Un petit solver SMT pour la théorie de l'égalité

Nathanaël Courant

1er juin 2016

1 Utilisation

`esmt fichier.cnfuf` éxécute le solver sur le fichier fourni. Le résultat est affiché sur la sortie standard, et est soit `No solution` si le fichier d'entrée n'a aucune solution, soit une liste de séries d'égalités, une par ligne, deux variables sur la même ligne étant différentes.

2 Implantation

2.1 Décision de la théorie de l'égalité

La procédure de décision pour la théorie de l'égalité s'effectue en temps amorti $O(\alpha(n)+\log(r))$ par opération, où n est le nombre de variables et r le nombre de clauses de différence.

Pour cela, on utilise la structure de données suivante :

```
type state = Union_find.t * ISet.t Parray.t * int
```

Ici, on a une structure *union-find* qui gère les égalités, un tableau qui envoie chaque racine de l'*union-find* vers les identifiants des inégalités dont un de ses enfants est membre, et un entier indiquant le plus petit identifiant d'inégalité libre.

Cela permet de répondre à toutes les requêtes en le temps annoncé (et l'utilisation de tableaux persistants rend le backtracking en complexité $O(1)$ amorti si la version avant backtracking n'est pas réutilisée).

2.2 Solveur SAT

Le solveur SAT est basé sur l'idée des *two-watched literals*, avec de plus une revérification complète des littéraux qui peuvent avoir été déduits d'égalités précédentes par la procédure de décision de la théorie de l'égalité avant de décider une nouvelle affectation de littéral.

Une autre version fournie dans le code, mais non utilisée, est la même chose sans les *two-watched literals*, elle semble être entre 3 et 5 fois moins rapide à l'éxécution, lorsque les clauses contiennent 3 littéraux chacune.

3 Tests et performance

Cette implantation se débrouille raisonnablement bien sur des problèmes qui ont jusqu'à de l'ordre de 25 variables et 500 clauses : ainsi, une solution est trouvée en environ 20 secondes ; pour 25 variables, 700 clauses, et 5 littéraux par clause, une solution est trouvée en 2 minutes.

Pour des problèmes plus grands (30 variables, 1000 clauses), les tableaux persistants provoquent un débordement de pile au moment du backtracking.

Le script utilisé pour générer les tests est fourni ; un encodage de la 3-coloriabilité du graphe de Petersen l'est également, ainsi qu'un programme cherchant la solution du problème du zèbre (30 variables, 90 clauses).