Mini-Projet: Résolution de problèmes 2-SAT

Problème 2-SAT

Le problème "2-Satisfiability" (abrégé en 2-SAT) est celui de déterminer si une collection de variables booléennes, associées à un ensemble de contraintes sur les paires de variables, peut admettre une valuation (vrai/faux) qui satisfait toutes les contraintes.

Un problème 2-SAT est souvent décrit par une expression booléenne avec une forme spéciale : une conjonction de disjonctions. Chaque disjonction a deux arguments qui peuvent être des variables ou leurs négations. Par exemple, la formule suivante est sous une forme normale conjonctive (CNF) :

$$\phi = (a \vee b) \wedge (b \vee \neg c) \wedge (\neg b \vee \neg d) \wedge (b \vee d) \wedge (d \vee a)$$

Chaque disjonction est équivalente à une implication entre la négation d'un des arguments et l'autre. Par exemple,

$$(b \vee \neg c) \;\equiv\; (\neg b \Rightarrow \neg c) \;\equiv\; (c \Rightarrow b).$$

Grâce à cette équivalence, un problème 2-SAT peut être écrit sous une "Forme Normale Implicative (FNI)", c'est-à-dire, une conjonction d'implications.

Première partie

Exercices de compréhension

1. Écrivez la formule ϕ sous FNI.

Une FNI peut être représentée par un *graphe d'implications* : chaque sommet de ce graphe représente une variable ou sa négation et chaque arc met en évidence une relation d'implication entre les variables (ou leurs négations).

2. Dessinez le graphe d'implication de ϕ. *(Attention : s'il y en a plusieurs, ce graphe doit prendre en compte toute FNI possible de ϕ).*

Un problème 2-SAT décrit sous une FNI est satisfiable (admet une solution) si et seulement si quelque soit la variable x et sa négation $\neg x$, on n'a pas les deux séquences d'implications $x \Rightarrow y... \Rightarrow \neg x$ et $\neg x \Rightarrow z... \Rightarrow x$ en même temps. Ceci voudrait dire que $x \Leftrightarrow \neg x$, ce qui est naturellement faux !

3. Comment peut-on utiliser le graphe d'implications pour vérifier la satifiabilité de la formule qu'il représente ?

4. Citez un algorithme qui réalise cette vérification et donnez sa complexité.

5. Dans le cas ou la formule est satisfiable, comment trouve-t-on une affectation (une valuation) aux variables qui satisfait la formule ?

Deuxième partie

Le mini-projet à réaliser

Objectif. Réaliser un outil permettant de résoudre les problèmes 2-SAT de façon efficace suivant, le procédé montré plus haut !

Entrée de l'outil. Un problème 2-SAT, encodé dans un fichier, sous format standard DIMACS (`http://www.satcompetition.org/2009/format-benchmarks2009.html`). Par exemple, la formule $F = (a \vee \neg c) \wedge (b \vee \neg a)$ est représentée comme suit :

```
p cnf 3 2
1 -3 0
2 -1 0
```

où,

- la ligne "p cnf 3 2" définit, respectivement, le nombre de variables, et le nombre des conjonctions ;
- les variables a, b, c sont, respectivement, encodées par $1, 2, 3$, et la négation par le signe $-$;
- la disjonction (le ou) est représentée par un espace entre les variables ;
- le 0 est utilisé comme séparateur des conjonctions (une conjonction par ligne).

Sortie de l'outil. L'outil doit répondre "UNSAT", si le problème n'a pas de solution, sinon il répond "SAT : " en proposant une solution possible. Par exemple, pour la formule F, la réponse sera : "SAT : 1 2 -3" (les variables 1 et 2 sont à *true*, la variable 3 est à *false*).

À rendre. Un tarball contenant :

- Les sources de votre projet ainsi que les bibliothèque annexes utilisées.
- Le makefile pour compiler votre projet.
- Un script shell, nommé *test_solveur*, qui permet de lancer votre outil sur l'ensemble des fichiers d'un répertoire donné en paramètre.
- Un fichier README expliquant les particularités de votre projet.

Date limite du rendu. Le 17/06/2017 à 23h42

Évaluation. Pour évaluer votre travail, seront considérés les points suivants :

- La qualité du code (architecture, clarté, documentation, etc.)
- La correction de la réponse de l'outil.
- Le temps de réponse de l'outil (au dessus/en dessous d'un seuil).