Narrowing the gap between verification and systematic testing

DISS. ETH NO. 22795

Narrowing the gap between verification and systematic testing

A thesis submitted to attain the degree of

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

Presented by

Maria Christakis
Dipl. Electrical and Computer Engineering
National Technical University of Athens, Greece

Born on October 9, 1986
Citizen of Greece

Accepted on the recommendation of

Prof.	Peter Müller	:	examiner
Prof.	Cristian Cadar	:	co-examiner
Dr.	Patrice Godefroid	:	co-examiner
Prof.	Thomas Gross	:	co-examiner

2015

Abstract

This dissertation focuses on narrowing the gap between verification and systematic testing, in two directions: (1) by complementing verification with systematic testing, and (2) by pushing systematic testing toward reaching verification.

In the first direction, we explore how to effectively combine unsound static analysis with systematic testing, so as to guide test generation toward properties that have not been checked (soundly) by a static analyzer. Our combination significantly reduces the test effort while checking more unverified properties. In the process of testing properties that are typically not soundly verified by static analyzers, we identify important limitations in existing test generation tools, with respect to considering sufficient oracles and all factors that affect the outcome of these oracles. Specifically, testing tools that use object invariants as filters on input data do not thoroughly check whether these invariants are actually maintained by the unit under test. Moreover, existing test generation tools ignore the potential interaction of static state with a unit under test. We address these issues by presenting novel techniques based on static analysis and dynamic symbolic execution. Our techniques detected a significant number of errors that were previously missed by existing tools. We also investigate in which ways it is beneficial to complement sound, interactive verification with systematic testing for the properties that have not been verified yet.

In the second direction, we push systematic testing toward verification, in particular, toward proving memory safety of the ANI Windows image parser. This is achieved by concentrating on the main limitations of systematic dynamic test generation, namely, imperfect symbolic execution and path explosion, in the context of this parser. Based on insights from attempting to reach verification of the ANI parser using only systematic testing, we then define IC-Cut, a new compositional search strategy for automatically and dynamically discovering simple function interfaces, where large programs can be effectively decomposed. IC-Cut preserves code coverage and increases bug finding in significantly less exploration time compared to the current search strategy of the dynamic symbolic execution tool SAGE. An additional novelty of IC-Cut is that it can identify which decomposed program units are exhaustively tested and, thus, dynamically verified.

Résumé

Cette thèse s'efforce de combler le fossé qui existe aujourd'hui entre vérification et test systématique en explorant deux axes: d'une part, en complétant la vérification par des tests systématiques, et d'autre part, en repoussant les limites des tests systématiques pour obtenir des garanties proches de celles obtenues par la vérification.

Dans un premier temps, nous étudions l'interaction des analyses statiques non-sûres avec le test systématique ; nous montrons comment orienter la génération de tests de manière à couvrir des propriétés qui n'ont pas été vérifiées (de manière sûre) par l'analyseur statique. Notre technique combinant les deux approches réduit de manière significative les efforts consacrés au test, tout en vérifiant davantage de propriétés. Lorsqu'il s'agit de tester les propriétés qui ne sont généralement pas vérifiées de manière correcte par les analyseurs statiques, nous mettons en exergue d'importantes limitations des outils de génération de tests préexistants ; en particulier, nous soulignons l'importance du choix d'un oracle correct, ainsi que l'importance de prendre en compte tous les facteurs susceptibles d'influencer les résultats de l'oracle. Plus précisément, les outils de test qui se basent sur des invariants d'objets pour filtrer leurs entrées ne vérifient généralement pas que ces invariants sont bel et bien maintenus dans le module présentement soumis aux tests. De plus, les outils actuels de génération de tests ne prennent pas en compte l'interaction entre l'état global du programme et le comportement du module testé. Nous proposons une réponse à ces problèmes, et présentons des techniques novatrices basées sur l'analyse statique et l'exécution symbolique dynamique. Nos techniques ont réussi à identifier un nombre tout à fait significatif d'erreurs qui n'avaient jusqu'alors jamais été décelées par les outils existants. Nous explorons également différents contextes où il peut s'avérer utile de compléter une phase de vérification interactive sûre par du test systématique visant les propriétés qui n'ont pas encore été vérifiées.

Dans un second temps, nous rapprochons le test systématique de la vérification, plus particulièrement, en essayant de prouver l'absence d'erreurs mémoires dans l'analyseur syntaxique d'images Windows ANI. Ce résultat est obtenu en se focalisant sur les limitations fondamentales de la génération systématique de tests dynamiques, à savoir, dans le cas de cet analyseur syntaxique, une exécution symbolique imparfaite et une explosion combina-

toire. Ce que nous avons appris en vérifiant l'analyseur syntaxique ANI, et ce uniquement à l'aide de test systématique, a été mis à profit dans le projet IC-Cut: IC-Cut est une nouvelle procédure de recherche pour le test systématique, qui opère de manière compositionnelle, et permet de découvrir les interfaces simples des fonctions existantes. Il est ainsi possible de décomposer de larges programmes. Comparé à la stratégie de recherche de l'outil SAGE (exécution symbolique dynamique), IC-Cut garde la même surface de code couvert, tout en consacrant beaucoup moins de temps à l'exploration. Un apport supplémentaire d'IC-Cut est qu'il permet d'identifier lesquels des modules ont été testés de manière exhaustive, c'est-à-dire, vérifiés de manière dynamique.