Réductions, Problèmes Complets

Réductibilité

- Soient $L_1, L_2 \subset \Sigma^*$. On dit que L_1 **est log-space réductible à** L_2, noté $L_1 \leq L_2$ s'il existe une fonction $f : \Sigma^* \to \Sigma^*$ calculable en espace $O(\log n)$, telle que, pour tout $x \in \{0,1\}^*$,

$$x \in L_1 \text{ ssi } f(x) \in L_2.$$

- On peut définir une notion similaire en considérant P au lieu de L pour obtenir $\leq_P$.

- Lemme: Si $g : \Sigma^* \to \Sigma^*$ et $g' : \Sigma^* \to \Sigma^*$ sont calculables respectivement en temps f et f' alors $g \circ g'$ est calculable en temps $f \circ f'$.

  Démonstration: construire une machine qui sur une entrée w, simule la machine qui calcule g' sur w pour obtenir $g'(w)$, puis simule la machine qui calcule g sur $g'(w)$ pour obtenir $g(g'(w))$.

- Lemme: Si $g : \Sigma^* \to \Sigma^*$ et $g' : \Sigma^* \to \Sigma^*$ sont calculables respectivement en espace f et f' alors $g \circ g'$ est calculable en espace $max(f', f \circ f')$.

  Démonstration: On ne peut pas faire exactement la même chose que dans la preuve précédente, car il faut à priori stocker quelquepart $g'(w)$ qui peut être de taille plus grande que $f'(n)$ où $n = |w|$. Toutefois, chaque pas du calcul de g sur une entrée w' nécessite au plus de lire un bit, disons le bit numéro i, de w'.

  Or étant donné le numéro de ce bit i, codé en binaire, ($0 \leq i \leq f'(n)$), il est possible de calculer le $i$ième bit de $g'(w)$ en utilisant un espace f': simuler g' sur w tout en maintenant un compteur j qui compte le nombre de symbole écrit jusque l'à, jusqu'à ce que j vaille i.

  En maintenant la valeur de i, on simule le calcul de g sur l'entrée $w' = g'(w)$ sans jamais écrire complètement w': à chaque fois que l'on a besoin de lire un nouveau bit de w', on utilise la procédure précédente pour le déterminer.

Préordre de réductibilité

- **Théorème:** $\leq$ est un préodre:
  1. $L \leq L$
  2. $L_1 \leq L_2$, $L_2 \leq L_3$ implique $L_1 \leq L_3$.

Démonstration. 1. Considérer la fonction identité comme fonction f.
2. Supposons $L_1 \leq L_2$ via la fonction f, et $L_2 \leq L_3$ via la fonction g. On a $x \in L_1$ ssi $g(f(x)) \in L_2$.

Le Lemme précédent garanti que $g \circ f$ est calculable en espace logarithmique.

Réductibilité: application

- **Théorème:** $L_1 \leq L_2$, $L_2 \in P$ implique $L_1 \in P$.

  Démonstration. L_1 **est décidé par l'algorithme qui sur une entrée** x **calcule** $f(x)$ **et retourne la réponse de l'algorithme pour** L_2 **sur** $f(x)$**.**

- **Corollaire:** $L_1 \leq L_2$, $L_1 \notin P$ implique $L_2 \notin P$.

  Démonstration. C'est la contraposé du théorème précédent.

- **Théorème:** les mêmes résultats sont vrais si on remplace P par les classes de complexité L, NL, NP, $PSPACE$,.... dans les résultats précédents.

  Démonstration identique.

Problème complet

- Soit C une classe de complexité standard.
  Un problème $L \in C$ est C**-complet** si
  1. $L \in C$.
  2. $\forall L' \in C$, $L' \leq L$.
- **Définition:** Deux problèmes L_1 et L_2 sont équivalents, noté $L_1 \equiv L_2$, si $L_1 \leq L_2$ et $L_2 \leq L_1$.
- **Théorème:** Tous les problèmes complets sont équivalents.

  Démonstration. Evident.

Sur l'existence de problèmes complets

- **Remarque**: si seulement l'EXISTENCE de problèmes complets nous intéresse . . . **Théorème:** il existe un problème
  1. NL-complet: $K = \{ < M, x, 1^k > \mid$ la MT NDM accepte x en espace $\log(k)\}$
  2. P-complet: $K = \{ < M, x, 1^k > \mid$ la MT M accepte x en temps $k\}$
  3. NP-complet: $K = \{ < M, x, 1^k > \mid$ la MT ND M accepte x en temps $k\}$
  4. PSPACE-complet: $K = \{ < M, x, 1^k > \mid$ la MT M accepte x en espace $k\}$.
  5. EXPTIME-complet: $K = \{ < M, x, k > \mid$ la MT M accepte x en espace $k\}$.

Démonstration. Exemple de NP: le problème K est NP: pour décidér si $w =< M, x, 1^k >\in K$, il suffit de simuler M sur x pendant un temps k. Cela nécessite un temps polynomial (quadratique) en k, et donc polynomial en $|w|$, par une machine de Turing non-déterministe.

Soit L un problème NP. L est reconnu par une machine de Turing non-déterministe M_0 en temps n^k. Considèrer $f : x \mapsto < M_0, x, 1^{|x|^k} >$. f est bien calculable en espace $O(\log n)$. On a $x \in L \Leftrightarrow < M_0, x, 1^{n^k} >\in K$, et donc $L \leq K$.

Circuit booléen

- **Définition:** un circuit booléen est un graphe orienté sans boucle. Chaque sommet est est de degré entrant 0,1 ou 2. Les sommets de degré entrant 0 sont étiquetés soit par 0, soit par 1 soit par une variable (dans ce dernier cas on dit que c'est une entrée), ou par la constante 0 ou par la constante 1. Les sommets de degré entrant 1 sont étiquetés par $\neg$. Les sommets de degré entrant 2 sont étiquetés par $\vee$ ou $\wedge$. Certains noeuds sont identifiés comme des sorties.
- Chaque circuit C à n-entrées et m-sorties définit naturellement une fonction booléenne à n-variables, i.e. une fonction $f : \{0,1\}^n \to \{0,1\}^m$.

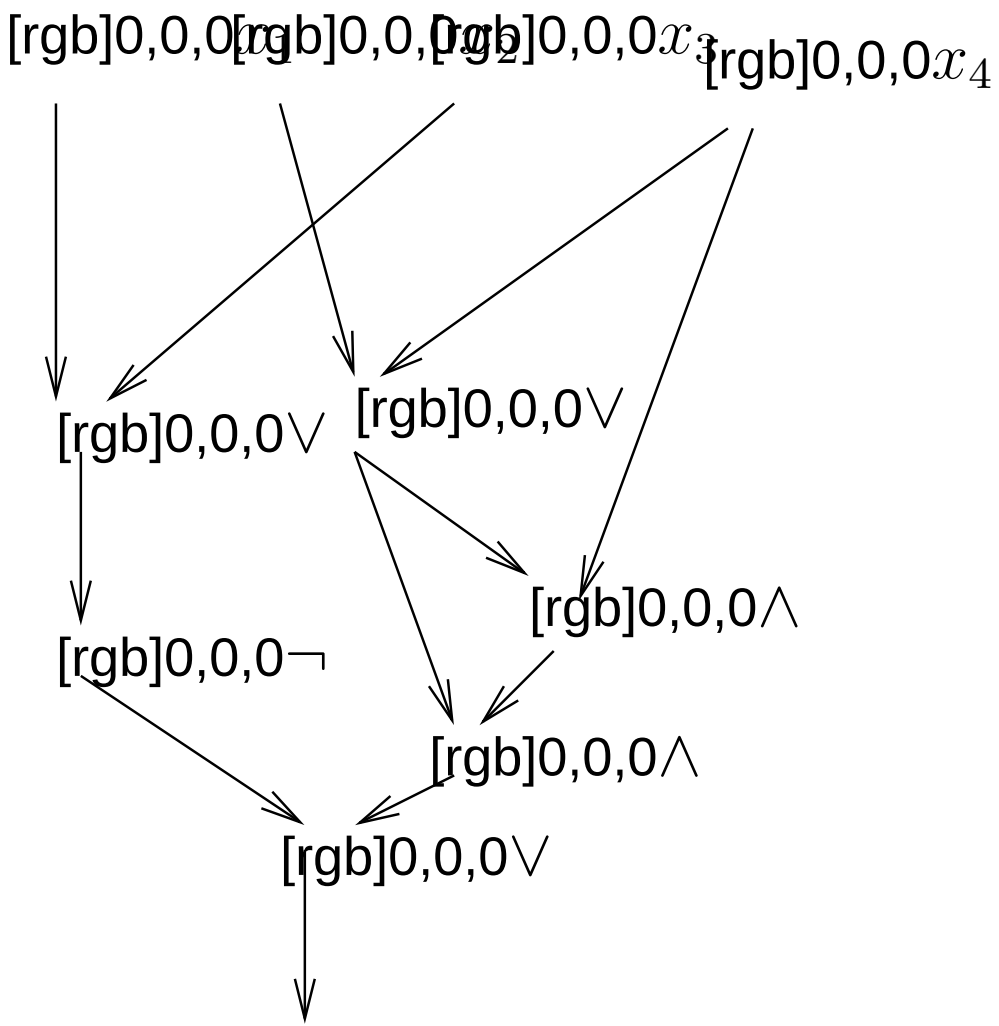

Figure 1: Un circuit.

Un problème P-complet

Problème ***CIRCUITVALUE*** :

- **Instance:** Un circuit C à n-variables, et $a \in \{0,1\}^n$
- **Question:** $C(a)$?
- **Théorème:** CIRCUITVALUE est P-complet.

Démonstration. Clairement CIRCUITVALUE (CV) est dans P. Soit L dans P. On montre que $L \leq CIRCUITVALUE$. L est reconnu par une machine M à un ruban en temps n^k. La configuration $(q, u_1 \# u_2)$ de la machine M sur l'entrée x au temps t peut se coder par le mot $u_1 q u_2 \in \Delta^*$ avec $\Delta = \Gamma \cup Q \cup B$. Soit L_t le codage de ce mot sur l'alphabet $\{0, 1\}$.

La longueur de L_t est borné par $c.n^k$, pour une constante c. On considère le tableau $(T_{i,j})_{1 \leq i \leq n^k, 1 \leq j \leq cn^k}$ où $T_{i,j} \in \{0, 1\}$ est la $j$ème lettre de L_i. $T_{i,j}$ est donnée par une fonction f_M de $T_{i-1,j-k_0}, T_{i-1,j-k_0+1}, \ldots, T_{i-1,j+k_0}$, pour une certaine constante k_0, indépendante de i et j. f_M peut donc se coder par un circuit de taille constante. En répètant $c.n^k$ fois ce circuit, on peut construire un circuit indépendant de i qui donne L_i en fonction de L_{i-1}. En répetant n^k fois ce dernier, on peut obtenir un circuit qui donne le tableau T, et donc le résultat du calcul de M sur l'entrée codée par L_1.

Considèrer la fonction f qui à x associe l'instance de CV constituée de ce circuit et du codage de la configuration initiale correspondante à x. On a $x \in L$ ssi $f(x) \in CV$.

La fonction f se calcule bien en utilisant un espace $O(\log n)$.

Autres problèmes P-complets

- **Définition:** Un circuit monotone est un circuit sans négation.
- Problème ***MONOTONECIRCUITVALUE*:**
  - **Instance:** Un circuit monotone C à n-variables, et $a \in \{0,1\}^n$
  - **Question:** $C(a)$?
- **Théorème:** MONOTONECIRCUITVALUE est P-complet.

Démonstration. MCV **est clairement dans** P **(en particulier car** $MCV \leq CV$ **de façon évidente via la fonction identité** f**.)**
$CV \leq MCV$**: on élimine les négations par les lois de Morgan:**
$\neg(x \vee y) = (\neg x) \wedge (\neg y)$ **et** $\neg(x \wedge y) = (\neg x) \vee (\neg y)$**. On double le nombre de portes: on construit une fonction** g **et** $\neg g$ **pour chaque porte.**

Le circuit obtenu se calcule facilement à partir du premier, i.e. avec un espace $O(\log n)$**.**

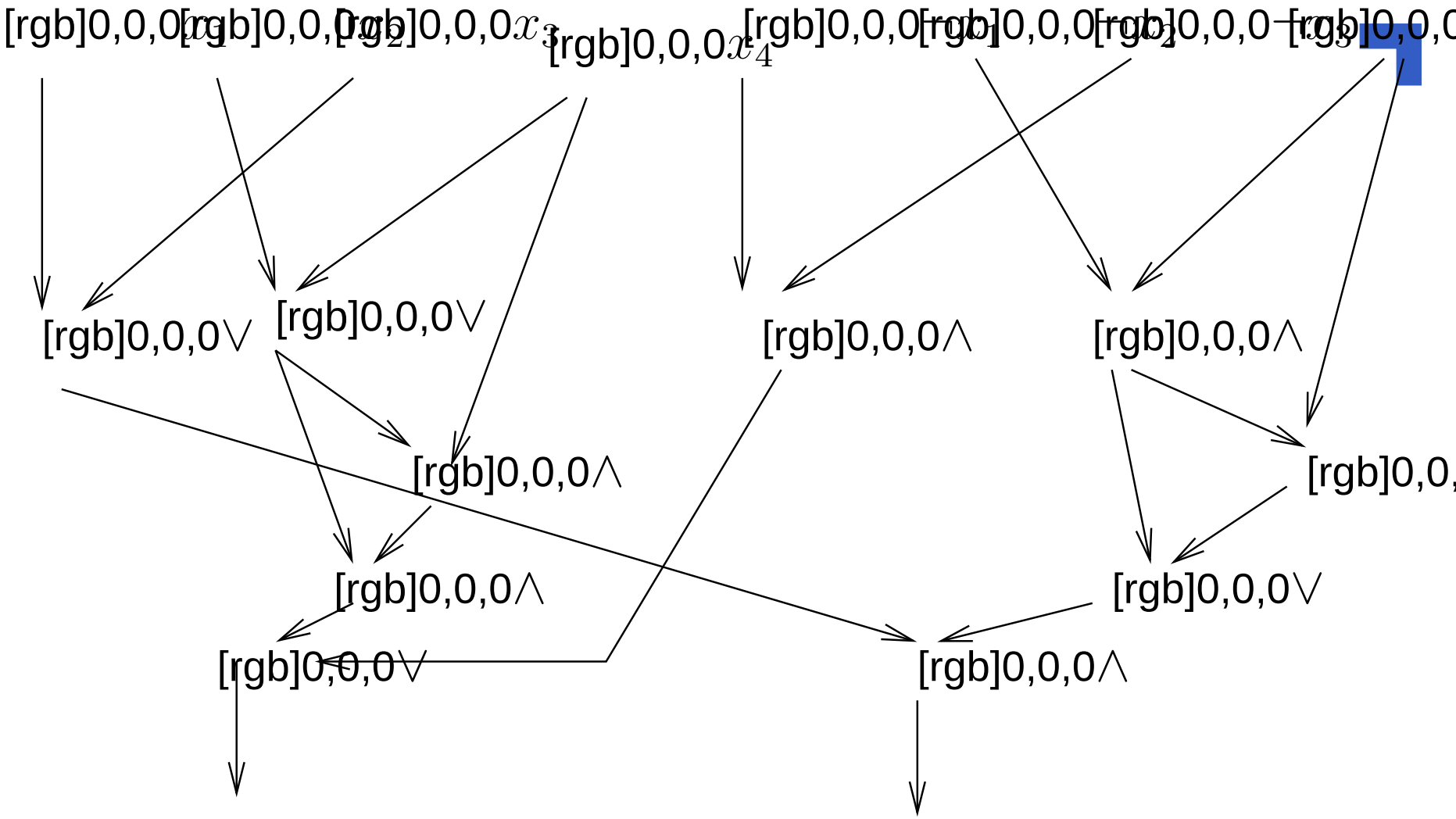

Figure 2: Le circuit monotone correspondant au circuit de la figure précédente.

Relation binaire polynômiale

- **Définition:** Une relation binaire $R(x, y)$ est polynômiale si:
  1. $R \in P$.
  2. $\exists$ un polynôme $p(n)$ tel que $\forall x, y, R(x, y) \Rightarrow |y| = p(|x|)$.

Relation binaire polynômiale

- **Théorème:** $L \in NP$ ssi $\exists R$ relation polynômiale telle que pour tout x, $x \in L$ ssi $\exists y\ R(x, y)$.

Démonstration. $\Leftarrow$**: considérer la machine qui devine un** y **de longueur** $p(|x|)$ **et calcule ensuite** $R(x, y)$**, et accepte ssi** $R(x, y)$**.**
$\Rightarrow$**: Soit** $L \in NP$**.** L **est reconnu par une certaine machine de Turing non deterministe** M **en temps** n^k**.**
Quitte à remplacer les choix non-déterministes de M **par des suties de choix non-déterministes binaires, on peut supposer que le degré de non-déterminisation de** M **vaut** 2**.**
Une suite de choix binaires non-déterministes de M **sur une entrée** w**, se code (de façon non-injective mais surjective) par un mot** y **sur** $\{0, 1\}^t$**, avec** $t = n^k$**: lorsqu'un choix non deterministe est effectué au temps** i**, le bit** i **vaut** $0, 1$ **suivant le choix effectué. Lorsqu'aucun choix non-deterministe n'est effectué au temps** i**, le bit** i **vaut** 0 **ou** 1 **indifféremment.**
Considérer la relation $R(x, y)$ **qui vaut** 1 **si** $|y| = |x|^k$ **et si la machine** M **accepte l'entrée** x **avec la suite de choix non-déterministes donnée par** y**.**

R **est bien calculable en temps polynomial (vérifier la longueur de** y **puis simuler** M **sur** x **avec les choix** y**), et on a bien toujours** $|y| = n^k$ **lorsque** $R(x, y)$**, et d'autre part** $x \in L$ **ssi** $\exists y\ R(x, y)$**.**

Problèmes NP-complets

- Problème ***CIRCUITSAT*** :
  - **Instance:** Un circuit C
  - **Question:** Existe t'il a tel que $C(a) = 1$?

  Problème ***SAT*** :
  - **Instance:** Une formule booléenne en CNF
  - **Question:** Existe t'il a tel que $C(a) = 1$?
- **Théorème:** CIRCUITSAT est NP-complet.

Démonstration. Soit $L \in NP$ et R la relation polynômiale correspondante. Soit C le circuit correspondant au calcul de la machine qui calcule R comme dans la preuve de la P-complètude de CircuitValue: C prend en entrée x, y et retourne en sortie $R(x, y)$.

Soit f la fonction qui à un mot x associe le circuit $C(x, .)$, où les entrées de C correspondantes à x sont fixées à x et celles correspondantes à y laissées comme variables.

On a $x \in L$ ssi $\exists y\ R(x, y)$ ssi $f(x) \in CIRCUITSAT$.

Problèmes NP-complets

- **Théorème:** SAT est NP-complet.

Démonstration. Puisque SAT est clairement dans NP, on montre qu'on peut transformer un circuit en une formule booléenne CNF: pour chaque porte g du circuit on introduit une formule ϕ_g: pour g correspondant à $\neg g'$ (respectivement $g_1 \vee g_2$, $g_1 \wedge g_2$), on pose $\phi_g = (\neg g \vee \neg g') \wedge (g \vee g')$, (resp. $(\neg g \vee g_1 \vee g_2) \wedge (\neg g_1 \vee g) \wedge (\neg g_2 \vee g)$, $(g \vee \neg g_1 \vee \neg g_2) \wedge (\neg g \vee g_1) \wedge (\neg g \vee g_2)$). Prendre ensuite $\phi = \bigwedge_g \phi_g$. Pour que ϕ soit satisfiable, il faut que chaque ϕ_g soit satisfaite, ce qui implique $g = \neg g'$ (resp. $g = g_1 \vee g_2$, $g = g_1 \vee g_2$).

Problèmes NP-complets

- Problème k***SAT*****:**
  - **Instance:** Une formule booléenne en CNF où chaque clause a k-littéraux
  - **Question:** Existe t'il a tel que $C(a) = 1$?
- **Théorème:** 3-SAT est NP-complet.

Démonstration. En fait la démonstration précédente produit une formule où chaque clause a au plus 3-littéraux. Les clauses de moins de 3 littéraux peuvent se remplacer par des clauses à trois littéraux: exemple $\alpha \vee \beta$ peut se remplacer par $(\alpha \vee \beta \vee \gamma) \wedge (\alpha \vee \beta \vee \neg\gamma)$.

Des problèmes NP-complets

- HAMILTON.

  Démonstration. Non donnée.

- Problème ***PARTITION*:**
  - **Instance:** Des entiers $a_1, \ldots, a_n$
  - **Question:** Existe t'il $S \subset \{1, 2, \ldots, n\}$ tel que $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$?

  Démonstration. Non donnée.

Des problèmes NP-complets

- Problème ***INDEPENDANTSET*** :
  - **Instance:** $G = (V, E)$,$k \in N$
  - **Question:** Existe t'il un ensemble de sommets stable de taille $\geq k$? (i.e. $V' \subset V, \forall u, v \in V', (u, v) \notin E$?)

Démonstration. INDEPENDANTSET est clairement dans NP. On montre que $3SAT \leq INDEPENDANTSET$**. Soit** $\phi = \wedge_{i=1,\ldots,n}(\alpha_{i,1} \vee \alpha_{i,2} \vee \alpha_{i,3})$ **avec** $\alpha_{i,j} \in \{X_1, \neg X_1, \ldots, X_l, \neg X_l\}$**. On construit** $G = (V, E)$ **avec** $V = \{c_{i,j} | i = 1 \ldots n, j = 1, 2, 3\}$**.** $E = \{(c_{i,1}, c_{i,2}), (c_{i,2}, c_{i,3}), (c_{i,1}, c_{i,3}) | i = 1, \ldots, n\} \cup (c_{i,j}, c_{i',j'}) | \exists X \alpha_{i,j} = X, \alpha_{i',j'} = \neg X$**. // Sens** $\Rightarrow$**: Supposons** ϕ **satisfiable. Cela implique** $\forall i \exists j_i \alpha_{i,j_i} = 1$**: l'ensemble** $V' = \{c_{i,j_i} | i = 1, \ldots, n\}$ **forme un ensemble stable de taille** n**. // Sens** $\Leftarrow$**: Un stable de taille** n **contient nécessairement exactement un sommet par triangle. Considérer** $\alpha_{i,j} = 1$ **ssi** $c_{i,j}$ **est dans le stable.**

Illustration

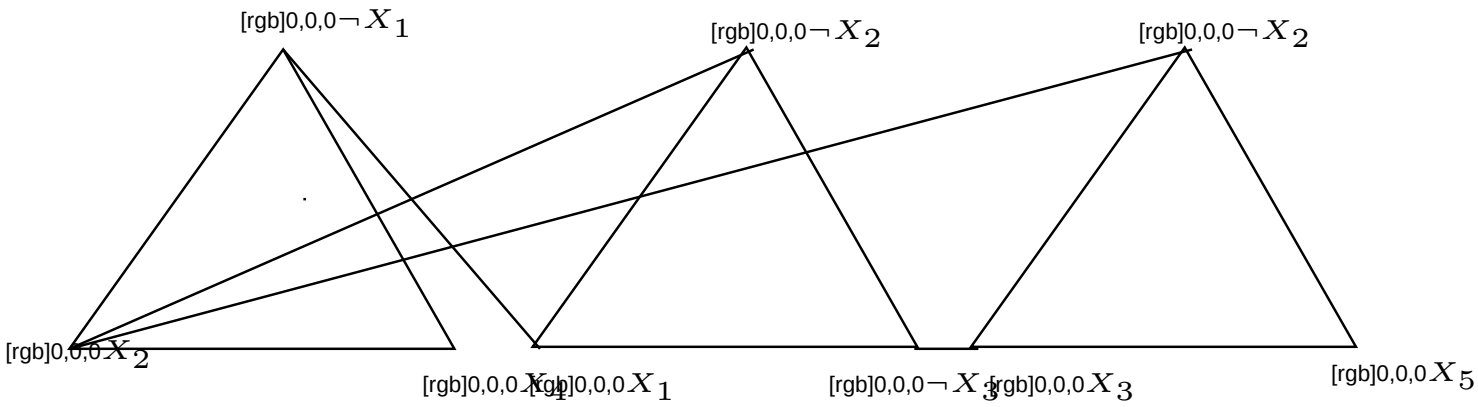

Figure 3: Le graphe correspondant à la satisfaisabilité de la formule $(\neg X_1 \vee X_2 \vee X_4) \wedge (X_1 \vee \neg X_2 \vee \neg X_3) \wedge (X_3 \wedge \neg X_2 \wedge \neg X_5)$**.**

Des problèmes NP-complets

- Problème ***VERTEXCOVER*:**
  - **Instance:** $G = (V, E)$, $k \in \mathbb{N}$
  - **Question:** Existe t'il une couverture de taille $\leq k$? (i.e. $V' \subset V, \forall e \in E, V' \cap e \neq \emptyset$?)

Démonstration. $INDEPENDANCESET \leq VERTEXCOVER$ **. La réduction est donnée par la fonction** $(G = (V, E), k) \mapsto (G = (V, E), n - k)$ **car un ensemble de sommets** V' **est stable ssi** $V - V'$ **est une couverture.**

- Problème ***CLIQUE*:**
  - **Instance:** $G = (V, E)$,$k \in N$
  - **Question:** Existe t'il une clique de taille $\geq k$?

Démonstration. $INDEPENDANCESET \leq CLIQUE$**. La réduction est donnée par la fonction** $(G = (V, E), k) \mapsto (G^c = (V, E^c), k)$ **car un ensemble de sommets** V' **est stable ssi** V' **est une clique dans** G^c**.**

Un problème PSPACE-complet

- Problème ***QBF*** **:**
  - **Instance:** $\exists x_1 \forall x_2 \exists x_3 \forall x_3 \ldots \phi(x_1, \ldots, x_n)$, où ϕ sans quantificateur.
  - **Question:** Est-ce que ϕ est vraie?
- **Théorème:** QBF est PSPACE-complet.

Démonstration. QBF est dans PSPACE, car il est facile de construire un algorithme récursif en réutilisant l'espace qui le résoud en espace polynomial (en fait linéaire).

Complétude: Soit $L \in PSPACE$ accepté en espace $c.n^k$ par une machine M. On peut supposer sans perte de généralité que L est accepté en temps au plus d^{n^k} (car $SPACE(f) \subset NSPACE(f) \subset TIME(d^{f(n)+log(n)})$ pour une certaine constante d) pour une certaine constante d, et que lorsque M accepte tous les rubans sont vides.

On introduit des variables booléennes S_i, $1 \leq i \leq c.n^k$ pour coder une configuration: S_i vaut 1 ssi la $i$ème lettre de la configuration codée en binaire vaut 1.

On définit $CHEMIN(I, I', t)$ vrai ssi la machine M passe de la configuration I à la configuration I' en moins de 2^t étapes . On a $x \in L$ ssi $CHEMIN(C[x], C_{final}, \log(d^{n^k}))$, où $C[x]$ est la configuration initiale correspondante au mot x, et $C_{final} = (q_a, \#, \ldots, \#)$ l'unique configuration acceptante.

On a envie d'écrire

$CHEMIN(I, I', t) \Leftrightarrow \exists I_m CHEMIN(I, I_m, t-1) \wedge CHEMIN(I_m, I', t-1)$**. On écrit plutôt**

$CHEMIN(I, I', t) \Leftrightarrow \exists I_m \forall (I_A, I_B) \in \{(I, I_m), (I_m, I')\} CHEMIN(I_A, I_B)$.

En écrivant récursivement $CHEMIN(I, I', t)$ comme ci-dessus pour $t = 1, \ldots, logd^{n^k}$, on obtient une formule quantifiée de taille $O(n^{2k})$ qui décide si $x \in L$. On peut la mettre sous forme prénexe QBF en observant que les quantificateurs sont toujours sur des variables à leur droite.

Un problème NL-complet

- **Théorème:** REACH est NL-complet.

  Démonstration. REACH est dans NL car il suffit à un algorithme non-déterministe de deviner les noeuds intérmédiaires du chemin. On a besoin seulement de l'espace $O(\log n)$, pour stocker en le numéro du noeud courant en binaire.

  REACH est Complet car $w \in L$ ssi $(G(M,w), C[w], C_{final}) \in Reach$, où $G(M,w)$ est le graphe des configurations, $C[w]$ la configuration initiale correspondante au mot w, et C_{final} l'unique configuration acceptante.

  La fonction qui à w associe $(G(M,w), C[w], C_{final})$ se calcule bien en espace $O(\log n)$.

- **Théorème:** 2-SAT est NL-complet.

  Démonstration. On ne fournira pas la démonstration.