Software Verification

2018-2019

TP1: SMT

Vincent Penelle

Points abordés

- Z3
- API Z3 de Ocaml
- Codage de solver de jeux (Unruly et Keen)

Exercise 1: Introduction à Z3

Z3 est un SMT-solveur de Microsoft Research. Il permet de décider la satisfiabilité de formules du premier ordre passées en paramètre. Ces formules peuvent parler de booléens, d'entiers, de réels, de tableaux, de fonctions abstraites, de bit-vectors, etc.

En cas de satisfiabilité, il permet de récupérer une affectation de variables satisfaisant la formule. En cas d'insatisfiabilité, il permet de récupérer une preuve d'indécidabilité.

Dans le cadre de ce TP, nous nous concentrerons sur la manipulation des booléens et des entiers, et nous serons amenés à récupérer un modèle (cette partie vous sera donnée). Nous n'utiliserons pas l'intégralité de ses fonctionnalités. Nous l'utiliserons via son API Ocaml, mais commençons par explorer un peu son fonctionnement.

1. Regardez rapidement le début du guide en ligne du solveur Z3 que vous trouverez ici[1], en regardant spécifiquement les sections ≪Basic Commands≫ (avant ≪Using Scopes≫), Propositional Logic et Arithmetic.

Exercise 2: L'API Ocaml de Z3 Nous nous tournons maintenant vers l'API de Z3 pour Ocaml que nous utiliserons dans le reste du cours.

Si vous n'êtes pas familiers de Ocaml, vous pouvez trouver des tutoriaux ici[2], ou, pour des interrogation précises, vous tourner vers la documentation[3]. N'hésitez pas à poser des questions!

La documentation de l'API Z3 est accessible ici[4].

1. Télécharger l'archive du TP[5], et ouvrez le fichier `z3_ml_example.ml`. Il contient des exemples d'utilisation de Z3 via Ocaml. Compilez-le avec la commande `make z3_ml_example.byte`, et lancez le résultat avec `./z3_ml_example.byte`. Observez notamment comment construire des formules booléennes et arithmétiques.

[1] `https://riseforfun.com/Z3/tutorial/guide`
[2] `http://ocaml.org/learn/`
[3] `http://caml.inria.fr/pub/docs/manual-ocaml/`
[4] `http://z3prover.github.io/api/html/ml/Z3.html`
[5] `TP-SMT.tar.gz`

Exercise 3: Unruly Unruly est un petit jeu de logique que vous trouverez ici[6]. Il consiste à remplir une grille (de dimensions paires) avec des cases blanches et noires de manière que chaque ligne et chaque colonnes contiennent un nombre égal de cases noires et blanches et qu'il n'y ait pas trois cases consécutives de même couleur.

Vous allez réaliser un solveur pour ce jeu. Tous les appels à Z3 et le parsing du jeu sont déjà implémentés, il ne vous reste plus qu'à créer la formule qui encode les contraintes du jeu. Chaque case sera représentée par une variable booléenne (vraie si blanc, fausse si noir).

1. Éxécutez la commande `make doc` et regardez la documentation produite pour UnrulySolver. Vous aurez à implémenter la fonction `game_formula`.
2. Donnez une formule encodant le fait que la ligne i de taille n a le même nombre de cases noires et blanches.
3. Donnez une formule encodant le fait que sur la ligne i de taille n, il n'y a pas 3 cases consécutives de même couleurs.
4. Implémentez la fonction `game_formula`, et testez le résultat sur les exemples fournis (attention, certains peuvent être lents).

Exercise 4: Keen Keen est un autre jeu de logique que vous trouverez là[7]. Il consiste à remplir une grille carrée de coté n avec des entiers de 1 à n tels que chaque entier n'apparaisse qu'une fois sur chaque ligne et sur chaque colonne. De plus, la grille est divisée en zones auxquelles sont associées une opération et un résultat : si l'opération est une addition (resp. multiplication), alors la somme (resp. produit) des nombre de la zone donne le résultat. Pour la soustraction et la division, les zones font nécessairement 2 cases et l'une des deux différences (resp. quotients) possibles doit être le résultat. Notez que dans le cas d'une division, le reste doit être nul.

Vous allez réaliser un solveur pour ce jeu. Tous les appels à Z3 et le parsing du jeu sont déjà implémentés, il ne vous reste plus qu'à créer la formule qui encode les contraintes du jeu. Chaque case sera représentée par une variable entière.

1. Éxécutez la commande `make doc` et regardez la documentation produite pour KeenSolver. Vous aurez à implémenter la fonction `game_formula`.
2. Donnez une formule encodant le fait que toutes les cases de la ligne i sont différentes.
3. Donnez une formule qui assure que la somme des cases d'une zone est égale à un résultat (vous choisirez une zone et un résultat quelconque pour l'exemple).
4. Donnez une formule qui assure que l'une des différence possibles entre deux cases est égale à un résultat (vous choisirez une zone et un résultat quelconque pour l'exemple).
5. Faites de même pour la multiplication et la division (attention au reste qui doit être nul).
6. Implémentez la fonction `game_formula`, et testez le résultat sur les exemples fournis (attention, certains peuvent être lents). Notamment, notez que les exemples sans multiplications sont résolus bien plus rapidement que ceux avec (n'essayez pas de résoudre la grille 9×9 avec multiplication!).

[6] `https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/unruly.html`
[7] `https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/unruly.html`