Weitere Modellreduktionstechniken für symbolische Kellersysteme

Dirk Richter (richterd@informatik.uni-halle.de)

Martin-Luther-Universität Halle-Wittenberg

Abstract. Zur Software-Modell-Prüfung sowie zum Modell basiertem Testen als auch bei der Codegenerierung sind die Größe und Komplexität von Modellen entscheidende Einflussfaktoren. Wir untersuchen Modelle in Form von symbolischen Kellersystemen, da diese in der Lage sind, Rekursion exakt nachzubilden. Für diese symbolischen Kellersysteme habe ich verschiedene Modellanalysen und Modellreduktionstechniken implementiert, welche die Software-Modell-Prüfung in meinen Tests für den Modellprüfer Moped erheblich beschleunigen (um bis zu 96% auf nur noch 4% der zuvor benötigten Laufzeit). In einigen Fällen wird die Modellprüfung aber auch erst überhaupt ermöglicht (bis zu 21,3% der Speicherüberläufe verhindert) und in anderen Fällen erübrigt sich die Modellprüfung sogar ganz. Letzteres z.B. dann, wenn durch eine Analyse meines Tools **SPSI** (Symbolical Pushdown System Improver) bereits die Korrektheit des Modells nachgewiesen werden konnte (je nach verwendeter Parameter 7%-10%).

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

1 Vortragsbeschreibung

Bereits im letzten Jahr wurde im Rahmen dieses Workshops die Technik Slicing auf Konfigurationenebene für symbolische Kellersysteme vorgestellt. Nun sollen neue Methoden und Techniken des Tools vorgestellt und diskutiert werden. Unter anderem wird gezeigt, dass verschiedene Analysen überraschenderweise im Gegensatz zur Anwendung bei herkömmlichen Programmiersprachen plötzlich **entscheidbar** werden. Hierzu zählen z.B. die Äquivalenzanalyse, die Intervallanalyse, aber auch Slicing. Im Gegensatz zu verbreiteten "Finite-State" Modellprüfern wie BLAST, SPIN, NuSMV/SMV, JavaPathFinder, Zing oder Bogor (Bandera Projekt) können hier Modellanalysen interprozedural durchgeführt werden und verbessern damit natürlich das Analyseergebnis. Auch ist es nicht nötig, sich auf eine konstante maximale Anzahl an Methodenaufrufen zu beschränken und dadurch eine exponentielle Modellvergrößerung zu riskieren.

Symbolische Kellersysteme können mittels JMoped aus Java Bytecode gewonnen werden. Unter Verwendung des Cross-Compilers Grashopper ist man aber nicht nur in der Lage, Java 1.6 Bytecode dafür zu verwenden, sondern auch Microsoft Intermediate Language (MSIL bzw. CIL). Es ist auch prinzipiell möglich, die Gültigkeit von Java Modeling Language (JML) Annotationen zu überprüfen.