Corrections[1]

Complexité de k-Sat

1. Montrez que 1-SAT peut être résolu en temps polynomial.

Algo (décision).
variable : un tableau t indexé booléen par les variables de la formule (assignement partiel).
On parcourt successivement la liste des clauses de la formule.
Si la clause est x et que $t[x]$ est 0 (faux)
alors on arrête et on répond NON
sinon on met $t[x] := 1$.
Si la clause est $\neg x$ et que $t[x]$ est 1 (vrai)
alors on arrête et on répond NON
sinon on met $t[x] := 1$.

complexité : linéaire.

2. Montrez que 2-SAT peut être résolu en temps polynomial.

étape 1 On se ramène d'abord à une formule ayant 2 variables différentes dans chaque clause.
On enlève les clauses de la forme $x \vee \neg x$ (trivialement satisfaites et on remplace une clause de la forme $x \vee x$ par x (de même pour $\neg x \vee \neg x$ par $\neg x$)
Ensuite, si une clause a un seul littéral positif x alors (on simule $x := 1$)
on enlève les clauses contenant x
on enlève $\neg x$ des autres clauses (et si une clause devient vide, on répond NON)
De même, si une clause a un seul littéral négatif $\neg x$ alors (on simule $x := 0$)
on enlève les clauses contenant $\neg x$
on enlève x des autres clauses (et si une clause devient vide, on répond NON)
On itère cette opération tant qu'il existe des 1-clause.

étape 2 On se ramène à un problème d'accessibilité dans un graphe.
On construit le graphe des implications : il y a deux sommets par variable de la formule x et $\neg x$, et pour une clause $x \vee \neg y$ on ajoute les deux arcs $y \to x$ et $\neg x \to \neg y$ (on procède de manière similaire pour les autres types de 2 clauses).

[1] Tout commentaire, suggestion de clarification ou correction de la correction bienvenus.

lemme : La formule est satisfaisable ssi il n'y a pas de chemin orienté de x à $\neg x$ pour toute variable x.

étape 3 On pourrait calculer la clôture transitive du graphe. Ensuite si il y a un chemin orienté de x à $\neg x$ pour une variable x on répond NON, sinon on répond OUI.

Il suffit en fait de connaître les composantes fortement connexes du graphe. Si pour une variable x, on a x et $\neg x$ dans la même composante fortement connexe alors on répond NON, sinon on répond OUI.

complexité ? Si la formule initial à n variables et m clauses. Le prétraitement de l'étape 1 est en $O(n.m)$ (pour chaque littéral, on parcourt la formule pour des mises-à-jour). L'étape 2 construit un graphe avec $2n$ sommets et $max(2m, n(n-1))$ arcs.
L'algorithme de Tarjan pour un graphe orienté $G = (V, E)$ fonctionne en temps $O(|V| + 2|E|)$, donc pour l'étape 3 on a une complexité $O(2n + 4m)$.

Complexité : $0(nm)$ pour le prétraitement et linéaire ensuite.
remarque : dans la littérature et contrairement à notre définition, 2-Sat est la version stricte où chaque clause a exactement 2 littéraux portant sur des variables distinctes.

3. Montrez que si 3-SAT peut être résolu en temps polynomial alors SAT peut l'être aussi.

En introduisant une nouvelle variable z, on peut simuler une clause $\ell_1 \vee \ell_2 \vee \ell_3 \vee \ldots \vee \ell_k$ où $k \geq 4$ par deux clauses $\ell_1 \vee \ell_2 \vee z$ et $\neg z \vee \ell_3 \ldots \vee \ell_k$. En répétant cette transformation, on obtient une formule de 3-Sat qui est satisfaisable si et seulement si la formule initial est satisfaisable. La nouvelle formule à au plus $n+m$ variables et $m.n$ clauses (on peut ignorer les clauses triviales comportant à la fois x et $\neg x$ et supposer qu'une variable apparaît au plus une fois dans une clause).

Conséquence : 3-Sat est NP-complet par le théorème de Cook – Sat est NP-complet – et la transitivité des réductions.

Remarque : dans la littérature et contrairement à notre définition, 3-Sat est la version stricte où chaque clause a exactement 3 littéraux portant sur des variables distinctes. On peut ajouter des variables pour augmenter la taille des clauses trop petites (ou celles ayant des répétitions). Par exemple, pour une clause $x \vee y$, on ajoute une nouvelle variable z et on remplace $x \vee y$ par les deux 3-clauses strictes $x \vee y \vee z$ et $x \vee y \vee \neg z$.

Complexité de Horn-Sat

1. Soit φ une formule de Horn à n variables. On suppose qu'il existe une clause constituée d'un seul littéral positif x dans φ. Montrez que satisfaire φ revient à satisfaire une formule de Horn à $n-1$ variables.

On est obligé de mettre x à vrai. Comme précédemment pour 2-Sat, on peut simuler ce choix forcé en supprimant les clauses mentionnant x (maintenant satisfaites) et en supprimant les occurrences de $\neg x$ dans les autres clauses. Notons que l'on ne change pas le nombre de littéraux positifs dans une clause. La formule résultante est donc de Horn sur $n-1$ variables.

2. En utilisant le fait qu'on peut écrire une clause de Horn sous forme implicative, en déduire un algorithme de résolution pour HORN-SAT.

On utilise la *résolution unitaire*.
Ceci consiste à itérer l'opération décrite à la question précédente.
Si on obtient à un moment une clause x et une clause $\neg x$, alors on répond NON.
Si il n'y a jamais de tel problème, on répond OUI.

Le fait que l'algo est correct lorsqu'il répond NON est clair (l'algo est correct pour n'importe quelle formule SAT dans ce cas). Pourquoi l'algo est correct lorsqu'il répond oui ? On est obligé de mettre à vrai les variables éliminés au cours de la résolution unitaire. On peut ensuite mettre à faux toutes les autres variables et satisfaire ainsi trivialement toutes les clauses restantes (on utilise ici le fait qu'il s'agit de clauses de Horn et donc d'implications qui sont correctes puisque évalués à "faux implique faux").

3. Quelle est sa complexité dans le pire des cas en fonction de n (nombre de variables) et m (nombre de clauses).

Notre algorithme ci-dessus est quadratique dans le pire des cas (prendre l'exemple de la formule $x_1 \wedge x_2 \wedge \ldots \wedge x_n$ où toutes les variables sont distinctes). Il existe un algo linéaire [2].

4. Appliquer cet algorithme à

$$\varphi := (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_2 \vee \neg x_4) \\ \wedge (x_2 \vee \neg x_3) \wedge (\neg x_4 \vee \neg x_3) \wedge (x_3) \wedge (x_5 \vee \neg x_1 \vee \neg x_2 \vee \neg x_3)$$

Complexité du coloriage

1. Montrez que 2-col est polynomial.

[2] William F. Dowling, Jean H. Gallier : *Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae*. J. Log. Program. 1(3) : 267-284 (1984).

Il existe plusieurs algorithmes possibles. Le plus simple est de tenter sa chance avec 2 couleurs. On choisit un sommet et on lui donne la couleur 1. On donne la couleur 2 à tous ces voisins et ainsi de suite. Si à un moment on doit colorier un sommet qui est voisin de deux sommets déjà coloriés alors on répond NON, sinon on répond OUI.

Correction de l'algo. Si l'algo ci dessus répond non, on peut montrer facilement qu'il existe un cycle impair. Il est facile de monter qu'un cycle impair a nombre chromatique 3. Le graphe a donc nombre chromatique 3 ou plus.

complexité : linéaire en le nombre d'arêtes (variante d'un parcours de graphe).

Une autre méthode. On se réduit à 2-SAT. Pour chaque sommet x, on a une variable x. Pour chaque arête (x, y), on ajoute les deux clauses codant $x \iff \neg y : \neg x \implies y$ et $c \implies \neg y$.

2. Donnez un programme DATALOG pour le complément de 2-col.

DATALOG est une sorte de prolog théorique. L'instance est donnée par des relations (ici l'arête du graphe $E(x, y)$). On a le droit d'introduire de nouvelles relations (ici Odd et $Even$ deux nouvelles relations binaires). Le programme consiste en un ensemble de règles de la forme suivante :

conjonction **positives** de relations (de l'entrée ou nouvelle) implique nouvelle relation.

On a un prédicat spécial *échec*. Si ce dernier est dérivé on rejette, sinon le programme accepte. Par tradition on écrit les règles à l'envers.

$$
\begin{aligned}
Odd(x, y) &\leftarrow E(x, y) \\
Even(x, y) &\leftarrow Odd(x, z) \wedge E(z, y) \\
Odd(x, y) &\leftarrow Even(x, z) \wedge E(z, y) \\
echec &\leftarrow Odd(x, x)
\end{aligned}
$$

3. Montrez que H-col se réduit à 2-col lorsque H est un graphe biparti avec au moins une arête.

4. Montrez que 2-col se réduit à H-col lorsque H est un graphe biparti avec au moins une arête.

Les deux questions ci-dessus reposent sur le fait que si H est un graphe biparti avec au moins une arête, alors H est *homomorphiquement équivalent* à K_2 (la clique à deux éléments) : c'est-à-dire qu'il existe un homomorphisme $h : H \to K_2$ et un homomorphisme $i : K_2 \to H$. L'homomorphisme h existe puisque h est biparti (étant donnée une bipartition de H, l'application H envoie les sommets d'une bipartition sur une couleur et ceux de l'autre sur l'autre couleur). L'homomorphisme i existe puique H a une arête (on peut donc envoyer l'arête de K_2 sur cette arête).

Par composition d'homomorphismes, on a donc que pour tout graphe G, il existe un homomorphisme de G vers H ssi il existe un homomorphisme de G vers K_2.

Les deux problèmes s'inter-réduisent via la réduction triviale : il s'agit en fait du même problème de décision. Notez toutefois que si on s'intéresse à compter le nombre de solutions ou bien énumérer ces dernières, les deux problèmes sont en général différents et pas forcément « calculatoirement » équivalents.

Preuve du théorème de Chandra Merlin

Inclusion de requêtes : $\varphi_{\mathcal{B}} \subseteq \varphi_{\mathcal{A}}$ est une abréviation pour :
pour toute structure relationnelle (base de donnée) $\mathcal{D}$, si $D \models \varphi_{\mathcal{B}}$ alors $D \models \varphi_{\mathcal{A}}$. Démontrez le théorème suivant.

Théorème 1 (Chandra, Merlin '77)**.** *Il y a équivalence entre les points suivants.*

1. $\varphi_{\mathcal{B}} \subseteq \varphi_{\mathcal{A}}$
2. $\mathcal{B} \models \varphi_{\mathcal{A}}$
3. $\mathcal{A} \to \mathcal{B}$

La preuve de ce théorème repose sur la correspondance entre structure relationnelle et formule primitive positive[3] :

- à toute structure $\mathcal{A}$ de domaine $\{a_1, a_2, \ldots, a_n\}$ on associe une formule primitive positive. $\varphi_{\mathcal{A}}$ dont les variables sont les sommets de $\mathcal{A}$

$$\exists a_1 \exists a_2 \ldots \exists a_n \bigwedge_{R \text{ symbole d'arité } r \text{ de la signature}} \bigwedge_{(a_{i_1}, a_{i_2}, \ldots, a_{i_r}) \in R^{\mathcal{A}}} R(a_{i_1}, a_{i_2}, \ldots, a_{i_r}).$$

- et à toute formule primitive positive, on peut associer de manière similaire une structure relationnelle.

(2) est équivalent à (3) : il suffit de noter qu'un homomorphisme de $\mathcal{A}$ vers $\mathcal{B}$ correspond exactement à un assignement de valeurs du modèle $\mathcal{B}$ aux variables de $\varphi_{\mathcal{A}}$ satisfaisant cette dernière.

Pour (1) implique (2) : il suffit d'observer que $\mathcal{B} \models \varphi_{\mathcal{B}}$ (via l'assignement trivial qui à une variable de $\varphi_{\mathcal{B}}$ associe la valeur correspondante de $\mathcal{B}$) et l'inclusion (1) pour la base de donnée $\mathcal{D} := \mathcal{B}$ implique (2).

Pour (3) implique (1). On suppose que h est un homomorphisme de $\mathcal{A}$ dans $\mathcal{B}$. Soit $\mathcal{D}$ une structure relationnelle quelconque (vue comme une base de donnée) telle que $\mathcal{D} \models \varphi_{\mathcal{B}}$. L'équivalence entre (2) et (3) démontrée ci-dessus montre qu'on peut reformuler ceci en terme d'existence d'un homomorphisme w de $\mathcal{B}$ vers $\mathcal{D}$. Par composition, $w \circ h$ est un homomorphisme de $\mathcal{A}$ vers $\mathcal{D}$. En utilisant de nouveau l'équivalence entre (2) et (3) (mais dans l'autre direction), cet homomorphisme est un assignement certifiant que $\mathcal{D} \models \varphi_{\mathcal{A}}$. On a donc bien montré (1).

Inter Réduction entre Sat et Generalized Sat

1. Réduisez k-Sat à Generalized Sat.

Pour chaque type de clause C, on introduit une relation R_C listant tous les tuples autorisées. Par exemple, on associe $\{0,1\}^3 \setminus \{(0,0,0)\}$ à la clause $x \vee y \vee z$. En général, on associe à la clause $C = \ell_1 \vee \ell_2 \vee \ldots \ell_k$ la relation $R_C := \{0,1\}^k \setminus \{t\}$, où t est le seul assignement interdit par la clause C. C'est-à-dire que $t[i] = 0$ si ℓ_i est un littéral positif et $t[i] = 1$ si ℓ_i est un littéral négatif. Comme k est une constante, la relation R_C a taille constante $2^k - 1$.

Par ce codage, on réduit naturellement l'instance de k-Sat à l'instance de Generalized Sat ayant le même ensemble de variable, une clause C portant sur des variables $x_1, x_2, \ldots, x_k$ étant remplacée par la contrainte $R_C(x_1, x_2, \ldots, x_k)$.

2. On considère le cas de Generalized Sat restreint à une seule relation R d'arité r. Réduisez ce problème à Sat.

Pour l'autre direction, il peut être nécessaire d'utiliser plusieurs clauses pour une relation. Soit No-Good$(R) := \{0,1\}^r \setminus R$ l'ensemble des tuples interdits par la contrainte/relation R. Pour chaque tuple t de No-Good(R) on construit une r-clause $C_t = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_r$ où ℓ_i est un littéral positif si $t[i] = 0$ et ℓ_i est un littéral négatif si $t[i] = 1$. On simule la contrainte codée par R par la collection de clauses suivante.

$$\bigwedge_{t \in \text{No-Good}(R)} C_t$$

[3](vocabulaire) En BdD, on parle de la requête conjonctive canonique de $\mathcal{A}$ et pour l'autre direction de la base de donnée canonique d'une requête conjonctive.

La réduction est polynomiale si on suppose que l'arité de la relation r est une constante (on pourrait avoir une relation ayant un seul tuple autorisée, et notre réduction associerait à R de taille r une formule de taille $r(2^r - 1)$).

Clôture des modèles de certaines clauses particulières

1. Soit C une clause de Horn à r variables. Soit R_C l'ensemble des assignements (un assignement est vu comme un tuple de r valeurs) t satisfaisant la clause C. Montrez que R_C est clos par $\wedge$: c'est-à-dire que pour tout t_1 et t_2 de R_C le tuple $t_1 \wedge t_2$ (on applique $\wedge$ composante par composante) est dans R_C.

Si C ne contient que des littéraux négatifs, alors un tuple appartient à R_C ssi il contient un zéro. La relation correspondante R_C est donc clairement clos puisque la conjonction de deux tels tuples contiendra également un 0.

Si C contient un littéral positif x, alors le seul assignement n'appartenant pas à R_C sera de la forme : variables correspondant à un littéral négatif à 1 et x à 0, soit $(1, 1, \ldots, 1, 0)$. Si $t = (1, 1, \ldots, 1, 0)$ était la conjonction de t_1 et t_2 deux tuples appartenant à R_C, il faudrait que $t_1[y] = t_2[y] = t[y] = 1$ pour toute variable $y \neq x$ et que $t1[x] = 0$ ou $t_2[x] = 0$. C'est-à-dire que l'un de t_1 ou t_2 serait égal à t qui pourtant ne satisfait pas C. La relation R_C est donc bien close par $\wedge$.

2. Montrez comment simuler la relation $R = \{000, 100, 010, 011, 111\}$ par plusieurs clauses de Horn.

No-Good$(R) := \{001, 101, 110\}$. Les deux derniers tuples peuvent être simulés par des clauses de Horn (ils ont au plus un 0, soit au plus un littéral positif). Seul le premier correspond à un tuple qui n'est pas de Horn : $x \vee y \vee \neg z$. Si on enlève l'un des deux littéral positif x ou y de cette clause, on obtient en fait un renforcement de la contrainte qui nous voulions imposer (à savoir que 001 ne soit pas autorisé) : les tuples de R tels que $z = 0$ resteront satisfait mais ceux tels que $z = 1$ pourraient être non satisfait. Par exemple, si on décidait de remplacer la clause par $x \vee \neg z$, on ne satisferait plus le tuple (011) pourtant autorisé. Il y a 2 tuples de R_C tels que $z = 1$: (111) et (011). Tous deux satisfont $y = 1$. Il suffit donc de remplacer la clause qui n'est pas de Horn $x \vee y \vee \neg z$ par $y \vee \neg z$.

On simule la relation R par les trois clauses de Horn :

$$(y \vee \neg z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z).$$

3. Montrez comment simuler une relation R close par $\wedge$ par plusieurs clauses de Horn.

On procède comme ci-dessus et on code chaque tuple t de No-Good(R) qui contient au plus un 0 par une clause de Horn. Il reste à gérer le cas des tuples ayant plusieurs 0. Soit C la clause (qui n'est pas de Horn) correspondant à un tel tuple. Comme ci-dessus, on ne va conserver qu'un littéral positif de C, ce qui correspond à renforcer la contrainte imposée par la clause C. On ignore les tuples de R satisfaisant C à cause d'un littéral négatif et on se penche sur les autres assignements : soit R^+ l'ensemble de ces tuples, c'est-à-dire satisfaisant $t[y] = 1$ si $\neg y$ est un littéral (négatif) quelconque de C et $t[x] = 1$ pour au moins une variable x telle que x soit un littéral positif de C. Puisque R est clos par $\wedge$, la conjonction t de deux tuples t_1 et t_2 de R^+ appartient à R, et même à R^+ puisque $t[y] = t_1[y] \wedge t_2[y] = 1 \wedge 1 = 1$ lorsque $\neg y$ est un littéral négatif de C. Soit t^+ le tuple minimum de R^+. On choisit une coordonnées x telle que x soit un littéral positif de C et $t^+[x] = 1$. On enlève tous les autres littéraux positifs de la clause C pour obtenir une clause de Horn C'. Le choix de conserver la variable x implique que C' satisfait tous les tuples de R (ceux de $R \setminus R^+$ grâce à un littéral négatif, ceux de R^+ car ils satisfont tous $t[x] = 1$).

4. Donnez un algorithme polynomial pour une restriction de Generalized Sat à des relations de Horn (indications : définir une généralisation de la notion de cohérence d'arc et utiliser la clôture par $\wedge$ pour guider le choix d'une solution particulière lorsque l'instance est cohérente).

On généralise la notion de la cohérence d'arc au cas non binaire.

On supposera que chaque variable x est équipée d'un domaine D_x. On dira que l'instance est 2-*cohérente* si, pour toute variable x, pour toute valeur $d \in D_x$ et toute contrainte R portant (entre autres variables) sur x, il existe un tuple supportant la valeur d : c'est-à-dire qu'il existe un tuple t de R satisfaisant $t[x] = d$.

Il est clair que si ce n'était pas le cas, on pourrait enlever la valeur d du domaine D_x. C'est le principe de l'algorithme suivant (que l'on peut implanter de manière plus efficace comme ac).

Gac (Generalized Arc-Consistency)
Tant que les domaines des variables ne sont pas stables
    Pour une variable x, une valeur $d \in D_x$, une contrainte R portant sur x
    Si $\{t \in R$ tel que $t[x] = d\} = \emptyset$ enlever d de D_x

Si Gac vide le domaine d'une instance quelconque, on peut répondre insatisfait. Si ce n'est pas le cas – pour un langage de contrainte de type Horn – on peut répondre satisfait.

En effet, on peut démontrer que l'assignement t' consistant à choisir pour une variable x, la valeur minimale (pour $\wedge$) de D_x est une solution. Soit R une contrainte quelconque. Pour chaque variable x sur laquelle R porte, par cohérence il existe un tuple t de R tel que $t[x] = d$, où d est la valeur minimale de D_x. En prenant la conjonction de

ces tuples, on obtient un tuple de R (puisque le langage est de type Horn) compatible avec l'assignement t'.

5. Soit C une 2-clause et R_C la relation binaire des assignements satisfaisant cette clause C. Montrez que R_C est close par l'opération de majorité m. Cette opération ternaire retourne l'argument le plus fréquent : e.g. $m(0,0,0) = m(0,0,1) = m(1,0,0) = m(0,1,0) = 0$.

Chaque relation correspondant à une 2-clause contient 3 tuples. Notons que $m(t,t,t) = t$ et $m(t_1,t_1,t_2) = t_1$ et que l'ordre des tuples n'influence pas le résultat de m. Il suffit donc de tester la valeur de $m(t_1,t_2,t_3)$ pour $R = \{t_1,t_2,t_3\}$.

Par exemple pour la 2-clause $x \vee y$ qui correspond à la relation $\{0,1\}^2 \setminus \{(00)\}$ on a $m(01,10,11) = 11$. Les autres cas sont similaires.

6. Soit R une relation close par l'opération m. Réduisez la restriction de Generalized Sat au langage de contrainte $\{R\}$ à 2-Sat (indication : remplacer la contrainte R par toutes les contraintes induites par les projections sur deux coordonnées).

Supposons que l'instance obtenue en postant toutes les contraintes binaires induites par projection sur deux coordonnées d'une contrainte de l'instance soit satisfaite. Nous avons en fait relaxé les contraintes et en général une solution de l'instance initiale sera solution de ce système projeté. Nous allons montrer l'inclusion inverse en utilisant le fait que R est close par l'opération de majorité m.

Prenons un ordre arbitraire sur les variables $x_1, x_2, x_3, \ldots$ et supposons que $R(x_1, x_2, x_3, \ldots)$ soit une contrainte de l'instance.

- Puisque t' satisfait la contrainte binaire induite sur les variables x_1 et x_2, il existe un tuple $t_{1,2}$ de R tel que $t_{1,2}[x_1] = t'[x_1]$ et $t_{1,2}[x_2] = t'[x_2]$.
- De même, il existe un tuple $t_{1,3}$ de R tel que $t_{1,3}[x_1] = t'[x_1]$ et $t_{1,3}[x_3] = t'[x_3]$.
- De même, il existe un tuple $t_{2,3}$ de R tel que $t_{2,3}[x_2] = t'[x_2]$ et $t_{2,3}[x_3] = t'[x_3]$.

On considère le tuple $t_{1,2,3} = m(t_1,t_2,t_3)$. Puisque R est clos par m ce tuple appartient à R et d'après la définition de la fonction majorité ce tuple satisfait $t_{1,2,3}[x_1] = t'[x_1]$, $t_{1,2,3}[x_2] = t'[x_2]$ et $t_{1,2,3}[x_3] = t'[x_3]$. En procédant de manière similaire, on peut obtenir des tuples $t_{2,3,4}$ et $t_{1,3,4}$ tels que $t_{1,2,3,4} = m(t_{1,2,3}, t_{2,3,4}, t_{1,3,4})$ et ainsi de suite jusqu'à obtention d'un tuple qui soit compatible avec l'assignement t'.

Ce raisonnement montre qu'on peu se réduire à une instance booléenne n'ayant que des relations binaires, ce qu'on peut coder par une instance 2-Sat (une restriction polynomiale de sat). On peut aussi proposer un méthode alternative consistant à établir la 3-cohérence.

On suppose l'instance équipée de *domaines doubles* $D_{x,y}$ pour toute paire de variables $\{x, y\}$. On dira que l'instance est 3-*cohérente* si aucun domaine double n'est vide et que pour tout domaine double et toute valeur $(d, d') \in D_{x,y}$, pour toute contrainte R portant à la fois sur x et y il existe un tuple $t_{d,d'}$ de R tel que $t_{d,d'}[x] = d$ et $t_{d,d'}[y] = d'$.

Il est clair que la 3-cohérence peut s'établir en temps polynomial. Si l'algorithme de cohérence échoue on rejette l'instance (comme on est en droit de le faire pour une instance quelconque) et si l'algorithme s'arrête avec des domaines doubles tous non vides, on peut l'accepter (puisqu'on peut utiliser la méthode expliquée ci-dessus pour construire une solution).

Expressivité d'un langage

1. Simulez la relation booléenne $x \neq y$ par des 2-clauses. En déduire une réduction de 2-col à 2-Sat.

La relation booléenne $x \neq y$ correspond à $x \vee y \wedge \neg x \vee \neg y$. On peut donc réduire 2-Col à 2-Sat. En effet, si on veut savoir si un graphe $\mathcal{G}$ est 2-colorable, cela correspond aux contrainte suivantes pour des variables booléennes :

$$\bigwedge_{(x,y)\in E^{\mathcal{G}}} x \neq y.$$

On peut donc en déduire que $\mathcal{G}$ est 2-colorable ssi la formule propositionnelle 2-Sat suivante est satisfaisable :

$$\bigwedge_{(x,y)\in E^{\mathcal{G}}} x \vee y \wedge \neg x \vee \neg y.$$

Cette formule a autant de variables que de sommets du graphe et exactement 2 clauses par arête du graphe. On a donc donné une réduction linéaire de 2-Col à 2-Sat.

2. Donnez une réduction de K_5-Col à C_5-Col. Quelle est la complexité de C_5-Col ?

On remplace chaque arête du graphe $\mathcal{G}$ dont on veut savoir si il a nombre chromatique 5 ou moins par un chemin de 3 arêtes. Soit $\mathcal{G}^\star$ le graphe obtenu en subdivisant ainsi les arêtes de $\mathcal{G}$. Notez que par un homomorphisme depuis le chemin à 3 arêtes $\mathcal{P}_3$ vers C_5, les extrémités du chemin $\mathcal{P}_3$ peuvent prendre toutes les combinaisons de sommets distincts mais ne peuvent pas être égales. On peut donc en déduire qu'il y a un homomorphisme de $\mathcal{G}$ à $\mathcal{K}_5$ ssi il y a en un de $\mathcal{G}^\star$ à $\mathcal{C}_5$, ce qu'on note :

$$\mathcal{G} \to \mathcal{K}_5 \text{ si, et seulement si, } \mathcal{G}^\star \to \mathcal{C}_5.$$

Par ailleurs, la construction de $\mathcal{G}^\star$ à partir de $\mathcal{G}$ est clairement polynomiale.

Le problème $\mathcal{C}_5$-COL est dans NP et comme $\mathcal{K}_5$-COL est NP-complet s'y réduit, on peut conclure que $\mathcal{C}_5$-COL est NP-complet.

3. Montrez que si H était un graphe polynomial non biparti, ayant un nombre de sommets minimal puis parmi ceux-ci un nombre d'arêtes maximal alors H contiendrait nécessairement un triangle.

On procède par l'absurde. Supposons que $\mathcal{H}$ ne contienne pas de triangle. Puisque $\mathcal{H}$ n'est pas biparti, il contient un cycle impair. Prenons $\mathcal{C}_k$ un cycle impair de $\mathcal{H}$ avec $k \geq 5$ minimal. Soit ${}^\star\mathcal{H}$ le graphe ayant le même ensemble de sommets que $\mathcal{H}$ tels que deux sommets x et y soient adjacents ssi il existe un homomorphisme depuis $\mathcal{P}_3$ vers $\mathcal{H}$ envoyant les extrémités du chemin $\mathcal{P}_3$ vers x et y. Notez que $\mathcal{H}$ est un sous-graphe de ${}^\star\mathcal{H}$; et que ce graphe ne contient pas de boucle (car $\mathcal{H}$ n'a pas de triangle). Notez aussi la réduction polynomiale de ${}^\star\mathcal{H}$-COL à $\mathcal{H}$-COL : on réduit un graphe $\mathcal{G}$ au graphe $\mathcal{G}^\star$ obtenue en subdivisant chaque arête de $\mathcal{G}$ comme dans la question précédente. On utilise le fait que :

$$\mathcal{G} \to^\star \mathcal{H} \text{ si, et seulement si, } \mathcal{G}^\star \to \mathcal{H}.$$

Ainsi ${}^\star\mathcal{H}$ est par construction un graphe polynomial (réduction en temps polynomial à un autre graphe polynomial) non biparti (il contient $\mathcal{H}$ qui n'est pas biparti) ayant le même nombre de sommets que $\mathcal{H}$. Par maximalité du nombre d'arêtes de $\mathcal{H}$, les deux graphes sont forcément identiques.

Or comme $k \geq 5$, on peut trouver deux sommets de $\mathcal{C}_k$ à distance 3 sur ce cycle ce qui implique que l'arête reliant ces deux sommets soit présente dans ${}^\star\mathcal{H}$ et donc dans les deux graphes. Il y a donc un triangle dans $\mathcal{H}$. Contradiction.

4. Soit $R_1(x_1, x_2, x_3)$ le modèle de $(x_1 \vee \neg x_2 \vee \neg x_3)$ et $R_2(x_3)$ celui de x_3. Calculez la relation R' définie par

$$\exists x_3 R_1(x_1, x_2, x_3) \wedge R_2(x_3).$$

$R_1 := \{0,1\}^3 \setminus \{(011)\} = \{000, 001, 010, 100, 101, 110, 111\}$ et $R_3 := \{1\}$.
$R_1(x_1, x_2, x_3) \wedge R_2(x_3)$ correspond à la relation $\{001, 101, 111\}$.
La relation R' est la projection de $\{001, 101, 111\}$ sur les deux premières coordonnées, soit

$$R' := \{00, 01, 11\}.$$

5. Exprimez la réduction de K_5 à C_5 comme une p.p. définition. On rappelle qu'une *p.p. définition* est déterminée par une formule du premier-ordre utilisant les symboles de relations de Γ, la conjonction $\wedge$, la quantification existentielle $\exists$ et l'égalité $=$.

$\varphi(x,y) := \exists x_2, x_3 E(x,x_2) \wedge E(x_2,x_3) \wedge E(x_3,y)$. Le graphe $\varphi(\mathcal{C}_5)$ est $\mathcal{K}_5$. La réduction définie par φ correspond à l'opération ${}^\star\mathcal{H}$ décrite ci-dessus.

Autour de notre preuve du théorème de Schaefer

1. Pour chacune des relations suivantes, indiquez si elles sont closes ou non pour chacune des opération suivantes c_0, c_1 (opérations constantes), $\wedge, \vee, l$ (opération ternaire : somme modulo 2 de ces arguments), m (majorité ternaire) et $\neg$ (unaire, complément).

$R_1 = \{000, 100, 010, 011, 111\}$ $R_2 = \{000, 111\}$
$R_3 = \{01, 11\}$ $R_4 = \{01, 10\}$
$R_5 = \{100, 010, 001\}$ $R_6 = \{001, 010, 011, 100, 101, 110\}$

	c_0	c_1	$\wedge$	$\vee$	l	m	$\neg$
R_1	✓	✓	✓				
R_2	✓	✓	✓	✓	✓	✓	✓
R_3		✓	✓	✓	✓	✓	
R_4					✓	✓	✓
R_5							
R_6							✓

2. Montrez que si une opération f préserve un ensemble de relations Γ, alors f préserve $[\Gamma]$, où $[\_]$ indique l'ensemble des relations qu'on peut exprimer par *p.p.* définitions à partir de celles de Γ. (indication : procédez par récurrence sur la structure de la formule p.p.).

Nous montrons que $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}([\Gamma])$ (notez que l'autre inclusion est triviale puisque $\Gamma \subseteq [\Gamma]$, on peut donc conclure que $\mathrm{Pol}(\Gamma) = \mathrm{Pol}([\Gamma])$).

Nous procédons par induction sur la complexité d'une formules primitive positive φ.

- Si φ est une formule atomique, alors elle est soit de la forme $x = y$ (la relation d'égalité est préservée par n'importe quel fonction), soit de la forme $R(x_1, \ldots, x_m)$, où R est une relation de Γ et certaines variables peuvent être identiques. Si f (d'arité n) préserve R (d'arité m) et $R' := \varphi(R)$, on montre facilement que f préserve R'. Si les variables sont toutes distinctes alors $R = R'$ et c'est clair. Supposons que certaines variables soient identiques, disons $x_1 = x_2$ distinctes de toutes les autres pour simplifier l'exposition. Soient $t'_1, \ldots, t'_n$ des tuples de R'. Ce sont aussi des tuples de R, donc $f(t'_1, \ldots, t'_n)$ appartient à R. Puisqu'on applique f composante par composante alors $f(t'_1, \ldots, t'_n)$ appartient à R'. En effet, puisque

$t'_1[x_1] = t'_1[x_2], \ldots, t'_n[x_1] = t'_n[x_2]$ on a $f(t'_1[x_1], \ldots, t'_n[x_1]) = f(t'_1[x_1], \ldots, t'_n[x_2])$ soit $f(t'_1, \ldots, t'_n)[x_1] = f(t'_1, \ldots, t'_n)[x_2]$.

- Supposons que f (d'arité n) préserve R (d'arité m) et que φ soit de la forme $\exists x_1 R(x_1, \ldots, x_m)$. Soit $R' := \varphi(R)$. Notez que R' est la projection sur les $m-1$ dernières coordonnées de R. Soient $t'_1, \ldots, t'_n$ des tuples de R'. Il existe des tuples $t_1, \ldots, t_n$ dans R étendant les tuples $t'_1, \ldots, t'_n$. Puisque f préserve R, le tuple $f(t_1, \ldots, t_n)$ appartient à R. Sa projection sur les dernières coordonnées appartient à R' et est égale à $f(t'_1, \ldots, t'_n)$. On a donc bien montré que f préserve R'.
- Supposons que f (d'arité n) préserve R (d'arité m) et R' (d'arité m') et que φ soit de la forme $R(x_1, \ldots, x_{m_1}) \wedge R'(x'_1, \ldots, x'_{m_2})$. On peut supposer sans perte de généralité que les variables x_i sont distinctes deux à deux ainsi que les variables x'_i (voir cas atomique similaire) mais que des variables peuvent être communes e.g. x_i et x'_j. Soit $R'' := \varphi(R, R')$. Soient $t''_1, \ldots, t''_n$ des tuples de R''. Il existe donc des tuples $t_1, \ldots, t_n$ dans R (projection des précédents sur les coordonnées $x_1, \ldots, x_n$) et des tuples $t'_1, \ldots, t'_n$ dans R' (projection des $t''_1, \ldots, t''_n$ sur les coordonnées $x'_1, \ldots, x'_n$). Puisque f préserve à la fois R et R' par induction, le tuple $f(t_1, \ldots, t_n)$ est dans R et le tuple $f(t'_1, \ldots, t'_n)$ est dans R'. Notez que puisque l'application de f se fait composante par composante, si des variables x_i et x'_j étaient identiques on aurait $t_1[x_i] = t'_1[x'_j], \ldots, t_n[x_i] = t'_n[x'_j]$ et donc on aurait $f(t_1[x_i], \ldots, t_n[x_i]) = f(t_1, \ldots, t_n)[x_i] = f(t'_1, \ldots, t'_n)[x'_j] = f(t'_1[x'_j], \ldots, t'_n[x'_j])$. Le tuple $f(t''_1, \ldots, t''_n)$ est donc bien dans R'' puisque c'est la combinaison adéquate des deux tuples $f(t_1, \ldots, t_n)$ et $f(t'_1, \ldots, t'_n)$.

3. Démontrez le théorème de Jeavons (pour simplifier, on utilisera des réductions polynomiales : on peut en fait utiliser des réductions logspace).

   Théorème 2 (Jeavons '98)**.** *Si* Γ_1 *et* Γ_2 *sont des langages de contraintes tels que* $[\Gamma_1] \subseteq [\Gamma_2]$ *alors* $\text{SAT}(\Gamma_1)$ *se réduit à* $\text{SAT}(\Gamma_2)$.

   La preuve est esquissée dans le script.

4. Démontrez que Not-All-Equal-3-Sat est NP-complet.
   Indications. On prendra initialement une variante du problème autorisant les littéraux positifs et négatifs. Faites la réduction en deux temps depuis 3-Sat via le cousin du problème considéré Not-All-Equal-4-Sat. L'idée consiste à ajouter une variable spéciale θ pour toute la formule : un assignement dans lequel z prend la même valeur que θ étant traduit en un assignement où z est faux. La réduction de Not-All-Equal-4-Sat à Not-All-Equal-3-Sat est similaire à celle de k-Sat à 3-Sat.

5. Démontrez qu'on ne peut pas simuler (au sens de la p.p. définissabilité) les relations de 3-Sat avec la relation de Not-All-Equal-3-Sat.

Vois transparents du cours.

6. Démontrez sans faire de réduction explicite que 1-In-3-Sat (vu comme le CSP booléen avec le langage $\{R_5\}$) est NP-complet par réduction depuis Not-All-Equal-3-Sat (vu comme le CSP booléen avec le langage $\{R_6\}$).

On utilise le théorème de Jeavons. On a déjà observé que $\mathrm{Pol}(R_6) = \langle \neg \rangle$ (puisque R_6 est préservée par $\neg$ et aucune autre fonction générant un clone minimal) et que R_5 n'est invariant que par les projections (n'étant présevée par aucune fonction générant un clone minimal). C'est-à-dire que $\mathrm{Pol}(R_6) \supseteq \mathrm{Pol}(R_5)$. Donc $[R_6] = \mathrm{Inv}\Big(\mathrm{Pol}(R_6)\Big) \subseteq \mathrm{Inv}\Big(\mathrm{Pol}(R_5)\Big) = [R_5]$ par la correspondance de Galois. On applique le théorème de Jeavons et on peut déduire que $\textsc{Sat}(R_6)$ se réduit à $\textsc{Sat}(R_5)$. Comme le premier est le problème NP-complet Not-All-Equal-3-Sat, on peut en déduire que le second –1-In-3-Sat– est aussi NP-complet.

Utilisation du théorème de Schaefer

1. Quelle est la complexité de Generalized Sat pour les langages suivants ?

$\Gamma_1 = \{R_1, R_2, R_3\}$ $\qquad$ $\Gamma_2 = \{R_1, R_4\}$

$\Gamma_3 = \{R_2, R_4\}$ $\qquad$ $\Gamma_4 = \{R_3, R_6\}$

- $c_1 \in \mathrm{Pol}(\Gamma_1)$ donc Γ_1 est 1-valide (complexité triviale).
- Pas de polymorphisme autre que les projections pour Γ_2 : NP-complet.
- $m \in \mathrm{Pol}(\Gamma_3)$ donc Γ_3 est polynomial.
- Γ_4 contient R_6 qui est NP-complet donc il est NP-complet. On peut aussi le vérifier par le fait que le langage Γ_4 n'a pas de polymorphisme autre que les projections. Ceci signifie que $[R_3, R_6]$ permet de simuler n'importe quelle contrainte (donc en particulier R_5).

2. Montrez que pour tous les cas polynomiaux du théorème de Schaefer, on peut également trouver une solution en temps polynomial si elle existe (indication : on utilisera les algos du théorème de Schaefer comme des boîtes noires).

Pour le cas 0-valide ou 1-valide, il suffit de vérifier que les relations sont non-vides puis de donner la solution constante.

Pour les autres langage de contrainte polynomiaux, il suffit de noter que l'on peut ajouter les relations constantes $\{0\}$ et $\{1\}$ sans en changer la complexité : ces deux

relations sont préservées par $\wedge, \vee, m$ et l. L'algorithme qui trouve une solution si elle existe va appeler un nombre polynomial de fois l'algorithme de décision.

Algorithme de recherche d'une solution
On appelle l'algo polynomial de décision pour vérifier si il existe une solution, si c'est le cas on prend un ordre arbitraire sur les n variables et on répéte l'opération suivante :

- prendre la variable suivante x et postez la contrainte $x = 0$
- appeler l'algo polynomial de décision
- si ce dernier répond non, on enlève la contrainte $x = 0$ et on poste la contrainte $x = 1$

On obtient ainsi une solution (si elle existe) en faisant au plus $2n + 1$ appels à un algo polynomial.

3. Quelle est la complexité du model checking pour le fragment $\exists, \wedge, \neq$ de la logique du premier ordre lorsqu'on prend le modèle comme paramètre ? Plus précisément, donner une classification complète de la complexité du problème de décision suivant selon le choix de $\mathcal{B}$.

$\{\exists, \wedge, \neq\}$-FO($\mathcal{B}$)

- **paramètre** : une structure $\mathcal{B}$.
- **entrée** : une formule φ du premier ordre utilisant seulement les symboles $\exists, \wedge, \neq$ et les symboles de relation de $\mathcal{B}$.
- **question** : est-ce-que $\mathcal{B} \models \varphi$?

Si $\mathcal{B}$ a $k = 3$ éléments ou plus, on peut réduire k-Col qui est NP-complet à notre problème en utilisant $\neq$.

Si $\mathcal{B}$ a $k = 1$ élément, alors le problème devient facile à résoudre : on enlève les variables et on les remplace par cet élément. Il ne reste plus qu'à vérifier que cet unique assignement possible est solution ou pas (en temps linéaire).

Le cas intéressant : lorsque $k = 2$, on se retrouve avec une variante de Sat(Γ) où Γ est le langage de contrainte booléen qui se compose des relations de $\mathcal{B}$ est de la relation boolénne $x \neq y$ soit $\{01, 10\}$. On applique le théorème de Schaefer. La relation $\{01, 10\}$ est close par $\wedge, \vee, l$ et m. Donc si les relations de $\mathcal{B}$ sont closes pour $\wedge, \vee, l$ ou m alors le problème sera polynomial sinon il sera NP-complet.

Classification de la complexité.

- Polynomial si $\mathcal{B}$ a un seul élément
- Polynomial si $\mathcal{B}$ a deux éléments et que $\mathcal{B}$ est Horn, dual-Horn, affine ou bijonctif
- NP-complet sinon.

Definitions and notation for some decision problems

Graph Colouring and variants

k-Col
- **parameter**: $k \geq 1$.
- **input**: an undirected graph G.
- **question**: is G k-colorable? i.e. can the vertices of G be assigned a color from $\{1, 2, \ldots, k\}$ such that adjacent vertices have different colours?

H-Col
- **parameter**: an undirected graph H.
- **input**: an undirected graph G.
- **question**: is there a homomorphism from G to H?

Sat and variants

A *literal* is a positive x or negative occurence $\neg x$ of a propositional variable x. A *clause* is a disjunction of literals. A propositional formula is said to be in *conjunctive normal form* (CNF) if it is written as a conjunction of clauses.

Propositional Satisfiability (Sat)
- **input**: a propositional formula φ in *conjunctive normal form*.
- **question**: is φ satisfiable? (i.e. does there exist a truth assignment such that every clause of φ has at least one true literal)?

k-Sat
- **parameter**: $k \geq 1$.
- **input**: a propositional formula φ in CNF with at most k literals per clause.
- **question**: is φ satisfiable?

Horn-Sat
- **input**: a propositional formula φ in CNF with at most one positive literal per clause.
- **question**: is φ satisfiable?

It is common in this context to simply write clauses in their implicative forms:

$$\begin{array}{ll} \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_k \vee p_{k+1} & \text{as } p_1 \wedge p_2 \wedge \ldots \wedge p_k \implies p_{k+1} \\ \neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_k & \text{as } p_1 \wedge p_2 \wedge \ldots \wedge p_k \implies false. \end{array}$$

1-In-3-Sat
- **input**: a collection of 3-clauses (e.g. $C(x, y, z)$).
- **question**: is there a truth assignment to the variables such that exactly one is true in each clause?

Not-All-Equal-3-Sat
- **input**: a collection of 3-clauses (e.g. $C(x, y, z)$).
- **question**: is there a truth assignment to the variables such that the variables in each clause are neither all true nor all false?

Generalized-Sat (Sat(Γ))
- **parameter**: a (possibly non finite) set of Boolean relations Γ.
- **input**: a conjunction of positive atoms $R(\bar{x})$, where R is a symbol occurring in Γ.
- **question**: is φ satisfiable?

Csp and variants

We adopt here the definition of Constraint Satisfaction Problem as a homomorphism problem between two relational structures.

Constraint Satisfaction Problem (csp)
- **input**: two relational structures over the same signature $\mathcal{A}$ and $\mathcal{B}$.
- **question**: is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?

Non-Uniform Constraint Satisfaction Problem (csp($\mathcal{B}$))
- **parameter**: a relational structure $\mathcal{B}$ (the *constraint language* a.k.a. the *template*).
- **input**: a relational structure over the same signature $\mathcal{A}$.
- **question**: is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?

Uniform Constraint Satisfaction Problem (csp($\mathscr{C}$, _))
- **parameter**: a fixed relational signature σ and a class of relational structures $\mathscr{C}$ over this signature.
- **input**: two relational structures over the same signature $\mathcal{A}$ and $\mathcal{B}$ where $\mathcal{A}$ belongs to $\mathscr{C}$.
- **question**: is there a homomorphism from $\mathcal{A}$ to $\mathcal{B}$?