Rapport à mi-parcours du projet RNTL Averiles (Analyse et Vérification de logiciels embarqués avec structures de mémoire dynamique)

LIAFA, CRIL, EDF, LSV, VERIMAG

1 Introduction

Dans ce rapport nous résumons l'avancement du projet RNTL Averiles à mi-parcours.

2 Rappel des objectifs

Dans la proposition du projet nous avons définies les objectifs suivants :

L'objectif du projet est de proposer des approches outillées pour vérifier l'absence de défauts dans l'utilisation de la mémoire allouée dynamiquement par un logiciel de systèmes complexes industriels. Les approches proposées s'appuieront sur des techniques de model-checking. Le projet comprend alors plusieurs volets :

1. définition de modèles adéquats pour de tels programmes. Ces modèles doivent permettre de raisonner sur les différentes classes de structures de mémoire (listes, arbres, DAG, etc) et les opérations associés, ainsi que prendre en compte différents aspects liés au contrôle (mono/multi-tâche). Nous chercherons à automatiser la génération de modèle à partir d'une analyse du code source du logiciel.
2. conception de nouvelles techniques algorithmiques pour la vérification symbolique pour de tels modèles. Cela comprend l'étude de structures de représentation symbolique pour les ensembles de configurations, et le développement d'algorithmes efficaces pour l'analyse d'accessibilité utilisant ces représentations.
3. développement de prototypes d'outils implémentant ces algorithmes de vérification et leur expérimentation sur exemples significatifs fournis par EDF R&D, comme par exemple le logiciel d'un système de consignation d'états sur une installation hydraulique afin de surveiller le bon fonctionnement de l'installation.
4. intégration des outils dans une plateforme commune.

3 Travaux réalisés

Par la suite nous résumons les travaux réalisé jusqu'à présent.

3.1 Modèles

Nous avons définie d'une part

1. un modèle concret, qui est un sous-ensemble rigoureusement défini du langage ANSI C, et d'autre part
2. un modèle abstrait, qui décrit des automates étendues avec des opérations sur la mémoire.

Les modèles sont décrits dans la fourniture F1.1.

3.2 Extraction de modèles

Nous avons définie les différents étapes du passage du C restreint vers les outils de vérification en passant par la plateforme commune. Les détails se trouvent dans la fourniture F1.2.

3.3 Algorithmes de vérification

Plusieurs algorithmes de vérification ont été développé basé essentiellement d'une part sur la traduction des programmes vers des automates à compteurs et d'autre part sur des techniques de model-checking symbolique. Les détails se trouvent dans la fourniture F1.3.

3.4 Intégration des outils

Nous avons définie la plateforme en commun et commencé sa réalisation. Les détails se trouvent dans la fourniture F1.4.

3.5 Expérimentation

Les outils de vérification ont été expérimente sur des exemples standards de programmes et une étude de cas fourni par EDF. Les détails se trouvent dans la fourniture F1.5.

4 Listes des fournitures

En concordance avec la proposition les fournitures suivantes ont été réalisées jusqu'à présent :

- Fourniture F1.1 : Modèles
- Fourniture F1.2 : Extraction de modèles
- Fourniture F1.3 : Algorithmes de vérification

- Fourniture F1.4 : Prototypes d'outil
- Fourniture F1.5 : Définition de l'architecture et plate-forme intégrant les outils de vérification à partir d'un même formalisme
- Fourniture F1.6 : Expérimentation

5 Listes des publications

Nous avons publié plusieurs résultats obtenus dans le cadre du projet AVERILES dans des conférences internationales.

[1] S. Bardin, A. Finkel, É. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. In *AVIS'06*, 2006.

[2] Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomáš Vojnar. Programs with lists are counter automata. In Thomas Ball and Robert 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, August 2006. Springer.

[3] Ahmed Bouajjani, Severine Fratani, and Shaz Qadeer. Context-bounded analysis of multithreaded programs with dynamic linked structures. In *Proceedings of the International Conference on Computer Aided Verification (CAV'06)*, volume 4590 of *Lecture Notes in Computer Science*, Berlin, Germany, July 2007. Springer.

[4] Ahmed Bouajjani, Peter Habermehl, Yan Jurski, and Mihaela Sighireanu. Rewriting systems with data – A framework for reasoning about systems with unbounded structures over infinite data domains. In Erzsébet Csuhaj-Varjú and Zoltán Ésik, editors, *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT'07)*, volume 4639 of *Lecture Notes in Computer Science*, pages 1–22, Budapest, Hungary, August 2007. Springer.

[5] Ahmed Bouajjani, Peter Habermehl, Adam Rogalewicz, and Tomáš Vojnar. Abstract regular tree model checking. In Jiří Srba and Scott A. Smolka, editors, *Proceedings of the 7th International Workshop on Verification of Infinite State Systems (INFINITY'05)*, volume 149 of *Electronic Notes in Theoretical Computer Science*, pages 37–48, San Francisco, CA, USA, February 2006. Elsevier Science Publishers.

[6] Ahmed Bouajjani, Peter Habermehl, Adam Rogalewicz, and Tomáš Vojnar. Abstract regular tree model checking of complex dynamic data structures. In Kwangkeun 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, August 2006. Springer.

[7] M. Bozga and R. Iosif. On flat programs with lists. In *VMCAI 2007*, volume 4349 of *LNCS*, pages 122–136. Springer, 2007.

[8] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. In *Proceedings of the International Conference on Automata, Languages and Programming (ICALP 2006)*, volume 4052, pages 577–588, Venice, Italy, July 2006.

[9] R. Brochenin, S. Demri, and E. Lozes. Reasoning about sequences of memory states. In *LFCS'07*, volume 3634 of *LNCS*, pages 100–114. Springer, 2007.

[10] Peter Habermehl, Radu Iosif, Adam Rogalewicz, and Tomas Vojnar. Proving termination of tree manipulating programs. In Kedar Namjoshi and Tomohiro Yoneda, editors, *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07)*, Lecture Notes in Computer Science, Tokyo, Japon, October 2007. Springer.

[11] Peter Habermehl, Radu Iosif, and Tomáš Vojnar. Automata-based verification of programs with tree updates. In Holger Hermanns and Jens Palsberg, editors, *Proceedings of the 12th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *Lecture Notes in Computer Science*, pages 350–364, Vienna, Austria, March 2006. Springer.

[12] G. Yorsh, A. Rabinovich, M. Sagiv, A. Meyer, and A. Bouajjani. A logic of reachable patterns in linked data-structures. In *Proceedings of the International Conference on Foundations of Software Science and Computer Structure (FOSSACS'06)*, volume 3921 of *Lecture Notes in Computer Science*, Vienna, Austria, March 2006.