Advancing Automated, Permission-Based Program Verification Using Symbolic Execution

Author(s):
Schwerhoff, Malte H.

Publication Date:
2016

Permanent Link:

https://doi.org/10.3929/ethz-a-010835519 →

Diss. ETH No. 23977

Advancing Automated, Permission-Based Program Verification Using Symbolic Execution

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

Presented by

Malte Hermann Schwerhoff
MSc ETH CS, ETH Zurich
Born on 22nd March 1981
Citizen of Germany

Accepted on the recommendation of

Prof. Peter Müller (examiner)
Prof. Bart Jacobs (co-examiner)
Prof. Viktor Kuncak (co-examiner)
Prof. Martin Vechev (co-examiner)

2016

Abstract

Proving the correctness of programs by rigorously applying formal methods such as deductive verification has attracted substantial interest in the last two decades, from both the scientific community and industry. This has led to numerous important theoretical results and successful practical applications, benefiting greatly from two parallel developments: (1) the introduction and exploration of *permission logics* such as separation logic, which are particularly suitable for *modularly* reasoning about *concurrent, heap-manipulating* programs, and (2) the emergence of a de-facto standard architecture for reusable *verification infrastructures*, which significantly reduces the effort involved with developing *automated* verification tools. The existence of automated verification tools is a prerequisite for scaling verification to the size and complexity of real-world software, but it also makes verification more easily accessible and applicable in general: as a result, researchers are provided with valuable feedback, enabling them to identify remaining challenges. Existing verification infrastructures, however, are not well-suited for permission-based verification, which impedes the further development of the field. In particular, central aspects such as program heaps and permissions must be encoded, which substantially complicates the development of efficient and precise verification tools.

This thesis aims to remedy the situation by developing a verification infrastructure that facilitates the development of automated, permission-based verification tools for race-free, concurrent and heap-manipulating programs. To achieve this goal, we make the following three contributions: first, we design Viper, a novel *intermediate verification language*, carefully balanced to be sufficiently expressive to enable the encoding of different programming languages, program properties and specification languages, while also facilitating the development of efficient and precise verification tools. Second, we develop Silicon, an automated, symbolic-execution-based verifier for the Viper language that exhibits stable and good performance, and is more complete than comparable verifiers. Third, we extend Viper and Silicon with support for two challenging features of permission logics — iterated separating conjunctions and magic wands — that have shown to be useful in by-hand proofs of various properties, but for which no existing verifier provides comparably direct and automated support.

The *Viper verification infrastructure*, which includes the Viper language and the automated verifier Silicon, already had an impact on the field of automated, permission-based verification: it is actively being used at three different universities for building several verification tools, it won an award at a verification competition and it was used for teaching at a summer school.

Zusammenfassung

In zahlreichen universitären und industriellen Forschungsprojekten wurde in den letzten zwei Jahrzehnten die Möglichkeit untersucht, die Fehlerfreiheit von Programmen mittels formaler Methoden mathematisch präzise zu beweisen. Diese Anstrengungen führten zu einer Vielzahl bedeutsamer Ergebnisse, sowohl auf der theoretischen als auch auf der praktischen Seite; Erfolge, die durch zwei parallel stattfindende Entwicklungen ermöglicht wurden: (1) die Einführung und intensive Weiterentwicklung von Separation Logic und anderen *Permission-Logiken*, die sich als besonders hilfreiches Instrument zur *modularen* Verifikation von nebenläufigen, heap-manipulierenden Programmen erwiesen, sowie (2) das Herauskristallisieren einer De-facto-Standardarchitektur für wiederverwendbare bzw. vielseitig einsetzbare *Verifikationsinfrastrukturen*, welche den mit der Entwicklung von *automatisierten* Verifikationswerkzeugen verbundenen Aufwand erheblich reduzieren. Erst solche automatisierten *Verifier* ermöglichen die Verifikation von grossen, in der Praxis genutzten Softwaresystemen: ohne Automatisierung liessen sich die zur Programmverifikation eingesetzten Techniken nicht mit vertretbarem Aufwand auf die Komplexität dieser Systeme skalieren. Die Anwendbarkeit ihrer Techniken auf zahlreiche unterschiedliche Systeme ermöglicht es zudem Forschern, wertvolle Rückmeldungen von Anwendern zu erhalten, die dabei helfen können, noch offene Probleme zu identifizieren. Existierende Verifikationsinfrastrukturen sind jedoch nur unzureichend dazu geeignet als Grundlagen für die Entwicklung von permission-basierten Verifiern verwendet zu werden: zentrale Aspekte der Programmverifikation wie Heaps und Permissions können nicht direkt repräsentiert werden und müssen daher kodiert werden, was die Entwicklung von effizienten und präzisen Verifikationswerkzeugen substanziell erschwert.

Das Ziel dieser Arbeit ist es daher eine Infrastruktur bereitzustellen, die die Entwicklung von automatisierten und permission-basierten Werkzeugen für die Verifikation von nebenläufigen, interferenzfreien und heap-manipulierenden Programmen massgeblich vereinfacht. Den Kern dieser Arbeit bilden die folgenden drei Beiträge: (1) der Entwurf von Viper, einer neuartigen *Zwischensprache*, die als Grundlage für die Entwicklung von effizienten und präzisen Verifiern dient und die ausdrucksstark genug ist, um als Zwischensprache für unterschiedliche Programmiersprachen, Spezifikationssprachen und Programmeigenschaften genutzt zu werden; (2) die Entwicklung von Silicon, eines automatisierten, auf Symbolic Execution basierenden Verifiers für Viper, der vollständiger als vergleichbare Verifier ist und trotzdem gute und stabile Laufzeitleistung erbringt; (3) die Erweiterung von Viper und Silicon um die Unterstützung für zwei anspruchsvolle, permission-logische Junktoren — die iterierte trennende Konjunktion sowie die trennende Implikation — die sich in manuell geführten Beweisen als äusserst nützlich erwiesen haben, aber bisher nicht von automatisierten Verifiern unterstützt wurden.

Die im Rahmen dieser Arbeit entwickelte *Verifikationsinfrastruktur Viper*, welche die Viper-Zwischensprache sowie den automatisierten Verifier Silicon beinhaltet, wird bereits an drei Universitäten zur Entwicklung von Verifikationswerkzeugen genutzt, sie gewann einen Preis während eines Verifikationswettbewerbs und sie wurde in der Lehre an einer Sommerakademie eingesetzt.