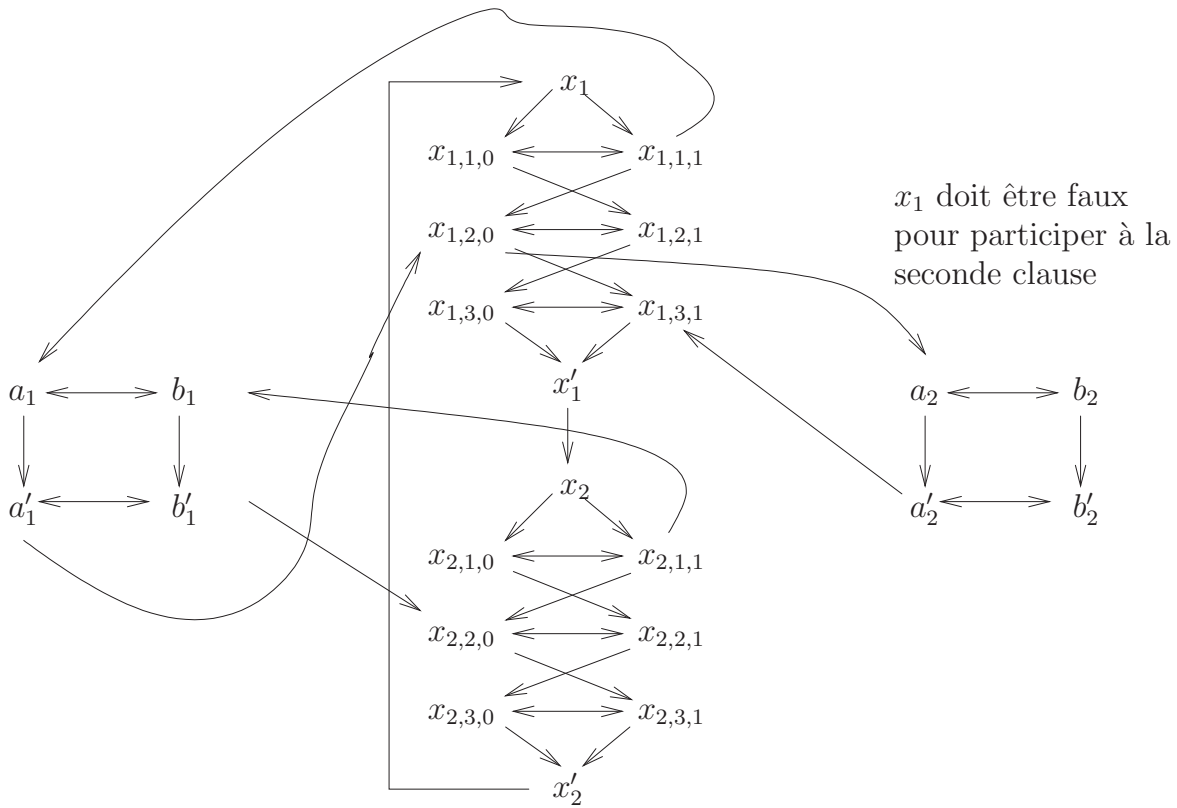
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



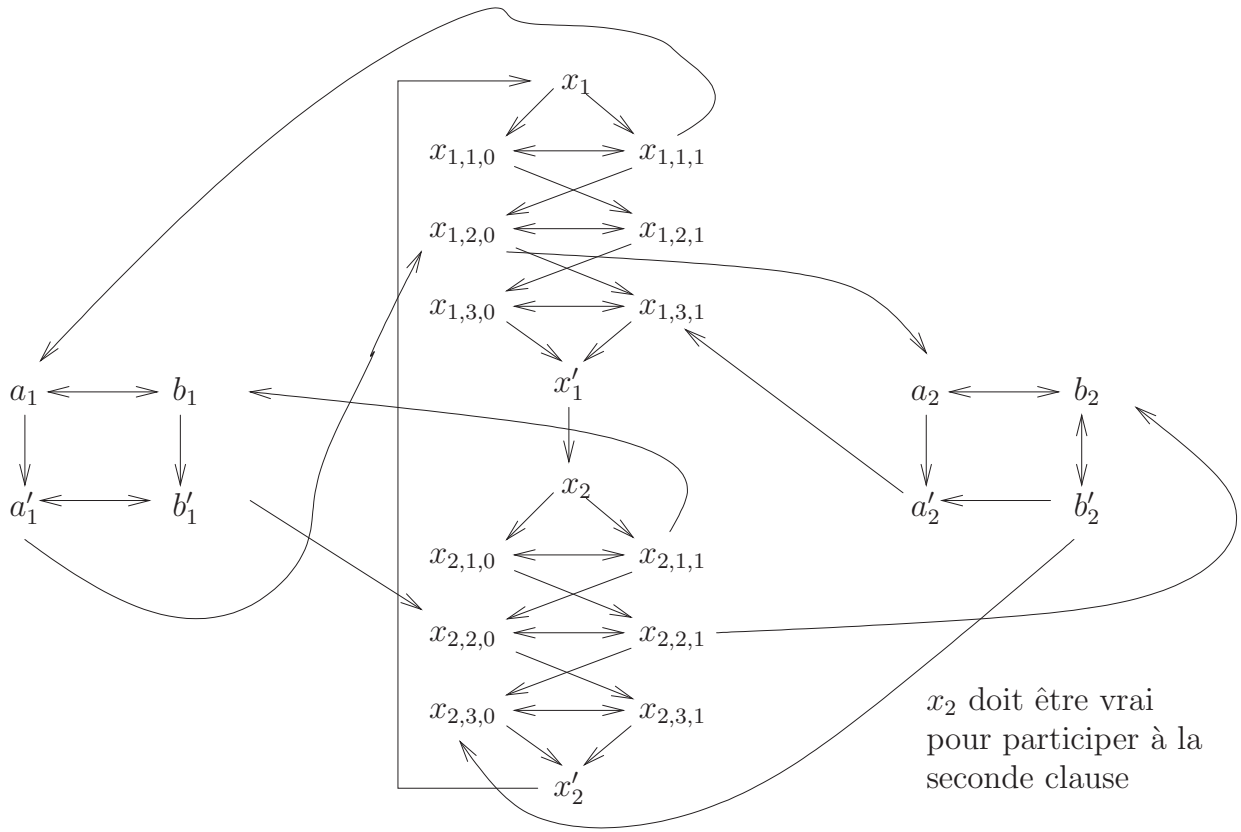
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



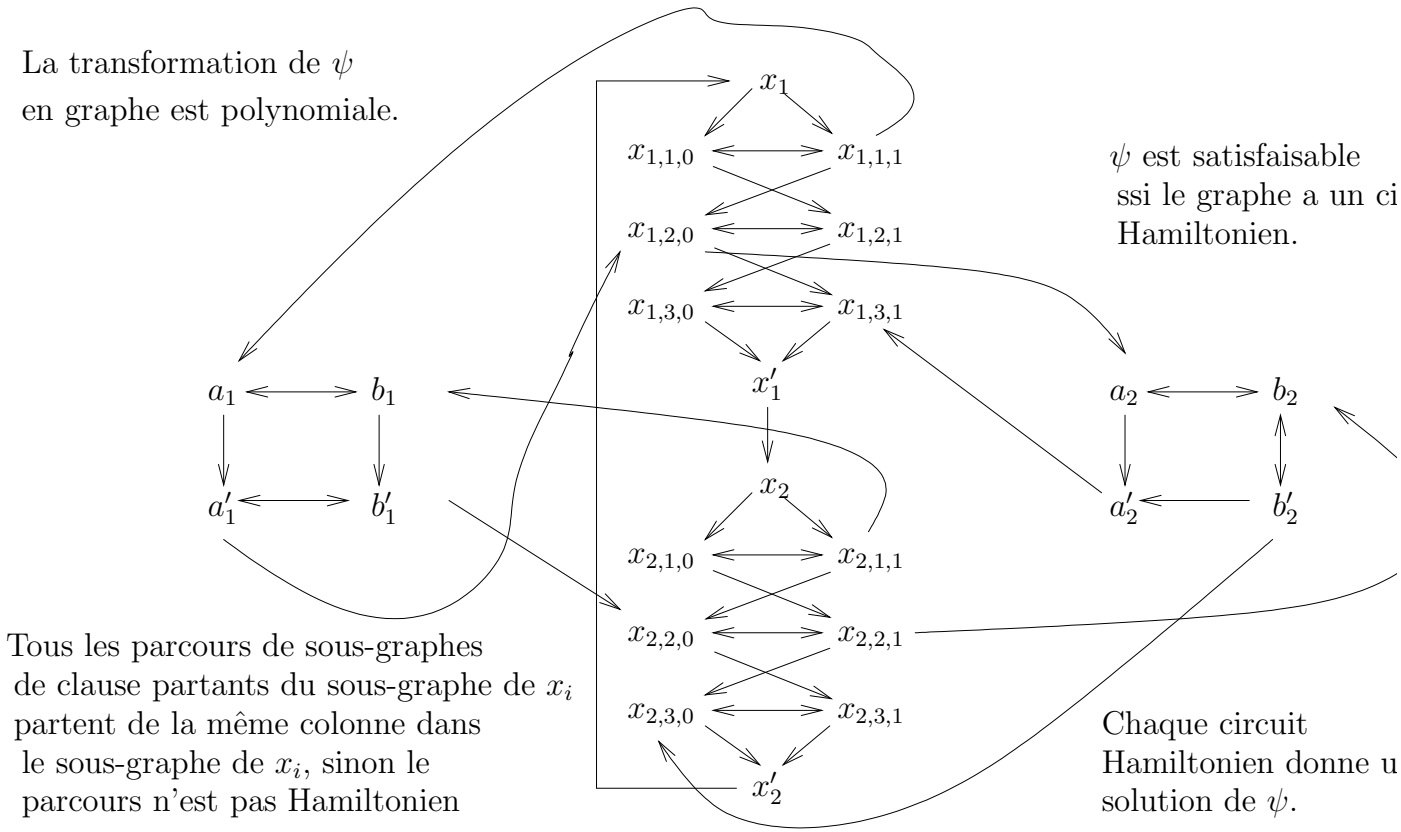
Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



Circuit Hamiltonien : un problème NP-complet - Complétude - Exemple

Ici sur une formule en 2-CNF $\psi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ mais fonctionne aussi en 3-CNF



Circuit Hamiltonien (version graphes non orientés) : un problème NP-complet - Complétude - Exemple

$H_u \in NP$ car :

- Si c est donné, on vérifie en temps polynomial qu'il s'agit d'un circuit Hamiltonien

Réduction

- Réduction de H_d à H_u
  À partir d'un graphe dirigé $G = (E, V)$ on construit un graphe non dirigé G' :
  - contenant trois nœuds v_0, v_1, v_2 pour chaque $v \in V$;
  - avec les arêtes $v_0 \leftrightarrow v_1$, $v_1 \leftrightarrow v_2$, $v_2 \leftrightarrow w_0$ ssi $v \rightarrow w \in E$.

  Alors, G' a un circuit Hamiltonien ssi G en a un
  (passer par $v_1 \leftrightarrow v_0$ empêche ensuite de ressortir de v_0)

D'autres problèmes NP-complets : Nombre chromatique d'un graphe

Donnés

- un graphe
- un entier k

déterminer si les nœuds de G peuvent être coloriés par k couleurs différentes sans que deux nœuds adjacents aient la même couleur.

Nombre chromatique d'un graphe : plus petit nombre de couleurs

Applications :

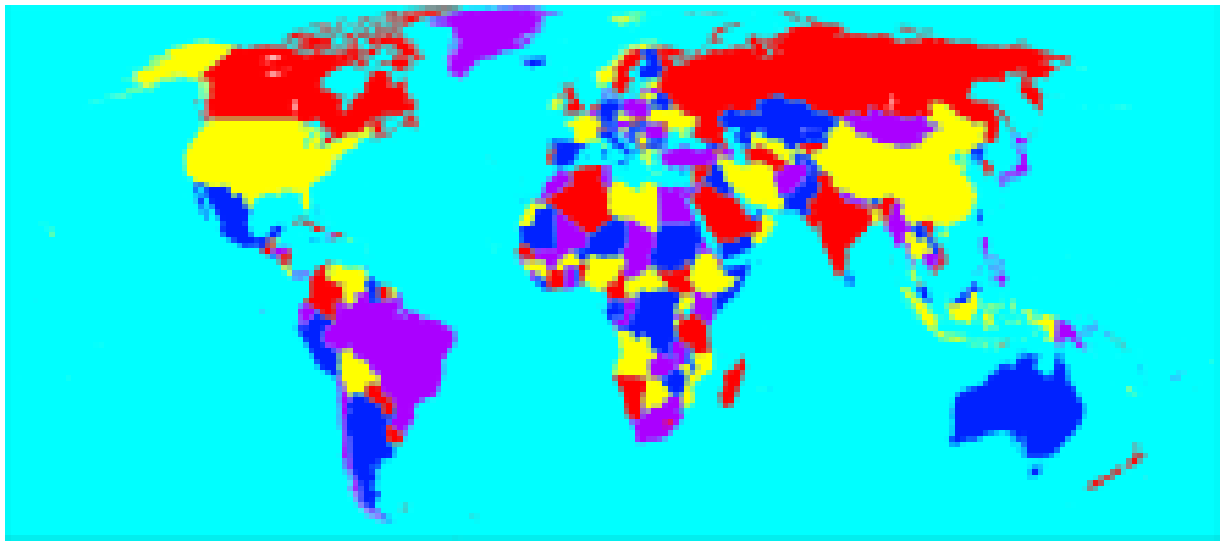
- Ordonnancements de tâches $\mathcal{T} = \{T_1, \ldots, T_n\}$
  - Un nœud = une tâche
  - conflit entre t et t' (par exemple parce qu'elles ont une ressource partagée) = arête $t \leftrightarrow t'$

  Nombre chromatique = plus petit temps d'exécution pour $\mathcal{T}$
- Allocation de registres (technique similaire)
- ...

D'autres problèmes NP-complets : Nombre chromatique d'un graphe

Cas particulier : le problème des 4 couleurs

Carte = plan divisé en régions



Source de l'image : wikipédia

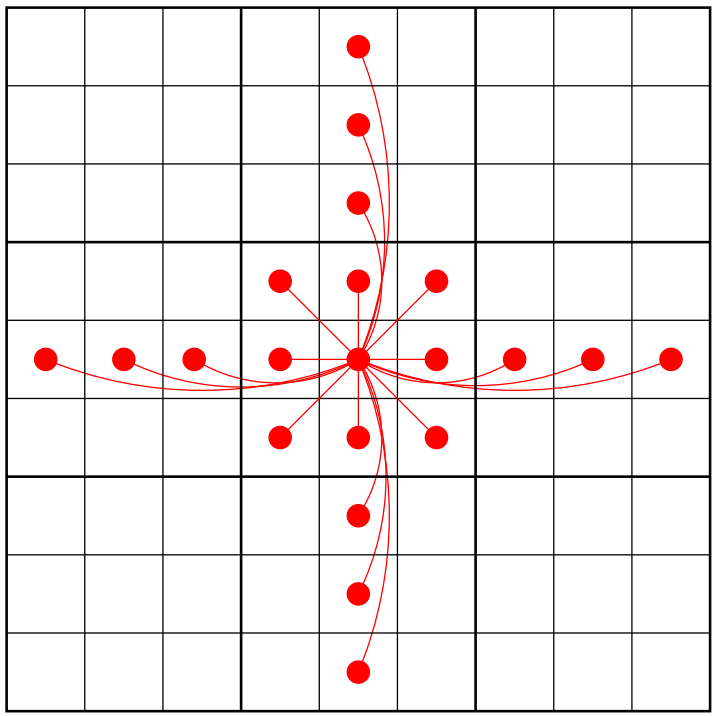
Théorème des 4 couleurs : 4 couleurs suffisent pour colorier n'importe quelle carte

- Formulation : Möbius (1840)
- Conjecture : Guthrie (1852)
- Preuve : Appel-Haken (1976)
  - requière l'examen de plusieurs milliers de cas
  - première preuve importante réalisée avec l'aide de l'ordinateur

D'autres problèmes NP-complets : Nombre chromatique d'un graphe

Cas particulier : Sudoku

- chaque case est un nœud
- on relie chaque case avec toutes celles qui ne peuvent avoir la même valeur
- il faut trouver un coloriage à 9 couleurs



Sudoku généralisé est NP-complet

- Réduction du carré latin à Sudoku
- Carré latin : matrice avec une unique valeur par ligne et par colonne
- Le problème de la complétion d'un carré latin est NP-complet

D'autres problèmes NP-complets : le voyageur de commerce *SP*

Problème Donnés :

- un graphe complet $G = (E, V)$
- V : les villes, E : les routes
- les arêtes de E sont valuées (longueur des routes)
- un entier k

Trouver un chemin Hamiltonien c de longueur $\leq k$.

Ce problème est dans NP car :

- Si c est donné, on vérifie en temps polynomial qu'il s'agit d'un circuit Hamiltonien de longueur $\leq k$

Réduction

- Réduction de H_d à *SP*
  - deux nœuds sont connectés : poids 1
  - ils ne le sont pas : poids $> k$

Les classes co-$\mathcal{C}$

- $\mathcal{C}$: une classe de complexité
- co-$\mathcal{C}$: classe des problèmes de $\mathcal{C}$ dont le complémentaire est aussi dans $\mathcal{C}$

$$\text{co-}\mathcal{C} = \{P : c(P) \in \mathcal{C}\}$$

- Exemple : co-NP est la classe des problèmes dont le complémentaire est dans NP.
- $L \in$ co-NP ssi il existe une MTND M fonctionnant en temps polynomial telle que $L(M) = L$ et

  $w \in L$ ssi toutes les exécutions de $M(w)$ sont acceptantes

- Exemple de problème co-NP : « toutes les instantiations des variables d'une formule en CNF sont-elles solutions ? »
- Question ouverte : « NP=co-NP ? »

Si P=NP Alors il existe un algorithme polynomial qui décide de la satisfaisabilité de ψ en FNC.

- oui, mais peut-on le trouver ?
- et peut-il donner une solution de ψ ?

NP clôt par complémentaire si et seulement si le complément d'un problème NP-complet est dans NP