Projet RNTL AVERILES
Fourniture F2.3 : Prototypes d'outil

LIAFA avec LSV, VERIMAG

1 Introduction

Dans le cadre du projet AVERILES nous avons dévéloppé plusieurs prototype d'outils de vérification. Il s'agit de L2CA [3], TOPICS et de ARTMC. Tous ces outils sont connectés à la plateforme commune CALIFE comme décrit dans la fourniture F1.5 (voir aussi la fourniture F2.4). Les outils L2CA et ARTMC sont aussi individuellement disponibles en téléchargement. En passant par la plateforme CALIFE chaque outil peut prendre en principe en entrée les programme en C essentiel concurrent définie dans la fourniture. Ensuite, la concurrence est résolue au niveau de la traduction du C essentiel vers le format commun. Des exemples de programmes traités se trouvent sur la page internet du projet.

2 Outils basés sur la traduction aux automates à compteurs

Dans cette partie nous présentons les deux outils L2CA (List to Counter Automata) [3] dévéloppé à VERIMAG et TOPICS (Translation Of Programs Into Counter Systems) dévéloppé au LSV. Ces outils sont basées sur les travaux [7] et [9] qui sont detaillés dans les rapport fourniture F1.3 et F2.2. Les programmes C qui peuvent être traités sont les programmes manipulant des listes simplement chaînées ainsi que des variables entières. Comme décrit dans les fournitures F1.3 (et F2.2.) le principe de la traduction des programmes manipulant ces structures est basé sur le fait que le nombre de structures de mémoire différent (en faisant abstraction de la longueur des segments de liste) est borné. Il est donc possible de traduire les programmes vers des systèmes à compteurs équivalents qui peuvent être analysés par des méthodes et outils dédiés. Les systèmes à compteurs sont des automates étendus dont les transitions permettent de manipuler des variables entières. Les systèmes à compteurs produits sont aux formats de l'outil FAST [6, 8] et ARMC (Abstraction Refinement Model Checking) [1] (uniquement pour L2CA). Comme indiqué dans [7] et [9], les propriétés que l'on souhaite véri-

fier sur un programme peuvent se réduire à des propriétés dans le système à compteurs produit. Les propriétés que l'on cherche à vérifier sont :

– l'absence de violation mémoire
– l'absence de fuite mémoire

Ces propriétés sont directement traduites sous forme de problèmes d'accessibilité dans un système à compteurs. Dans la suite nous présentons les aspects propres à chaque outil.

2.1 L2CA

L'outil L2CA prend en entrée un programme en C essentiel (sans concurrence) ainsi qu'une description d'une configuration de mémoire initiale. L2CA permet aussi de vérifier des propriétés de terminaison en utilisant l'outil ARMC [1].

2.2 TOPICS

Les programmes que TOPICS [4] prend en entrée peuvent aussi utilisés des tableaux (de taille connue) d'entiers et de listes. Il est prévu d'étendre l'outil de façon à pouvoir analyser des programmes manipulant des structures de données plus complexes ayant pour base une liste simplement chaînée, comme par exemple les listes doublement chaînées. TOPICS permet de vérifier aussi l'absence de débordement lors de l'accès aux tableaux

L'outil TOPICS peut prendre en entrée soit un fichier respectant la syntaxe du C essentiel AVERILES soit un fichier XML contenant une description de l'automate étendu au format CALIFE-XML AVERILES. Dans ce dernier cas, il faut également fournir à TOPICS un autre fichier contenant la description des types manipulés par le programme ainsi que le type de chaque variable de pointeurs et de tableaux. Il est également possible de décrire un ensemble (pouvant être infini) de configurations initiales dans un autre fichier.

3 Model-checking régulier abstrait

L'outil ARTMC [2] est l'implémentation prototype d'une méthode de vérification [10, 11] automatique de propriétés importantes de programmes manipulant des structures de données liées dynamiquement (voir fournitures F1.3 et F2.2). La méthode peut traiter des programmes non-récursifs avec des variables sur un domaine fini et des structures de données dynamiques avec *plusieurs* pointeurs next.

4 Autres outils prototype

Dans le cadre du travail [5] concernant des abstractions monotones un prototype a été aussi dévéloppé.

5 Expérimentation

La fourniture F2.5 détaille l'expérimentation des outils dans le cadre du projet AVERILES.

Références

[1] ARMC. http ://www.mpi-sb.mpg.de/~rybal/armc/.

[2] ARTMC. http ://www.fit.vutbr.cz/research/groups/verifit/tools/artmc/.

[3] L2CA. http ://www-verimag.imag.fr/~async/L2CA/l2ca.html.

[4] TOPICS. http ://www.lsv.ens-cachan.fr/~sangnier/TOPICS/index.php.

[5] P. A. Abdulla, A. Bouajjani, J. Cederberg, F. Haziza, and A. Rezine. Monotonic abstraction for programs with dynamic memory heaps. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, volume 5123 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2008.

[6] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. Fast : Fast acceleration of symbolic transition systems. In *CAV'03*, volume 2725 of *LNCS*, pages 118–121. Springer, 2003.

[7] S. Bardin, A. Finkel, É. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. In R. Bharadwaj, editor, *Proceedings of the 5th International Workshop on Automated Verification of Infinite-State Systems (AVIS'06)*, Vienna, Austria, Apr. 2006.

[8] S. Bardin, J. Leroux, and G. Point. Fast extended release. In *CAV'06*, volume 4144 of *LNCS*, pages 63–66. Springer, 2006.

[9] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In Th. Ball and R. B. Jones, editors, *Proceedings of the 18th International Conference on Computer Aided Verification (CAV'06)*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531, Seattle, Washington, USA, Aug. 2006. Springer.

[10] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In K. Yi, editor, *Proceedings of the 13th International Symposium Static Analysis (SAS'06)*, volume 4134 of *Lecture Notes in Computer Science*, pages 52–70, Seoul, Korea, Aug. 2006. Springer.

[11] P. Habermehl, R. Iosif, A. Rogalewicz, and T. Vojnar. Proving Termination of Tree Manipulating Programs. In *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07)*, volume 4762 of *Lecture Notes in Computer Science*, pages 145–161, Tokyo, Japan, 2007. Springer.