Lecture Notes in Computer Science 10712

More information about this series at http://www.springer.com/series/7408

Andrei Paskevich · Thomas Wies (Eds.)

Verified Software

Theories, Tools, and Experiments

9th International Conference, VSTTE 2017
Heidelberg, Germany, July 22–23, 2017
Revised Selected Papers

Springer

Editors
Andrei Paskevich
Paris-Sud University
Orsay
France

Thomas Wies
New York University
New York, NY
USA

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-72307-5 ISBN 978-3-319-72308-2 (eBook)
https://doi.org/10.1007/978-3-319-72308-2

Library of Congress Control Number: 2017961803

LNCS Sublibrary: SL2 – Programming and Software Engineering

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the proceedings of the 9th International Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2017), held during July 22–23, 2017 in Heidelberg, Germany, and co-located with the 29th International Conference on Computer-Aided Verification.

The goal of the VSTTE conference series is to advance the state of the art in the science and technology of software verification, through the interaction of theory development, tool evolution, and experimental validation. We solicited contributions describing significant advances in the production of verified software, i.e., software that has been proven to meet its functional specifications. Submissions of theoretical, practical, and experimental contributions were equally encouraged, including those that focus on specific problems or problem domains. We were especially interested in submissions describing large-scale verification efforts that involve collaboration, theory unification, tool integration, and formalized domain knowledge. We also welcomed papers describing novel experiments and case studies evaluating verification techniques and technologies. The topics of interest included education, requirements modeling, specification languages, specification/verification/certification case studies, formal calculi, software design methods, automatic code generation, refinement methodologies, compositional analysis, verification tools (e.g., static analysis, dynamic analysis, model checking, theorem proving, satisfiability), tool integration, benchmarks, challenge problems, and integrated verification environments.

The inaugural VSTTE conference was held at ETH Zurich in October 2005, and the following editions took place in Toronto (2008 and 2016), Edinburgh (2010), Philadelphia (2012), Menlo Park (2013), Vienna (2014), and San Francisco (2015).

This year we received 20 submissions. Each submission was reviewed by three members of the Program Committee. The committee decided to accept 12 papers for presentation at the conference. The program also included four invited talks, given by Jan Hoffmann (CMU, USA), Shaz Qadeer (Microsoft, USA), Christoph Weidenbach (MPI for Informatics, Germany), and Santiago Zanella-Beguelin (Microsoft, UK).

We would like to thank the invited speakers and the authors for their excellent contributions to the program this year, the Program Committee and external reviewers for diligently reviewing the submissions, and the organizers of CAV 2017 for their help in organizing this event. We also thank Natarajan Shankar for his tireless stewardship of the VSTTE conference series over the years.

The VSTTE 2017 conference and the present volume were prepared with the help of EasyChair.

October 2017

Andrei Paskevich
Thomas Wies

Organization

Program Committee

June Andronick	University of New South Wales, Australia
Christel Baier	Technical University of Dresden, Germany
Sandrine Blazy	Université de Rennes 1, France
Arthur Charguéraud	Inria, France
Ernie Cohen	Amazon Web Services, USA
Rayna Dimitrova	UT Austin, USA
Carlo A. Furia	Chalmers University of Technology, Sweden
Arie Gurfinkel	University of Waterloo, Canada
Hossein Hojjat	Rochester Institute of Technology, USA
Marieke Huisman	University of Twente, The Netherlands
Bart Jacobs	Katholieke Universiteit Leuven, Belgium
Rajeev Joshi	NASA Jet Propulsion Laboratory, USA
Zachary Kincaid	Princeton University, USA
Shuvendu Lahiri	Microsoft, USA
Akash Lal	Microsoft, India
Francesco Logozzo	Facebook, USA
Peter Müller	ETH Zürich, Switzerland
Jorge A. Navas	SRI International, USA
Scott Owens	University of Kent, UK
Andrei Paskevich	Université Paris-Sud, France
Gerhard Schellhorn	Universität Augsburg, Germany
Peter Schrammel	University of Sussex, UK
Natarajan Shankar	SRI International, USA
Mihaela Sighireanu	Université Paris Diderot, France
Julien Signoles	CEA LIST, France
Michael Tautschnig	Queen Mary University of London, UK
Tachio Terauchi	Waseda University, Japan
Oksana Tkachuk	NASA Ames Research Center, USA
Mattias Ulbrich	Karlsruhe Institute of Technology, Germany
Thomas Wies	New York University, USA

Additional Reviewers

Dubslaff, Clemens
Haneberg, Dominik
Kumar, Ramana
Myreen, Magnus O.
Pfähler, Jörg
Trieu, Alix

Abstracts of Short Papers

Everest: A Verified and High-Performance HTTPS Stack

Santiago Zanella-Beguelin

Microsoft Research, UK

Abstract. The HTTPS ecosystem is the foundation of Internet security, with the TLS protocol and numerous cryptographic constructions at its core. Unfortunately, this ecosystem is extremely brittle, with frequent emergency patches and headline-grabbing attacks (e.g. Heartbleed, Logjam, Freak). The Everest expedition, joint between Microsoft Research, Inria and CMU, is a 5-year large-scale verification effort aimed at solving this problem by constructing a machine-checked, high-performance, standards-compliant implementation of the full HTTPS ecosystem. This talk is a report on the progress after just over one year into our expedition, and will overview the various verification tools that we use and their integration, including:

- F*, a dependently-typed ML-like language for programming and verification at high level;
- Low*, a subset of F* designed for C-like imperative programming;
- KreMLin, a compiler toolchain that extracts Low* programs to C;
- Vale, an extensible macro assembly language that uses F* as a verification backend.

Our flagship project is miTLS, a reference implementation of TLS using cryptographic components programmed and verified in F*, Low*, and Vale. We compile all our code to source quality C and assembly, suitable for independent audit and deployment. miTLS supports the latest TLS 1.3 standard, including Zero Round-Trip Time (0-RTT) resumption, and has been integrated in `libcurl` and the `nginx` web server.

Design Principles of Automated Reasoning Systems

Christoph Weidenbach

Max Planck Institute for Informatics, Germany

Abstract. An automated reasoning system is the implementation of an algorithm that adds a strategy to a calculus that is based on a logic. Typically, automated reasoning systems "solve" NP-hard problems or beyond. Therefore, I argue that automated reasoning system need often to be specific to a given problem. The combination of a system and a problem is called an application.

In the talk I discuss design principles based on this layered view of automated reasoning systems and their applications. I select and discuss design principles from all six layers: application, system, implementation, algorithm, calculus, and logic.

Why Verification Cannot Ignore Resource Usage

Jan Hoffmann

Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Verified programs only execute as specified if a sufficient amount of resources, such as time and memory, is available at runtime. Moreover, resource usage is often directly connected to correctness and security properties that we wish to verify. This talk will show examples of such connections and present recent work on automatic inference and verification of resource-usage bounds for functional and imperative programs. These automatic methods can be combined with other verification techniques to provide stronger guarantees at runtime.

Constructing Correct Concurrent Programs Layer by Layer

Shaz Qadeer

Microsoft Research, USA

Abstract. CIVL is a refinement-oriented verifier for concurrent programs implemented as a conservative extension to the Boogie verification system. CIVL allows the proof of correctness of a concurrent program — shared-memory or message-passing — to be described as a sequence of program layers. The safety of a layer implies the safety of the layer just below, thus allowing the safety of the highest layer to transitively imply the safety of the lowest.

The central theme in CIVL is reasoning about atomic actions. Different layers of a program describe the behavior of the program using atomic actions, higher layers with coarse-grained and lower layers with fine-grained atomic actions. The formal and automated verification justifying the connection among layers combines several techniques — linear variables, reduction based on movers, location invariants, and procedure-local abstraction.

CIVL is available in the master branch of Boogie together with more than fifty micro-benchmarks. CIVL has also been used to refine a realistic concurrent garbage collection algorithm from a simple high-level specification down to a highly-concurrent implementation described in terms of individual memory accesses.

Contents