Diss. ETH No. 20900

Program Analyses for Automatic and Precise Error Detection

Dissertation
submitted to
ETH Zurich
for the degree of
Doctor of Sciences

by

Michael Pradel
Diplom-Informatiker, Technische Universität Dresden
Diplôme d'Ingénieur, Ecole Centrale Paris
born March 10, 1983
citizen of Germany

accepted on the recommendation of
Prof. Dr. Thomas R. Gross, examiner
Prof. Dr. Jonathan Aldrich, co-examiner
Prof. Dr. Andreas Zeller, co-examiner

2012

Abstract

One of the largest challenges in software development is to ensure that the software is correct. Almost all software that is complex enough to accomplish a useful task contains programming errors. Unfortunately, developers must allocate their time to various activities and often, they do not have enough time for searching programming errors.

The goal of this dissertation is to support developers in finding programming errors despite a limited time budget. Therefore, we focus on program analyses with three properties. First, the analyses are *automatic*, that is, the only input required to analyze a program is the source code (or byte code) of the program itself. In particular, an automatic analysis does not rely on formal specifications or manually written test suites. Second, the analyses are *precise*, that is, they report warnings that are guaranteed to point to programming errors or that have a high chance of pointing to programming errors, instead of false positives. Third, the analyses can be applied to real-world software with *low human and computational effort*, that is, they provide developers a push button approach for existing code.

This dissertation argues that automatic program analysis allows for precisely detecting errors with little effort. The key idea is to leverage programs as their own oracles, for example, by leveraging a program as an executable specification for itself or by checking a program against properties inferred from the program itself. To support our thesis, we present five automatic and precise program analyses that effectively and efficiently detect programming errors. The analyses presented in this dissertation consider different kinds of errors (for example, incorrect API usages and thread safety violations), different kinds of programs (sequential and concurrent), and leverage different analysis techniques (static and dynamic). We evaluate our approach with mature and well-tested Java and C programs and show that it reveals errors automatically, precisely, and with low effort.

Zusammenfassung

Eine der grössten Herausforderungen der Softwareentwicklung ist es, die Korrektheit eines Programmes sicherzustellen. Die inhärente Komplexität jeder anspruchsvollen Software führt mit sehr hoher Wahrscheinlichkeit zu Programmierfehlern. In der ihnen zur Verfügung stehenden Zeit haben Entwickler allerdings selten die Möglichkeit, eine gründliche Fehlersuche zu realisieren.

Das Ziel dieser Dissertation ist es, Softwareentwicklern Möglichkeiten aufzuzeigen, Programmierfehler trotz eines knappen Zeitbudgets zu finden. Um dieses Ziel zu erreichen, präsentieren wir Programmanalysen mit den drei folgenden Eigenschaften: 1) Die Analysen sind automatisch. Die einzige Eingabe, die eine Analyse benötigt, ist der Quelltext des zu analysierenden Programmes. Insbesondere werden weder ausführbare Tests noch eine formale Spezifikation benötigt. 2) Die Analysen sind präzise. Die einzige Ausgabe, die eine Analyse liefert, sind Warnungen, welche mit Sicherheit oder mit sehr hoher Wahrscheinlichkeit auf tatsächliche Programmierfehler hinweisen. Insbesondere produziert eine präzise Analyse wenige oder gar keine falsch-positive Warnungen. 3) Die Analysen sind nicht aufwendig. Sie benötigen nur einen geringen Rechenaufwand und Softwareentwickler können sie mit wenig Arbeitsaufwand nutzen.

Die vorliegende Arbeit vertritt die These, dass automatische und präzise Programmanalyse helfen kann, mit geringem Aufwand Programmierfehler zu finden. Die Kernidee hierbei ist, Programme als ihre eigenen Testorakel zu nutzen. So verwenden wir beispielsweise ein Programm als ausführbare Spezifikation für sich selbst, oder überprüfen, ob ein Programm Spezifikationen einhält, welche vom Programm selbst abgeleitet wurden. Wir stellen fünf Programmanalysen vor, welche wirksam und effizient Fehler aufdecken. Diese Programmanalysen betrachten verschiedene Arten von Fehlern (z.B. inkorrekte API-Benutzungen oder Verletzungen von Thread Safety), verschiedene Arten von Programmen (sequentiell und nebenläufig), und verwenden diverse Analysetechniken (statisch und dynamisch). Zur Evaluierung unseres Ansatzes wenden wir die Analysen auf ausgereifte und praxiserprobte Java- und C-Programme an. Unsere Ergebnisse zeigen, dass die Analysen Programmierfehler automatisch, präzise und ohne grossen Aufwand finden.