Sous-classes de formules propositionnelles à satisfiabilité tractable

Nous nous plaçons pour ce problème en logique propositionnelle classique.

On définit la taille d'un littéral à 1, la taille d'une clause au nombre de littéraux qu'elle contient, et la taille d'un ensemble de clauses à la somme des tailles des clauses de l'ensemble.

On rappelle que le problème SAT est la question de savoir, étant donnée une formule propositionnelle Φ en forme clausale, si elle est satisfiable ou non. Ce problème est NP-complet : en particulier, il est conjecturé qu'il n'existe aucun algorithme en temps polynomial permettant de résoudre ce problème. (C'est-à-dire en temps $O(n^k)$ pour une certaine constante $k \geq 0$, quelle que soit la taille n de l'ensemble de clauses représentant Φ.)

Notre but dans ce problème est d'examiner quelques sous-classes de formules pour lesquelles on connaît des algorithmes en temps polynomial.

1. On appelle *2-clause* toute clause ayant au plus deux littéraux. Par exemple, $A \vee B$, $A \vee \neg B$, $\neg A \vee \neg B$ ou $\neg A$ sont des 2-clauses, mais $A \vee \neg B \vee C$ n'en est pas une. La restriction de SAT aux ensembles de 2-clauses est appelé 2-SAT.

   (a) Montrer que tout résolvant entre 2-clauses est une 2-clause.

   (b) On se donne un ensemble de 2-clauses S, de taille n. On pose $S_0 = S$, S_{n+1} l'ensemble des clauses de S_n plus tous les résolvants entre clauses de S_n, et $S_\infty = \bigcup_{n \in \mathbb{N}} S_n$. Montrer que le cardinal de S_∞ est borné par un polynôme en n.

   (c) Proposer une réalisation de l'algorithme de résolution propositionnelle qui termine sur tout ensemble S de 2-clauses, et en déduire que 2-SAT est soluble en temps polynomial. (On ne demande pas un algorithme efficace.)

   (d) L'algorithme de Davis-Putnam tourne-t-il en temps polynomial, exponentiel, ou autre sur les ensembles de 2-clauses? Commenter.

   (Pour information, une méthode plus efficace que la résolution, due à Even, Itai et Shamir, est fondée sur une construction de graphe : on recode d'abord les clauses unitaires L sous forme de $L \vee L$, de sorte à n'avoir que des clauses à deux littéraux; pour chaque variable A apparaissant dans S, on crée deux nœuds, A et $\neg A$; pour chaque clause $L \vee L'$, on ajoute un arc de $\neg L$ vers L' et un de $\neg L'$ vers L [par convention, on identifie $\neg\neg A$ avec A]; alors S est insatisfiable si et seulement s'il existe un chemin d'un atome A vers $\neg A$ dans le graphe.)

2. On appelle *clause de Horn* toute clause ayant au plus un littéral positif. Par exemple, A, $\neg B$, $\neg A \vee \neg B$, $\neg A \vee \neg B \vee C$ sont des clauses de Horn, mais $A \vee B$ et $\neg A \vee \neg B \vee C \vee D$ n'en sont pas.

   On appelle *but* toute clause n'ayant que des littéraux négatifs, par exemple $\neg A$ ou $\neg A \vee \neg B$. On appelle *fait* toute clause comprenant exactement un littéral positif, par exemple A.

On appelle HORN-SAT la restriction de SAT aux ensembles de clauses de Horn.

On se donne dorénavant un ensemble de clauses de Horn S.

(a) Montrer que si S ne contient pas de fait, alors S est satisfiable.

(b) Montrer que si S ne contient pas de but, alors S est satisfiable.

(c) Déduire de la question 2a un algorithme en temps polynomial décidant HORN-SAT. (On utilisera un argument inspiré de ceux prouvant la correction de l'algorithme de Davis-Putnam.)

(Pour information encore, l'algorithme préféré de décision, dû à Dowling, Gallier et Minoux, de HORN-SAT est basé sur une construction de graphe, représentant efficacement l'algorithme ci-dessus.)

3. Soit S un ensemble de clauses quelconques. On appelle *renommage* $R(S, \mathcal{A})$ de S par un ensemble $\mathcal{A}$ d'atomes l'ensemble de clauses obtenu en réécrivant A en $\neg A$ et $\neg A$ en A, pour tout $A \in \mathcal{A}$. Par exemple, si S contient $A \vee B$, $A \vee \neg B \vee C$, $\neg B \vee \neg A$, effectuer le renommage par $\{A\}$ produit l'ensemble $R(S, \{A\})$ contenant $\neg A \vee B$, $\neg A \vee \neg B \vee C$, $\neg B \vee A$. On remarquera que dans l'exemple, S n'est ni un ensemble de 2-clauses ni un ensemble de clauses de Horn, mais que $R(S, \{A\})$ est un ensemble de clauses de Horn.

On appelle *Horn-renommage* de S tout ensemble $\mathcal{A}$ de variables propositionnelles tel que $R(S, \mathcal{A})$ soit un ensemble de clauses de Horn. Un ensemble de clauses qui possède un Horn-renommage est appelé *Horn-renommable*. La restriction de SAT à de tels ensembles de clauses s'appelle HORN-REN-SAT.

(a) Soit $\mathcal{A}$ un renommage arbitraire. Montrer que S est satisfiable si et seulement si $R(S, \mathcal{A})$ l'est.

(b) Soit C une clause quelconque. Pour chaque variable A apparaissant dans C, on crée une nouvelle variable propositionnelle x_A. Décrire un ensemble de 2-clauses $F(C)$ sur les variables x_A, tel que les interprétations qui satisfont $F(C)$ sont exactement les interprétations I telles que $R(C, \{A \mid I(x_A) = \top\})$ est une clause de Horn. (On note $\top$ pour vrai.)

(c) Quelle est la taille de $F(C)$ en fonction de celle de C ? En déduire, en utilisant les résultats de la question 1, que savoir si un ensemble de clauses S est Horn-renommable est décidable en temps polynomial. Semble-t-il faisable en temps polynomial de trouver un renommage s'il en existe un ?

(d) Adapter l'algorithme de Davis-Putnam (dans l'esprit de la question 2c) en un algorithme en temps polynomial décidant la satisfiabilité des ensembles S de clauses Horn-renommables. (On supposera qu'on a vérifié à l'avance que S était effectivement Horn-renommable.)