Université d'Aix-Marseille - Master Informatique 1^{ere} année

UE Complexité - TD 3 - Machines de Turing, Logique des propositions et graphes

Préambule : Dans cette planche, nous abordons les Machines de Turing Déterministes, puis la logique des propositions (qui *a priori* est maintenant maîtrisée par tous) et en particulier la compréhension de ce qu'est le **problème SAT** qui est un problème central de la Théorie de la Complexité, puis nous regardons certains liens existant avec la notion de graphe.

Exercice 1. Retour sur les Machines de Turing Déterministes

Dans ce qui suit, nous considèrerons le modèle de Machines de Turing Déterministes vu en cours.

Question 1. Définissez un programme pour Machine de Turing Déterministe dont l'entrée contient des mots définis sur l'alphabet $\Sigma = \{a,b,c\}$ et qui accepte uniquement les mots "triés", c'est-à-dire les mots compoés de a, puis de b, et enfin de c. Cela revient à reconnaître le langage $\{a^n b^m c^p : n,m,p \geq 0\}$. Vous donnerez la complexité de votre programme.

Question 2. Définissez un programme pour Machine de Turing Déterministe qui reconnaît le langage $\{x^k y^k : k > 0\}$. Vous donnerez la complexité de votre programme.

Question 3. Donnez un algorithme qui prend en entrée un tableau de n caractères, qui ne peut contenir que les caractères "x", "y", ou "B", et qui vérifie si ce tableau contient un mot de la forme $x^k y^k$, où k est un entier non nul, c'est-à-dire si à partir de la première case du tableau, on a une séquence de k cases contenant le caractère "x", suivies de k cases contenant le caractère "y", suivi du caractère "B", s'il reste au moins une case dans le tableau (c'est-à-dire si $2k < n$). Vous donnerez la complexité de votre algorithme, en supposant que toutes les actions élémentaires comme le test, l'affectation, l'incrémentation, etc. sont réalisables en temps constant.

Exercice 2. Retour sur la logique des propositions

Dans ce qui suit, nous ne considèrerons que des formules de la logique des propositions exprimées sous forme normale conjonctive (appelées FNC voire aussi CNF qui est isssu de l'anglais). Leur formulation varie selon les auteurs, mais il est d'usage de considérer que dans une FNC, les clauses (disjonctions de littéraux) sont telles qu'il n'y a pas deux occurrences du même littéral et qu'il ne peut y avoir un littéral et son opposé dans une même clause.

Question 1. Un peu d'échauffement : pour se rappeler la logique des propositions

- Donner une FNC définie sur 4 variables et par 4 clauses d'arité respectivement 1 (i.e. avec 1 littéral), 2 (i.e. avec 2 littéraux), 3 (i.e. avec 3 littéraux) et 4 (i.e. avec 4 littéraux). Est-ce que cette formule est satisfaisable (donner un modèle dans ce cas) ?
- Donner une FNC définie sur 2 variables et par des clauses *binaires* (d'arité 2) et qui est incohérente (i.e. sans modèle ; on dit aussi *insatisfaisable*, *inconsistante* ou *antilogique*)
- Donner une FNC définie sur 3 variables et par des clauses binaires et qui est incohérente (bien évidemment, cette FNC en doit pas contenir toutes les clauses présentes dans la formule précdente)
- Est-il possible de définir une FNC ne contenant que des clauses binaires et qui soit tautologique (toute interprétation est un modèle) ? Justifier votre réponse. Que pensez-vous de cette question pour des FNC avec des clauses d'arité quelconque ?
- Donner une FNC définie sur 3 variables et des clauses d'arité 3 et qui admet exactement 4 modèles.

Question 2. Un peu de modélisation : le problème dit des "pigeons".

Il s'agit d'exprimer par une FNC un problème de pigeons et de pigeonnier. On considère n pigeons et 1 pigeonnier consitité de $n-1$ niches (cela s'appelle en fait un *boulin*). Dans une niche, il ne peut rentrer qu'un pigeon, et on voudrait que chaque pigeon soit dans une niche. Construisez la FNC exprimant ce problème pour 3 pigeons, et donc un pigeonnier de 2 boulins. Une aide : on peut exprimer ce problème par des variables propositionnelles de la forme p_{ij} sachant que l'interprétation à vrai de p_{ij} signifierait que le pigeon i est dans le boulin j. Ainsi, pour imposer qu'un pigeon i soit dans un des 2 boulins, on peut exprimer cela avec une clause de la forme $(p_{i1} \vee p_{i2})$ dans la FNC modélisant ce problème. En effet, satisfaire cette clause impose d'avoir soit p_{ij} vrai (signifiant que le pigeon i est dans le boulin 1), soit p_{i2} vrai (signifiant que le pigeon i est dans le boulin 2) D'autres clauses que celles de ce type doivent bien sûr être présentes dans la formule.

Question 3. Appliquer l'algorithme de Quine à la formule FNC de la question précédente. Vous dessinerez l'arborescence sémantique correspondante.

Question 4. Appliquer l'algorithme DPLL à la formule FNC de la question précédente. Vous dessinerez l'arborescence sémantique correspondante.

Exercice 3. Le problème SAT, ses extensions et quelques restrictions

Le problème de la satisfaisabilité en logique des propositions, appelé SAT, s'exprime par :

> **SAT**
> **Donnée** : Une formule F de la logique des propositions exprimée par une FNC.
> **Question** : La formule F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Avant tout, précisons que SAT est parfois défini avec des instances étant des formules quelconques de la logique des propositions (i.e. pas nécessairement des FNC). Cela étant, la restriction que nous donnons ici n'a pas d'incidence au niveau de l'étude de cette question. À partir de ce problème, nous allons exprimer ses extensions puis étudier certaines restrictions.

Question 1. Extensions de SAT.

1. Définir SAT-RECHERCHE, le problème de recherche associé à SAT.
2. Définir MAX-SAT, le problème d'optimisation associé à SAT. Ici, le critère d'optimisation à considérer est le nombre de clauses pouvant être satisfaites pour une formule F, que cette formule soit satisfaisable ou non.
3. Définir #SAT, le problème de dénombrement associé à SAT.
4. Définir ENUM-SAT, le problème d'énumération associé à SAT.

Question 2. Une CNF dont chaque clause possède exactement k littéraux (tous différents comme indiqué en préambule) est appelée formule *k-SAT*. Cela permet d'exprimer des instances de SAT sous forme uniforme et aussi de définir des variantes simplifiées de SAT dans les termes suivants :

> **k-SAT**
> **Donnée** : Une CNF F dont toutes les clauses ont k littéraux.
> **Question** : La formule F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Notons que cette formulation varie selon les auteurs, en ce sens que parfois on appelle *instances k-SAT*, les instances de SAT dont les clauses ont au plus k littéraux. Mais il est d'usage de considérer que k-SAT porte précisément sur les instances de SAT dont toutes les clauses ont exactement k littéraux. Pour le cas où toutes les clauses ont 2 littéraux, le problème est appelé 2-SAT. Il vous faut dans cette question proposer une méthode efficace (i.e. de complexité polynomiale) pour résoudre le probleme 2-SAT.

Question 3. La restrictions aux clauses de Horn.

Dans ce qui suit, nous nous intéressons à un autre type de restriction de SAT, appelée *Horn-SAT* et qui considère les CNF exprimées avec des clauses de Horn, à savoir des clauses qui ont au plus 1 littéral positif. À partir de cela, on peut définir le problème suivant :

> **Horn-SAT**
> **Donnée** : Une CNF F dont toutes les clauses sont des clauses de Horn.
> **Question** : La formules F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Vous devez traiter les 4 questions suivantes :

1. Que pensez-vous de la satisfaisabilité d'une instance de Horn-SAT quand aucune clause n'est d'arité 1 ?
2. Que pensez-vous de la satisfaisabilité d'une instance de Horn-SAT quand il peut y avoir des clauses d'arité 1 ?
3. Indiquez une méthode pour trouver un modèle d'une instance Horn-SAT, s'il en existe un.
4. Si le problème Inv-Horn-SAT considère la restriction de SAT aux instances de SAT pour lesquelles les formules sont exprimées par des clauses qui ont au plus 1 littéral négatif, que pensez-vous des réponses aux questions précédentes pour Inv-Horn-SAT ?

Exercice 4. Implémentation de l'interprétation d'une formules FNC.

Dans ce qui suit, nous allons considérer des formules de la logique des propositions exprimées sous forme normale conjonctive et dont l'arité des clauses est 3 (on considère donc des instances de 3-SAT). Ces formules sont traditionnellement représentées dans des fichiers (vous verrez cela plus en détail notamment en TP) dans lesquels une clause de la forme $(x_1 \vee \neg x_3 \vee x_4)$ sera exprimée par la suite de 3 nombres 1, -3 et 4 (l'indice de la variable est positif ou négatif selon la nature du littéral). On supposera ici qu'une FNC définie sur n variables et par m clauses ternaires est stockée dans un tableau à 1 dimension dont les m premières cases contiennent chacune une des m clauses, chaque clause étant représentée dans un tableau d'entiers à 1 dimension et de taille 3. Une interprétation des n variables propositionnelles est pour sa part représentée dans un tableau de booléens dont les n premières cases contiennent chacune l'interprétation de la variable dont l'indice est celui de la case du tableau : si `inter` est le tableau, alors `inter[i]` a pour valeur l'interprétation de la variable dont l'indice est i, c'est-à-dire x_i.

À partir de cette représentation des FNC et des interprétations, il faut donnez un algorithme qui prend en entrées une telle formule et une interprétation de ses variable et qui fournit en résultat l'évaluation de la formule, soit *vrai*, soit *faux*. Donnez une évaluation de sa complexité.

Exercice 5. Un problème classique de graphes et son expression en logique

Dans ce qui suit, nous nous intéressons au célèbre problème de la *coloration de graphes*. Nous rappelons qu'un graphe non-orienté peut être colorié avec k couleurs si on peut affecter une couleur parmi k possibles, à chacun de ses sommets, de telle sorte que deux sommets reliés par une arête aient des couleurs différentes. Nous allons nous limiter ici au cas d'une palette à 3 couleurs, ce qui permet de définir le problème suivant :

> **3-COLORATION**
> **Donnée** : Un graphe non-orienté $G = (S,A)$
> **Question** : Existe-t-il une coloration de G avec 3 couleurs ?

Question 1. Montrer comment une instance du problème 3-COLORATION peut être exprimée avec une formule de la logique des propositions exprimée sous forme normale conjonctive. Vous préciserez la taille de la formule pour le cas d'un graphe de n sommets et de m arêtes.

Question 2. Donnez un algorithme qui prend en entrée un graphe non-orienté $G = (S,A)$ et qui construit une telle formule associée au problème 3-COLORATION. Donnez sa complexité.