https://doi.org/10.1007/s00165-018-0474-2
BCS © 2018
Formal Aspects of Computing (2018) 30: 627–628

Special section of Tests and Proofs 2016

Bernhard K. Aichernig[1], Carlo A. Furia[2], Marie-Claude Gaudel[3] and Rob Hierons[4]

[1] Graz University of Technology, Graz, Austria
[2] Università della Svizzera Italiana, Lugano, Switzerland
[3] Université de Paris-Sud, Orsay, France
[4] The University of Sheffield, Sheffield, UK

This special section includes two articles extending work that was originally presented at the 2016 edition of Tests and Proofs (TAP). The TAP conference series was started in 2007 to promote research in verification and formal methods that targets the interplay of proofs and testing.

Under the traditional view, tests and proofs are strictly alternative approaches. Since proving aims at establishing correctness, whereas testing aims at uncovering errors, a correct program needs no testing, and there’s no point in trying to prove a buggy one. Research published at TAP recognizes that such a traditional view is obsolete. Testing and proving are increasingly seen as complementary rather than mutually exclusive techniques: rigorous testing can increase the confidence in the correctness of program parts that are hard to reason about formally, and proving can help make testing more efficient and systematic. Thus, combining tests and proofs means both advancing techniques of each kind and their combination, with the ultimate goal of improving software and system dependability.

The article *How Testing Helps to Diagnose Proof Failures* by Guillaume Petiot, Nikolai Kosmatov, Bernard Botella, Alain Giorgetti, and Jacques Julliand demonstrates how adding testing to an automated deductive verification process can help to debug failed verification attempts. Automated deductive verifiers, such as the WP module of Frama-C targeted by the paper, often deal with proofs of properties expressed in undecidable logic fragments. As a result, a verifier that wants to be sound is also necessarily incomplete: when a proof fails it could be either because the specification is not verifiable (the program has a bug) or simply because the proof heuristics failed to build a proof (but might succeed if given more information). Figuring out which alternative is the case is tricky and cannot be determined by proofs alone. The paper presents a technique that transparently runs automatically generated tests whenever a correctness proof fails. Thanks to the natural complementarity of tests and proofs, a failed test conclusively indicates the presence of a bug, whereas many passing tests suggest that the cause of a failed proof may be just insufficient annotations. A tool implementing the paper’s technique has been released as an open-source Frama-C plugin, and the same approach is generally applicable to any deductive verification framework where testing is possible.

The article *Tests and Proofs for Custom Data Generators* by Catherine Dubois and Alain Giorgetti addresses the problem of finding counterexamples for false conjectures in the Coq proof assistant. In modern interactive theorem proving, it is common to automatically test non-trivial conjectures before any proof attempts. However, the automatic generation of meaningful test data for complex data structures that involve invariants is non-trivial. Therefore, the testing frameworks provide mechanisms for designing custom data generators. The authors show that with the help of formal methods one can design, specify, implement and check effective data generators for complex data structures with invariants. The authors rely on random testing and bounded exhaustive testing to validate Coq definitions and conjectures and evaluate these techniques on an impressive case study including permutations and rooted maps. The case study itself is also a contribution to combinatorics and to the formalization of mathematics in Coq.

Correspondence and offprint requests to: B. K. Aichernig, E-mail: aichernig@ist.tugraz.at

For this special section, we would like to thank the authors of the papers, who painstakingly extended and improved their work following the reviewers' suggestions. The anonymous reviewers were rigorous and demanding, yet always constructive in their criticism; we thank them for their availability and punctuality. Finally, we would like to thank the editor in chief Jim Woodcock for accepting the proposal for a special section of TAP 2016 and all the technical staff of Formal Aspects of Computing who handled the practical aspects involved in the production of this special section.

Bernhard K. Aichernig
Carlo A. Furia
Marie-Claude Gaudel
Rob Hierons