Exercices: SAT

Lionel Vaux Auclair

Logique
HUGO@Centrale, 2021–2022

Exercice 1.

1. Écrivez un programme (par exemple en C) qui prend un nombre n en argument, et qui fait 2^n opérations (par exemple incrémenter un compteur).

2. Vérifiez qu'avec $n = 30$, ça prend un temps non nul et peut-être quelques secondes (sinon, votre machine est rapide : essayez avec $n = 31$ ou $n = 32$).

3. Vérifiez qu'avec $n = 40$, on a déjà envie d'arrêter avant la fin.

Exercice 2. Testez DPLL avec la formule :

$$(p \vee q \vee r) \wedge (p \vee \neg q \vee \neg r) \wedge (p \vee \neg w) \wedge (\neg q \vee \neg r \vee \neg w) \wedge$$
$$(\neg p \vee \neg q \vee r) \wedge (u \vee x) \wedge (u \vee \neg x) \wedge (q \vee \neg u) \wedge (\neg r \vee \neg u)$$

Exercice 3. On va utiliser un solveur **SAT**, par exemple CaDiCaL. [1]

1. Téléchargez et compilez le solveur.

Le format standard pour utiliser un solveur est le format DIMACS : [2]

- une ligne de la forme `p cnf n k` où n est le nombre de variables et k le nombre de clauses ;
- une ligne pour chaque clause avec :
  - pour chaque littéral, le numéro de la variable, précédé d'un `-` si elle est niée ;
  - un `0` pour terminer la ligne

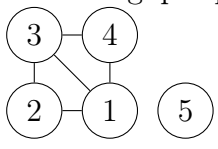
Par exemple, pour $(x \vee y) \wedge (x \vee \neg y \vee z) \wedge \neg x$:

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

2. Créez un fichier texte avec les lignes précédentes, et lancez le solveur dessus.

3. Codez la formule de l'exercice 2 et testez avec le solveur.

4. Représentez en logique propositionnelle le problème de coloriage du graphe suivant :



1. `http://fmv.jku.at/cadical/`
2. `http://www.satcompetition.org/2009/format-benchmarks2009.htmlx`

5. Codez le problème au format DIMACS et testez avec le solveur.

Il est assez pénible de retraduire une éventuelle solution fournie par le solveur, en une solution pour le problème initial. Si on veut faire ce travail, il vaut mieux programmer un peu. Par exemple :

6. Formalisez les contraintes à respecter pour un sudoku en logique propositionnelle, avec des variables $x_{i,j,k}$ signifiant « il y a un k à la ligne i et à la colonne j ».

7. Écrivez un programme qui, étant donné un tableau 9×9 contenant des nombres entre 0 et 9 (0 comptant pour une case vide) construit un fichier au format DIMACS représentant le problème de sudoku correspondant.

8. Écrivez un programme qui, étant donnée une éventuelle solution produite par le solveur, affiche le sudoku résolu.