Slicing zur Modellreduktion von symbolischen Kellersystemen

Dirk Richter
richterd@informatik.uni-halle.de
Wolf Zimmermann
zimmer@informatik.uni-halle.de

1 Einleitung

Während der Modellprüfung werden vorgegebene Eigenschaften überprüft (z.B. dass vor der Landung eines Flugzeugs stets das Fahrwerk ausgefahren ist). Ist die Modellprüfung erfolgreich, so erfüllt das Modell die vorgegebenen Eigenschaften **sicher** (in jedem Fall) im Gegensatz zum Testen, wo lediglich die Existenz von Fehlern festgestellt werden kann. Bei der Software-Modell-Prüfung werden Modelle aus Softwareprogrammen gewonnen. Modellprüfung als auch das modellbasierte Testen bzw. die Testfallgenerierung sind jedoch sehr aufwändig. Die Verwendung von Kellersystemen bei der Software-Modell-Prüfung ermöglicht dabei unter anderem die Berücksichtigung von rekursiven Methoden bzw. Funktionen. Modellprüfung von Software zur Steigerung der Softwarequalität ist bereits im Einsatz, wie die Projekte JavaPathFinder (NASA) [KT00], SMV [McM93] oder Moped [Jav01] zeigen. Allerdings sind die derzeit verfügbaren Werkzeuge (Tools) nicht in der Lage mittlere oder große Softwareprojekte zu überprüfen. Vielfach werden nur Variablen mit kleinem Zustandsraum unterstützt (logische/boolean Variable in Bebop [Tom00]) oder sogar vollständig auf **endliches** Modellprüfen beschränkt, da sonst der Suchraum für den Modellprüfer „explodiert“. Wir untersuchen Ansätze, die solchen Einschränkungen möglichst nicht unterliegen.

2 Zusammenfassung

Durch Einsatz von Slicing im Kontrollflussgraph auf Konfigurationenebene haben wir symbolische Kellersysteme (genauer: Remopla-Modelle) derartig transformiert, dass nur noch diejenigen Remopla-Modell-Transitionen im Modell enthalten sind, welche sowohl von den Startkonfigurationen erreichbar sind als auch zu angegebenen Fehlerkonfigurationen führen können. Dies reduziert die Zeit für den Modellprüfer Moped für Modelle mit hinreichend großem Konfigurationenraum. Analog wird damit aber auch der für Softwaretesten zu überdeckende Modellraum kleiner, wodurch weniger Softwaretests nötig werden.

Literatur

[Jav01] Javier Esparza, Stefan Schwoon. *A BDD-based model checker for recursive programs*. Lecture Notes in Computer Science, Band 2102, Seiten 324–336, Springer, 2001.

[KT00] Klaus Havelund und Thomas Pressburger. *Model Checking Java Programs Using Java PathFinder*. International Journal of Software Tools for Technology Transfer (STTT), Band 2(4), Seiten 366–381, url:citeseer.ist.psu.edu/havelund98model.html, 2000.

[McM93] K. McMillan. *Symbolic Moldel Checking*. Kluwer Academic Publishers, 1993.

[Tom00] Tom Ball, Sriram Rajamani. *A symbolic model checker for Boolean programs*. In SPIN Workshop 2000, Lecture Notes in Computer Science (LNCS) 1885, Seiten 113–130, Springer-Verlag, 2000.