Effiziente Algorithmen für die automatische Synthese von Software aus szenariobasierten Anforderungen

Zu vergebende Masterarbeit

Keywords

Software aus Anforderungen ableiten, Szenariobasierter Entwurf, Formale Methoden, Algorithmen, Automaten, Spieltheorie, Java, Eclipse

1. PROBLEMSTELLUNG

Statt einen Computer zu programmieren, ihm einfach sagen, was er in bestimmten Situationen tun oder nicht tun soll – das ist der Traum vieler Informatiker [4]. Dieser Traum ist heute durch den Ansatz des *szenariobasierten Entwurfs* und Verfahren für die *formale Synthese* von Software greifbar geworden. Die Forschung muss allerdings noch einige Hürden überwinden, bis diese Verfahren erfolgreich in der Praxis eingesetzt werden können. Und eins ist klar – der Mensch wird weiterhin präzise überlegen und beschreiben müssen, was der Computer tun soll. Der Computer wird nie allein entscheiden können, was "richtig" ist, wenn der Mensch widersprüchliche Anforderungen eingibt.

Beim szenariobasierten Entwurf beschreiben wir in einzelnen *Szenarien*, was ein System nach einer Folge von Ereignissen, zum Beispiel Eingaben des Benutzers, tun *kann*, tun *muss* oder *nicht tun darf*. Das entspricht dem, wie heute Use Cases in Anforderungsspezifikationen beschrieben werden. Beim szenariobasierten Entwurf werden dafür präzise visuelle oder textuelle Sprachen verwendet. Es gibt dann Algorithmen, mit denen die Szenarien im Zusammenspiel ausgeführt werden können [5, 2]. Diese können zur Simulation des zu entwickelnden Systems eingesetzt werden oder sogar gleich als Implementierung des Systems dienen.

Ein Problem dieses Ansatzes ist allerdings, dass sich leicht Widersprüche zwischen Szenarien einschleichen, wenn zum Beispiel ein Szenario fordert, dass etwas passiert, was in einem anderen Szenario verboten ist. Diese Widersprüche bedeuten zwangsläufig, dass bei einer Ausführung immer eine Anforderung verletzt werden kann.

Daher wird auf das Verfahren der *formalen Synthese* zurückgegriffen. Dieses Verfahren untersucht, ob es eine *Strategie* gibt, das System so auszuführen, dass bei allen möglichen Folgen von Eingaben nie ein Szenario verletzt wird [3].

Die formale Synthese ist also äußerst hilfreich, die Berechnung ist jedoch leider sehr teuer (im schlimmsten Fall exponentiell in der Anzahl der Szenarien), sodass dieses Verfahren für größere Spezifikationen nicht eingesetzt werden kann. Dieses Problem haben auch andere formale Verfahren, wie zum Beispiel das *Model-Checking* [1]. Hier wurden aber durch verschiedene Ideen revolutionäre Fortschritte erzielt, sodass diese Verfahren heute für viele praxisrelevante Probleme erfolgreich eingesetzt werden können.

2. AUFGABENBESCHREIBUNG

Das Ziel dieser Arbeit ist, die Effizienz des formalen Syntheseverfahrens zu verbessern, damit auch diese Technik erfolgreich in der Praxis eingesetzt werden kann. Dabei kann auf bestehende Ideen aus anderen formalen Verfahren zurückgegriffen werden. Wir haben bereits einige Ideen identifiziert, die als Startpunkt dienen können. Zum Beispiel kann ausgenutzt werden, dass oft viele Schritte in verschiedenen Szenarien unabhängig voneinander sind. Auch unser Verfahren der *inkrementellen Synthese* [3] kann erweitert werden. Eine der Verbesserungsideen soll in der Arbeit konkret ausgearbeitet und auf Basis unseres Werkzeugs ScenarioTools[1] prototypische implementiert und bewertet werden.

3. LITERATURVERZEICHNIS

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[2] C. Brenner, J. Greenyer, and V. Panzica La Manna. The ScenarioTools play-out of Modal Sequence Diagram specifications with environment assumptions. In *Proceedings of the 12th International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2013)*, volume 58. EASST, 2013. `http://goo.gl/m9L6qX`.

[3] J. Greenyer, C. Brenner, M. Cordy, P. Heymans, and E. Gressi. Incrementally synthesizing controllers from scenario-based product line specifications. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (FSE 13)*, ESEC/FSE 2013, pages 433–443, New York, NY, USA, 2013. ACM. `http://goo.gl/lSy3n4`.

[4] D. Harel. Can programming be liberated, period? *Computer*, 41(1):28–37, Jan 2008. `http://goo.gl/z1mpuh`.

[5] D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003. `http://www.wisdom.weizmann.ac.il/~playbook/`.

Kontakt

Prof. Dr. Joel Greenyer
Fachgebiet Software Engineering,
Leibniz Universität Hannover
`greenyer@inf.uni-hannover.de`

[1] `http://scenariotools.org`