Mines Informatique MP 2002 — Corrigé

Ce corrigé est proposé par Mathieu Giraud (ENS Lyon) ; il a été relu par Xavier Goaoc (ENS Cachan) et Sébastien Desreux (ENS Ulm).

L'épreuve se compose de trois parties (deux exercices et un problème) indépendantes et de longueurs largement inégales. Beaucoup de points du programme sont concernés, notamment la description d'algorithmes, leur compréhension et l'évaluation de leur complexité. Quelques preuves sont aussi demandées.

- Le premier exercice traite de logique des propositions en résolvant le problème de la satisfiabilité des formules de Horn. Il demande de résoudre de petits exemples, d'effectuer quelques preuves et de concevoir un algorithme polynomial.

- Le problème est plutôt long (l'énoncé recommande d'y consacrer les deux tiers du temps imparti). Il étudie divers aspects du problème dit du « sac à dos » dans ses variantes fractionnaire (première partie) et entière (seconde partie). Il met en jeu diverses techniques algorithmiques (dont la récursion) et comporte aussi quelques questions concrètes (exemples) et un peu de programmation.

- Enfin, l'exercice sur les automates vise à prouver que le langage des conjugués d'un langage régulier est lui aussi régulier. Il utilise pour cela des techniques usuelles sur les automates et les langages réguliers.

Chaque partie s'ouvre par une certaine quantité de notations qu'il est préférable de bien assimiler avant de se lancer dans la résolution, pourquoi pas en commençant par examiner un exemple. De plus, beaucoup de questions sont largement indépendantes et, comme le rappelle l'énoncé, « tout résultat fourni dans l'énoncé peut être utilisé, même s'il n'a pas été démontré ».

Remarquons enfin que la part de la programmation est assez faible (seulement trois questions dans le problème) ; ce corrigé propose à chaque fois des solutions en Caml et en Pascal.

Indications

Partie I

1.2 Utiliser les littéraux négatifs.

1.3 Simplifier une à une les clauses de H en distinguant différents cas selon la présence de p_k et de $\neg p_k$.

1.4 Réduire le nombre de clauses de H jusqu'à pouvoir décider de la satisfiabilité.

Partie II

2.1.a Vérifier tout d'abord que la solution proposée (x) est effectivement solution. La preuve de la maximalité est plus délicate ; prendre une solution quelconque et considérer les objets d'indices $a = \min\{i \in [\![0\,;\, n-1]\!] \mid y_i < 1\}$ et $b = \max\{i \in [\![0\,;\, n-1]\!] \mid y_i > 0\}$.

2.1.b Appliquer la méthode de la question 2.1.a en n'oubliant pas de vérifier que les données proposées respectent les conditions sur les u_i/p_i. On pourra commencer par déterminer i^*.

2.1.d Vérifier avec soin les cas où i^* vaut 0 ou n.

2.2.a La récursion de l'algorithme pourra porter sur l'ensemble à partitionner. À chaque étape, l'appel récursif concerne seulement une des deux moitiés de cet ensemble.

2.2.c Il n'est pas demandé dans cette question de détailler un nouvel algorithme mais simplement d'indiquer comment utiliser les résultats précédents.

2.3 Une solution au problème en 0–1 est aussi une solution du problème fractionnaire.

2.5 Le pire des cas de la méthode par séparation est atteint lorsque l'arbre de recherche est complet.

2.6.c Procéder de la même manière qu'à la question 2.6.b ; utiliser la question 2.3.

2.6.e Déterminer une évaluation de A à partir de celles des feuilles C, E, F et G calculées dans les quatre questions précédentes.

Partie III

3.1 À partir d'un automate reconnaissant L, construire un nouvel automate en ajoutant des nouveaux états i et t répondant aux conditions de normalisation et simulant le comportement de certaines transitions des états de I et de T.

3.2 Construire deux automates reconnaissant respectivement les mots f_1 et les mots f_2, puis les fusionner.

3.3 Exprimer $\mathrm{Conj}(\mathrm{L} \setminus \{1_{\mathrm{A}^*}\})$ en fonction des langages G_q puis utiliser le résultat de la question 3.2. Ne pas oublier de traiter le cas du mot 1_{A^*}.

1. Exercice de logique des propositions

> Lorsque nous donnons des valuations, nous écrivons directement l'égalité $p_k = \mathit{vrai}$ (alors que l'on peut aussi écrire $v(p_k) = \mathit{vrai}$).

1.1

i) $(\neg p_1 \vee p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_3)$ est satisfiable par la valuation $p_1 = p_2 = p_3 = \mathit{vrai}$;

ii) $(p_2) \wedge (\neg p_1 \vee \neg p_3) \wedge (\neg p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_4)$ n'est pas satisfiable car les clauses (p_2) et $(\neg p_2)$ ne peuvent être satisfaites simultanément ;

iii) $(p_2) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_3)$ n'est pas satisfiable. Supposons en effet que les deux premières clauses soient satisfaites. Alors la première clause implique $p_2 = \mathit{vrai}$, puis la deuxième clause nécessite $p_1 = \mathit{vrai}$, ce qui a pour conséquence que la troisième clause ne peut pas être satisfaite.

1.2 Supposons que H soit une formule sous forme normale conjonctive telle que chacune de ses clauses contienne au moins un littéral négatif. On considère alors la valuation qui associe à chaque variable propositionnelle la valeur *faux*. Puisque chaque clause contient au moins un littéral négatif, elle est satisfaite. Il s'ensuit :

Une telle formule H est satisfiable.

1.3 Écrivons la formule sous la forme $\mathrm{H} = \mathrm{C}_1 \wedge \mathrm{C}_2 \wedge \cdots \wedge \mathrm{C}_m$ et supposons que la clause C_r est restreinte au littéral p_k. On construit à partir de H une nouvelle formule H′ en omettant la clause C_r et en transformant chaque clause C_ℓ, $\ell \neq r$, de la manière suivante :

1. si le littéral p_k y apparaît, alors on enlève C_ℓ ;
2. si le littéral $\neg p_k$ y apparaît, alors on garde C_ℓ, mais en y supprimant $\neg p_k$;
3. dans les autres cas, on conserve C_ℓ.

Les clauses contenant à la fois p_k et $\neg p_k$ sont traitées par le cas 1. Après cette transformation, la formule H′ :

- est une formule de Horn (car on n'a ajouté aucun littéral positif) ;
- ne fait apparaître ni p_k ni $\neg p_k$ donc $\mathcal{V}_{\mathrm{H}'} \subseteq \mathcal{V}_{\mathrm{H}} \setminus \{p_k\}$.

> Il n'y a pas égalité entre $\mathcal{V}_{\mathrm{H}'}$ et $\mathcal{V}_{\mathrm{H}} \setminus \{p_k\}$ car d'autres variables ont pu disparaître lors de l'élimination de certaines clauses.

Il ne reste plus qu'à montrer que H′ est satisfiable si et seulement si H l'est.

- Supposons H satisfaite par une valuation v. La clause C_r de H, réduite à $\{p_k\}$, impose que p_k soit évalué à *vrai* dans v. Soit v' la restriction de v à $\mathcal{V}_{\mathrm{H}'}$. Cette valuation satisfait les clauses de H′ produites par le cas 3 (clauses inchangées) et celles produites par le cas 2 (puisque $p_k = \mathit{vrai}$). Ainsi la formule H′ est satisfaite par v'.

- Réciproquement, supposons H′ satisfaite par une certaine valuation v'. On étend alors la valuation v' en fixant $p_k = \textit{vrai}$, ce qui satisfait les clauses traitées par le cas 1. Finalement H est aussi satisfaite.

On a réduit H en une autre formule de Horn équivalente H′.

1.4 Utilisons le résultat de la question 1.3 pour diminuer le nombre de clauses de H ; il faut d'abord en vérifier les conditions :

- une de ses clauses doit être réduite à un littéral positif p_k (sinon, on sait d'après la question 1.2 que la formule est satisfiable) ;
- aucune autre clause ne doit être réduite à $\neg p_k$ (sinon la formule n'est pas satisfiable).

Ces deux conditions fournissent les deux cas d'arrêt de l'algorithme.

ALGORITHME $\mathcal{S}$: **Satisfiabilité d'une formule de Horn**
Entrée : une formule de Horn H.
Sortie : *vrai* si H est satisfiable, *faux* sinon.

Répéter

1. Parcourir les clauses de H jusqu'à trouver une clause C_r réduite à un seul littéral positif (p_k).
   Si une telle clause n'existe pas, renvoyer *vrai*.
2. Parcourir de nouveau les clauses de H en cherchant s'il existe une clause réduite au seul littéral négatif ($\neg p_k$).
   Si c'est le cas, renvoyer *faux*.
3. Réduire la formule H en suivant la méthode exposée à la question 1.3.

Fin Répéter

L'étape 1 permet aussi de reconnaître comme vraie la formule vide T.

Évaluons maintenant la complexité de cet algorithme. Les étapes 1 et 2 nécessitent d'examiner chaque clause, ce qui se fait pour chacune de ces étapes en $O(m)$ opérations élémentaires. L'étape 3 demande, dans le pire des cas, d'observer chaque littéral de chaque clause. Il y a n variables propositionnelles distinctes et m clauses, donc au plus $O(mn)$ clauses. L'étape 3 se fait donc en au plus $O(mn)$ opérations élémentaires.

Enfin, la boucle générale s'exécute au plus $m+1$ fois puisqu'au moins une clause est supprimée à chaque itération. L'algorithme total exige au plus $O((m+1)(m+mn))$ opérations.

Cet algorithme polynomial a une complexité en temps de $O(m^2 n)$ dans le pire des cas.

Ce résultat montre la particularité des formules de Horn. Le problème général de la satisfiabilité des formules logiques est en effet NP-complet : on ne connaît aucun algorithme polynomial pour le résoudre et on conjecture même qu'un tel algorithme n'existe pas.