Université Gustave Eiffel

Le 16/12/2022,
M1 Informatique

Semestre 7

Compte Rendu : MINISAT

Binôme:

MENAA Mathis
REGUEME Yohann

Encadrants:

M.THAPPER

Introduction

L'objectif de ce TP est de mettre en pratique et d'utiliser le programme *minisat,* qui est un solveur de satisfaisabilité de formules proportionnelles (SAT).

Ce rapport décrit la documentation fonctionnelle et technique de notre programme **sudoku.py**.
Nous devions modéliser des formules CNF représentant les contraintes de résolution d'une grille de sudoku pour obtenir la solution à une grille donnée si elle existe.

Collaboration : Durant ce TP, mon binôme et moi avons constamment travaillé ensemble sur tous les exercices.

Mode d'emploi

Ligne de commande : `python3 sudoku.py <input.txt>`

Input.txt : Fichier d'entrée correspondant à la grille à résoudre au format ci-dessous.

Un exemple de format :

```
4
0 0 0 4
2 0 0 0
0 1 0 0
0 0 1 0
```

En première ligne, on retrouve la taille de la grille.
Les lignes suivantes correspondent aux valeurs dans la grille.

Le programme va alors traiter les données de l'input pour former un fichier intermédiaire **.cnf**. Ensuite le solveur minisat va être appelé sur ce fichier .cnf qui renverra dans le terminal la grille résolue si une solution existe.

Observation / Résultats

AtLeastOne / AtMostOne - Exercices 2 - 5

Cette première partie à été consacrée à la mise en place de deux fonctions cruciales dans notre programme. Elles nous permettent de décrire les contraintes de résolution de la grille sur les lignes/colonnes ainsi que sur les sous grilles (si il existe qu'un seul chiffre sur une colonne/une ligne/une sous grille.

Formalisation des règles - Exercice 5 - 9

Une fois nos fonctions essentielles créées, nous avons dû écrire les fonctions nous permettant de vérifier les contraintes sur les colonnes, lignes et les sous grilles à l'aide des fonctions de la précédente partie. Nous les utiliserons pour établir notre solveur de grille de sudoku.

Création du programme sudoku.py

Création d'une classe **SudokuSolver**, permettant de :

- Générer un fichier *cnf* décrivant la grille en argument
- Exécuter le fichier *cnf* créer avec **minisat**
- Renvoyer le résultat de la commande **minisat** :
    - Insatisfaisable si la grille n'a pas de solution
    - Affiche la solution de la grille

Après tests de toutes les grilles fournies sur elearning, notre programme trouve une solution à chacune d'elles.
On remarque cependant que plus la grille est grande, plus le programme met du temps à trouver une solution :

- Taille 4x4 : 64 variables et 448 + X clauses (Time : 0,04s).
- Taille 9x9 : 729 variables et 11988 + X clauses (Time : 0,07s).
- Taille 16x16 : 4096 variables et 123904 + X clauses (Time : 0,3s).
- Taille 25x25 : 15625 variables et 752500 + X clauses (Time : 1,77s).

(X est le nombre de clause de la configuration initial)

Cela nous montre la difficulté du problème, au vu du nombre de clauses croissantes entre les tailles de grilles.

Pour conclure, nous avons compris l'importance du problème SAT qui permettait de résoudre divers problèmes complexes. Ce TP nous aura aussi permis d'utiliser le logiciel minisat en l'utilisant à travers un programme python.