Lecture Notes in Computer Science 9154

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

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

Jasmin Christian Blanchette
Nikolai Kosmatov (Eds.)

Tests and Proofs

9th International Conference, TAP 2015
Held as Part of STAF 2015
L'Aquila, Italy, July 22–24, 2015
Proceedings

Springer

Editors
Jasmin Christian Blanchette
Inria Nancy & LORIA
Villers-lès-Nancy
France

Nikolai Kosmatov
CEA LIST Nano-Innov
Saclay
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-21214-2 ISBN 978-3-319-21215-9 (eBook)
DOI 10.1007/978-3-319-21215-9

Library of Congress Control Number: 2015942786

LNCS Sublibrary: SL2 – Programming and Software Engineering

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Foreword

Software Technologies: Applications and Foundations (STAF) is a federation of a number of leading conferences on software technologies. It provides a loose umbrella organization for practical software technologies conferences, supported by a Steering Committee that provides continuity. The STAF federated event runs annually; the conferences that participate can vary from year to year, but all focus on practical and foundational advances in software technology. The conferences address all aspects of software technology, from object-oriented design, testing, mathematical approaches to modeling and verification, model transformation, graph transformation, model-driven engineering, aspect-oriented development, and tools.

STAF 2015 was held at the University of L'Aquila, Italy, during July 20–24, 2015, and hosted four conferences (ICMT 2015, ECMFA 2015, ICGT 2015, and TAP 2015), a long-running transformation tools contest (TTC 2015), seven workshops affiliated with the conferences, a doctoral symposium, and a project showcase (for the first time). The event featured six internationally renowned keynote speakers, a tutorial, and welcomed participants from around the globe.

This has been the first scientific event in computer science after the earthquake that occurred in 2009 and affected L'Aquila. It is a small, and yet big step toward the grand achievement of restoring some form of normality in this place and its people.

The STAF Organizing Committee thanks all participants for submitting and attending, the program chairs and Steering Committee members for the individual conferences, the keynote speakers for their thoughtful, insightful, and engaging talks, the University of L'Aquila, Comune dell'Aquila, the local Department of Human Science, and CEA LIST for their support: *Grazie a tutti!*

July 2015 Alfonso Pierantonio

Preface

This volume contains the papers presented at the 9th International Conference on Tests and Proofs (TAP 2015), held during July 22–24, 2015, in L'Aquila, Italy, as part of the Software Technologies: Applications and Foundations (STAF) federated event.

TAP 2015 was the ninth event in a series of conferences devoted to the synergy of proofs and tests. Abandoning the traditional separation of formal verification and testing as orthogonal research fields, TAP aims at the identification of common ground across the different research communities. In particular, both testing and proving follow the goal of improving the quality of software and hardware, but with different means. Therefore, TAP provides a forum for the cross-fertilization of ideas and approaches from the formal verification community and the testing community, abandoning earlier dogmatic views on the incompatibility of proving and testing. TAP offers a meeting place for researchers who combine proofs and tests in an interdisciplinary manner by taking the best from both worlds.

Since its first edition at the ETH Zürich in 2007, TAP has been organized annually with great success. TAP was hosted by the Monash University Prato Centre near Florence in 2008, the ETH Zürich in 2009, the University of Malaga in 2010, the ETH Zürich in 2011, the Czech Technical University in Prague in 2012, the Budapest University of Technology and Economics in 2013, and the University of York in 2014. From 2010 to 2012, TAP was co-located with the TOOLS conference series on advanced software technologies. In 2013, TAP became part of STAF, which was formed after the end of TOOLS.

For the ninth edition of TAP, held at Università degli Studi dell'Aquila, we initially received 28 abstracts, leading to 21 submissions that were considered for reviewing. After a rigorous reviewing process and a discussion phase, we finally accepted 11 long papers and one short paper. For each paper, we required at least three reviews from the Program Committee or from external reviewers assigned by Program Committee members. The accepted papers contribute to various testing techniques (model-based, property-based, grammar-based, bounded-exhaustive), fault localization, model-driven engineering, as well as model coverage, consistency, and validation, among others. Many papers rely on interactive and automatic theorem provers, including SMT solvers and model checkers.

We wish to sincerely thank all authors who submitted their work for consideration. Further, we would like to thank the Program Committee members as well as the additional reviewers for their enthusiasm and their professional work in the review and selection process. Their names are listed on the following pages.

TAP 2015 featured two keynotes: "Mind the Gap: At the Crossroads of Design, Implementation, and Foundations" by Einar Broch Johnsen and "Reasoning about C Concurrency and Compilers" by Francesco Zappa Nardelli. Furthermore, Carlo A. Furia gave an invited tutorial on "Testing, Fixing, and Proving with Contracts." Many thanks to the three invited presenters for accepting our invitations.

Finally, we would also like to thank the organizers of the STAF event, in particular the general chair Alfonso Pierantonio and the publication chairs Louis Rose and Javier Troya, for their hard work and their support in making the conference a success. We thank the Università degli Studi dell'Aquila for providing the facilities. We thank the EasyChair developers for allowing us to use their conference management system and Springer for publishing these proceedings. We thank CEA LIST and the Chair of Software Engineering of ETH Zürich for financial support.

July 2015

Jasmin Christian Blanchette
Nikolai Kosmatov

Organization

Program Committee

Bernhard K. Aichernig	TU Graz, Austria
Dirk Beyer	University of Passau, Germany
Nikolaj Bjørner	Microsoft Research, Redmond, Washington, USA
Jasmin Christian Blanchette	Inria Nancy – Grand-Est, France
Achim D. Brucker	SAP AG, Karlsruhe, Germany
Koen Claessen	Chalmers University of Technology, Gothenburg, Sweden
Robert Clarisó	Universitat Oberta de Catalunya, Spain
Marco Comini	University of Udine, Italy
Catherine Dubois	ENSIIE-CEDRIC, Evry, France
Juhan Ernits	Tallinn University of Technology, Estonia
Gordon Fraser	University of Sheffield, UK
Angelo Gargantini	University of Bergamo, Italy
Christoph Gladisch	Karlsruhe Institute of Technology, Germany
Martin Gogolla	University of Bremen, Germany
Arnaud Gotlieb	SIMULA Research Laboratory, Oslo, Norway
Reiner Hähnle	Technical University of Darmstadt, Germany
Bart Jacobs	Katholieke Universiteit Leuven, Belgium
Jacques Julliard	University of Franche-Comté, France
Thierry Jéron	Inria Rennes – Bretagne Atlantique, France
Gregory Kapfhammer	Allegheny College, Pennsylvania, USA
Nikolai Kosmatov	CEA LIST, Saclay, France
Victor Kuliamin	Russian Academy of Sciences, Moscow, Russia
Panagiotis Manolios	Northeastern University, Boston, Massachusetts, USA
Karl Meinke	KTH, Stockholm, Sweden
Alexandre Petrenko	CRIM, Montreal, Canada
Andrew Reynolds	École Polytechnique Fédérale de Lausanne, Switzerland
Martina Seidl	Johannes Kepler University Linz, Austria
Nikolai Tillmann	Microsoft Research, Redmond, Washington, USA
T.H. Tse	The University of Hong Kong, SAR China
Margus Veanes	Microsoft Research, Redmond, Washington, USA
Luca Viganò	King's College London, UK
Burkhart Wolff	University of Paris-Sud, France
Fatiha Zaïdi	University of Paris-Sud, France

Additional Reviewers

Dury, Arnaud
Hentschel, Martin
Lorber, Florian
Majdoub, Lotfi
Omer Landry, Nguena Timo
Pradhan, Dipesh
Soeken, Mathias

Abstracts of Invited Talks

Testing, Fixing, and Proving with Contracts

Carlo A. Furia

Chair of Software Engineering, Department of Computer Science,
ETH Zurich, Zürich, Switzerland
caf@inf.ethz.ch
bugcounting.net

In the mind of the practitioner, the term "formal specification" often conjures up ghastly images of impenetrable logic formulas that only highly-trained experts can digest. Our experience with using contracts indicates, however, that developing simple elements of formal specification embedded in the program text as assertions requires an effort that is generally compatible with standard development practices [2]. In exchange for it, even the very simple contracts that programmers write can improve the effectiveness of a variety of analysis and verification techniques.

In this tutorial, I presented a number of tools we developed for the Eiffel programming language that take advantage of contracts in various guises. AutoTest generates random unit tests and uses contracts as oracles to automatically detect bugs; it has been used to find hundreds of errors in standard libraries. AutoFix builds source-code patches that turn failing tests into passing tests, thus automatically providing program repair for real software faults. AutoProof supports the static full-fledged verification of functional properties in object-oriented software; it has been used to verify the complete functionality of a realistic general-purpose container library. The various tools demonstrate a variety of techniques both dynamic (tests and runtime checking) and static (correctness proofs), which all leverage contracts to improve their effectiveness.

Bibliographic notes. *AutoTest* [4] combines techniques to improve the effectiveness of random testing: adaptive generation strategies [1], test-case minimization [3], and guided object selection for precondition satisfaction [18]. In the latest years, AutoTest has been mainly used as a supplier of test cases to support dynamic analysis of contract-equipped software [9,17,20].

AutoFix [19] takes advantage of flexible fault-localization and fix generation techniques [8], and has been evaluated on over 200 faults from different code bases [7]. SpeciFix [5] is an AutoFix component that can fix contracts instead of implementations.

AutoProof [16] supports incremental static verification by including heuristics to debug failed verification attempts [15], extensive source language support [14], and a methodology to reason about complex object dependencies [11]. It has been used to verify, among other challenges, algorithmic benchmarks [12] and the full functional correctness of a realistic, general-purpose library of data structures [10].

AutoTest, AutoFix (with SpeciFix), and AutoProof are all integrated in *EVE*, the Eiffel Verification Environment, which provides a uniform interface that integrates the results of the various analysis tools [6,13].

References

1. Ciupa, I., Leitner, A., Oriol, M., Meyer, B.: ARTOO: adaptive random testing for object-oriented software. In: 30th International Conference on Software Engineering (ICSE 2008), pp. 71–80. ACM (2008)
2. Estler, H.-C., Furia, C.A., Nordio, M., Piccioni, M., Meyer, B.: Contracts in practice. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 230–246. Springer, Heidelberg (2014)
3. Leitner, A., Oriol, M., Zeller, A., Ciupa, I., Meyer, B.: Efficient unit test case minimization. In: 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), pp. 417–420. ACM (2007)
4. Meyer, B., Fiva, A., Ciupa, I., Leitner, A., Wei, Y., Stapf, E.: Programs that test themselves. IEEE Computer **42**(9), 46–55 (2009)
5. Pei, Y., Furia, C.A., Nordio, M., Meyer, B.: Automatic program repair by fixing contracts. In: Gnesi, S., Rensink, A. (eds.) FASE 2014 (ETAPS). LNCS, vol. 8411, pp. 246–260. Springer, Heidelberg (2014)
6. Pei, Y., Furia, C.A., Nordio, M., Meyer, B.: Automated program repair in an integrated development environment. In: Companion Proceedings of the 37th International Conference on Software Engineering (ICSE). ACM (May 2015) (Demonstrations Track)
7. Pei, Y., Furia, C.A., Nordio, M., Wei, Y., Meyer, B., Zeller, A.: Automated fixing of programs with contracts. IEEE Transactions on Software Engineering **40**(5), 427–449 (2014)
8. Pei, Y., Wei, Y., Furia, C.A., Nordio, M., Meyer, B.: Code-based automated program fixing. In: Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 392–395. ACM (November 2011)
9. Polikarpova, N., Furia, C.A., Pei, Y., Wei, Y., Meyer, B.: What good are strong specifications? In: Proceedings of the 35th International Conference on Software Engineering (ICSE), pp. 257–266. ACM (May 2013)
10. Polikarpova, N., Tschannen, J., Furia, C.A.: A fully verified container library. In: Bjørner, N., de Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 414–434. Springer, Heidelberg (2015)
11. Polikarpova, N., Tschannen, J., Furia, C.A., Meyer, B.: Flexible invariants through semantic collaboration. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 514–530. Springer, Heidelberg (2014)
12. Tschannen, J., Furia, C.A., Nordio, M.: AutoProof meets some verification challenges. International Journal on Software Tools for Technology Transfer, 1–11 (February 2014). Special section on the VerifyThis 2012 Verification Competition
13. Tschannen, J., Furia, C.A., Nordio, M., Meyer, B.: Usable verification of object-oriented programs by combining static and dynamic techniques. In: Barthe, G., Pardo, A., Schneider, G. (eds.) SEFM 2011. LNCS, vol. 7041, pp. 382–398. Springer, Heidelberg (2011)
14. Tschannen, J., Furia, C.A., Nordio, M., Meyer, B.: Automatic verification of advanced object-oriented features: the autoproof approach. In: Meyer, B., Nordio, M. (eds.) LASER 2011. LNCS, vol. 7682, pp. 133–155. Springer, Heidelberg (2012)

15. Tschannen, J., Furia, C.A., Nordio, M., Meyer, B.: Program checking with less hassle. In: Cohen, E., Rybalchenko, A. (eds.) VSTTE 2013. LNCS, vol. 8164, pp. 149–169. Springer, Heidelberg (2014)
16. Tschannen, J., Furia, C.A., Nordio, M., Polikarpova, N.: AutoProof: auto-active functional verification of object-oriented programs. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 566–580. Springer, Heidelberg (2015)
17. Wei, Y., Furia, C.A., Kazmin, N., Meyer, B.: Inferring better contracts. In: Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), pp. 191–200. ACM (May 2011)
18. Wei, Y., Gebhardt, S., Meyer, B., Oriol, M.: Satisfying test preconditions through guided object selection. In: Third International Conference on Software Testing, Verification and Validation, ICST 2010, pp. 303–312. IEEE Computer Society (2010)
19. Wei, Y., Pei, Y., Furia, C.A., Silva, L.S., Buchholz, S., Meyer, B., Zeller, A.: Automated fixing of programs with contracts. In: Proceedings of the 19th International Symposium on Software Testing and Analysis, ISSTA 2010, pp. 61–72. ACM (July 2010)
20. Wei, Y., Roth, H., Furia, C.A., Pei, Y., Horton, A., Steindorfer, M., Nordio, M., Meyer, B.: Stateful testing: finding more errors in code and contracts. In: Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 440–443. ACM (November 2011)

Mind the Gap: At the Crossroads of Design, Implementation, and Foundations

Einar Broch Johnsen

Department of Informatics, University of Oslo, Oslo, Norway
einarj@ifi.uio.no

To reduce complexity, general-purpose modeling languages strive for abstraction [4]. But what is the right level of abstraction? Design-oriented models capture the logical or physical organization of software, abstracting from their dynamic behavior. Foundational models capture core features in a way suitable for meta-theory, but rely on cumbersome encodings of other features. Specifications close to actual code get obfuscated by the low-level intricacies of specific implementations.

This talk reports on research on abstract behavioral specifications [3], aiming for a middle ground in the gap between design, implementation, and foundations. We consider executable, yet abstract models which are faithful to the control and data flow of concurrent and distributed object-oriented systems, yet abstract enough to facilitate formal verification [5]. We then consider how novel technologies such as virtualization and cloud computing reintroduce low-level concerns at the abstraction level of the models. Deployment decisions form an integral part of resource-aware, virtualized applications. We discuss how these technologies raise new challenges for model-based analysis [1,2].

References

1. Albert, E., de Boer, F.S., Hähnle, R., Johnsen, E.B., Schlatte, R., Tapia Tarifa, S.L., Wong, P.Y.H.: Formal modeling of resource management for cloud architectures: An industrial case study using Real-Time ABS. Journal of Service-Oriented Computing and Applications **8**(4), 323–339 (2014)
2. Hähnle, R., Johnsen, E.B.: Resource-aware applications for the cloud. IEEE Computer (2015, to appear)
3. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: a core language for abstract behavioral specification. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) Formal Methods for Components and Objects. LNCS, vol. 6957, pp. 142–164. Springer, Heidelberg (2011)
4. Kramer, J.: Is abstraction the key to computing? CACM **50**(4), 36–42 (2007)
5. Wong, P.Y.H., Albert, E., Muschevici, R., Proença, J., Schäfer, J., Schlatte, R.: The ABS tool suite: modelling, executing and analysing distributed adaptable object-oriented systems. STTT **14**(5), 567–588 (2012)

Supported by the EU project FP7-610582 *Envisage: Engineering Virtualized Services.* http://www.envisage-project.eu

Reasoning About C Concurrency and Compilers

Francesco Zappa Nardelli

INRIA, Paris, France
francesco.zappa_nardelli@inria.fr

The C and C++ languages were originally designed without concurrency support, but the recent revision of the C and C++ standards introduced an intricate but precise semantics for threads; today's C and C++ compilers, whose optimisers were initially developed in absence of any well-defined concurrency memory model, are being extended to support this new concurrency standard. This is a fantastic opportunity to put at work all our tools to formalise, test, and reason about large scale semantics and software.

In this talk, after recalling the C and C++ memory models, we will explore in a theorem prover the correctness of compiler optimisations and present simple necessary conditions that can be used as a reference by compiler implementers. As an application, we will show how this theory enables building an automatic compiler fuzzer that hunts concurrency compiler bugs: subtle wrong code generation bugs which are observable only when the miscompiled functions interact with concurrent contexts.

Perhaps surprisingly, we will also show that by leveraging the semantics of low-level relaxed atomic accesses (which allows programmers to take full advantage of weakly-ordered memory operations), it is possible to build counterexamples to several common source-to-source program transformations (such as expression linearisation and roach motel reorderings) that modern compilers perform and that are deemed to be correct. We will evaluate a number of possible local fixes, some strengthening and some weakening the model, and perhaps conclude, that, currently, there is no really satisfactory proposal for the semantics of a general-purpose shared-memory concurrent programming language.

References

1. Batty, M., Owens, S., Sarkar, S., Sewell, P., Weber, T.: Mathematizing C++ concurrency. In: POPL 2011. http://doi.acm.org/10.1145/1926385.1926394
2. Morisset, R., Pawan, P., Nardelli, F.Z.: Compiler testing via a theory of sound optimisations in the C11/C++11 memory model. In: PLDI 2013. http://doi.acm.org/10.1145/2491956.2491967
3. Vafeiadis, V., Balabonski, T., Chakraborty, S., Morisset, R., Nardelli, F.Z.: Common compiler optimisations are invalid in the C11 memory model and what we can do about it. In: POPL 2015. http://doi.acm.org/10.1145/2676726.2676995

This talk is based on work done with Thibaut Balabonski, Soham Chakraborty, Robin Morisset, and Viktor Vafeiadis, and on discussions with Mark Batty and Peter Sewell. It is supported by the ANR grant WMC (ANR-11-JS02-011).

Contents