Äquivalenzanalysen - exakt oder nicht - im Vergleich

Dirk Richter (richterd@informatik.uni-halle.de),
Martin-Luther-Universität Halle-Wittenberg

Abstract: Zur Software-Modell-Prüfung sowie zum Modell basierten Testen als auch bei der Codegenerierung sind die Größe und Komplexität von Modellen entscheidende Einflussfaktoren. Aus Quellcode (wie C oder Java) gewonnene Modelle in Form von symbolischen Kellersystemen (SPDS) erlauben nicht nur präzisere Ergebnisse, sondern führen auch ohne Modellexplosion bei **exakter** Nachbildung von Rekursion zu weniger Fehlalarmen. Für diese SPDS wurden zwei verschiedene Ansätze verfolgt, die die innere Struktur der Zustände ausnutzen, um den Zustandsraum der Modelle weiter zu verkleinern. Eigene Experimente zeigen, dass diese die Modellprüfung erheblich beschleunigen bzw. die Modellprüfung erst ermöglichen oder sogar erübrigen.

Schlüsselworte: Kellersystem, Modellanalyse, Remopla, Moped, Software-Modell-Prüfung

1 Einleitung

Im Gegensatz zu verbreiteten 'Finite-State' Modellprüfern wie BLAST [HJMS03], SPIN [G. 04], NuSMV/SMV (http://www.cs.cmu.edu/ modelcheck), JavaPathFinder [VHB+03], F-Soft [ISG+05] oder Bogor (Bandera Projekt) [HD04] können mit dem Tool JMoped und dem Modellprüfer Moped Modell-Analysen,-Tests und -Prüfungen interprozedural durchgeführt werden und verbessern damit das Analyse- und Testergebnis. Viele Modellprüfer (darunter alle eben genannten außer Moped) beschränken die Rekursionstiefe oder verbieten Methodenaufrufe sogar. Durch diese Unter- bzw. Überapproximation von Methodenaufrufen enstehen Fehlalarme (False Negatives sowie False Positives), die durch korrekte Abbildung von Methodenaufrufen und Rekursion auf SPDS vermieden werden können. Es ist nicht nötig, sich auf eine konstante maximale Anzahl an Methodenaufrufen zu beschränken und eine exponentielle Modellvergrößerung z.B. durch Inlining zu riskieren.

Schon Liang und Harrold zeigten [LH99, LH03], wie Slicing, Points-To- und andere Datenfluss-Analysen von einer um so besseren Äquivalenzanalyse um so mehr profitieren. In [RZ07] und [Ric08] wurden bereits Techniken vorgestellt, um SPDS zu untersuchen und zu vereinfachen. Dort wurde u.A. gezeigt, dass verschiedene Modellanalysen im Gegensatz zur Anwendung bei herkömmlichen Programmiersprachen **entscheidbar** werden, was prinzipiell die Existenz exakter Modellanalyseverfahren für SPDS nachweist – d.h. das Analyseergebnis enthällt weder eine Über- noch eine Unterapproximation.

Hier soll nun die approximative Äquivalenzanalyse für SPDS aus [Ric08] mit einem neuen **exakten** Verfahren verglichen werden.

Literatur

[BCC+05] L. Burdy, Y. Cheon, D.R. Cok, M.D. Ernst, J.R. Kiniry, G.T. Leavens, R. Leino und E. Poll. *An overview of JML tools and applications*. International Journal on Software Tools for Technology Transfer (STTT), Volume 7, pp. 212-232, Springer, 2005.

[ES01] J. Esparza und S. Schwoon. *A BDD-based model checker for recursive programs*. LNCS Volume 2102, pp. 324-336, Springer, 2001.

[G. 04] G. J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[HD04] J. Hatcliff und M. B. Dwyer. *Bogor: An Extensible Framework for Domain-Specific Model Checking*. Newsletter of European Association of Software Science and Technology (EASST), Technical Report, SAnToS-TR2004-9., 2004.

[HJMS03] T. A. Henzinger, R. Jhala, R. Majumdar und G. Sutre. *Software Verification with BLAST*. In 10th International Workshop on Model Checking of Software (SPIN), LNCS Volume 2648, pp. 235–239. Springer, 2003.

[ISG+05] F. Ivancic, I. Shlyakhter, A. Gupta, M. K. Ganai, V. Kahlon, C. Wang und Z. Yang. *Model Checking C Programs Using F-SOFT*. Proceedings of the 2005 International Conference on Computer Design (ICCD), 2005.

[LH99] D. Liang und M. J. Harrold. *Equivalence analysis: a general technique to improve the efficiency of data-flow analyses in the presence of pointers*. Proceedings of the 1999 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, 1999.

[LH03] D. Liang und M. J. Harrold. *Equivalence analysis and its application in improving the efficiency of program slicing*. Transactions on Software Engineering and Methodology (TOSEM), Volume 11 Issue 3, 2003.

[Ric08] D. Richter. *Modellreduktionstechniken für symbolische Kellersysteme*. Proceedings of the 25. Workshop of GI-section 'Programmiersprachen und Rechenkonzepte', University Kiel, 2008.

[RZ07] D. Richter und W. Zimmermann. *Slicing zur Modellreduktion von symbolischen Kellersystemen*. Proceedings of the 24. Workshop of GI-section 'Programmiersprachen und Rechenkonzepte', University Kiel, 2007.

[Sch02] S. Schwoon. *Model-Checking Pushdown Systems*. Technische Universität München, 2002.

[SSE05] D. Suwimonteerabuth, S. Schwoon und J. Esparza. *jMoped: A Java Bytecode Checker Based on Moped*. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS, Springer, 2005.

[VHB+03] W. Visser, K. Havelund, G. Brat, S. Park und F. Lerda. *Model Checking Programs*. Automated Software Engineering Volume 10(2), pp. 203-232, Kluwer Academic, 2003.