Acta Informatica (2014) 51:127–128
DOI 10.1007/s00236-014-0198-6

EDITORIAL

Editorial: special issue on synthesis

Doron Peled · Sven Schewe

Published online: 19 April 2014

It is spectacular to observe, in retrospect, how deep computers have been integrated into our lives in quite a short period. This incurs a great responsibility, as their failure can result in damage or disruption of our well being. *Formal methods* are a collection of techniques for testing and verifying the correctness of software, based on logic and automata theory. In essence, formal methods can be used to verify the compatibility of a system with its formal specification. Modern formal methods provide a counterexample when this compatibility does not hold. The ability to provide counterexamples is perhaps the most important feature of such methods. It can be used to inform the designers or programmers of the problem and to locate it. However, a counterexample may not give ample information about the cause of an error, which may happen far before its affect is observed. Moreover, even when it is clear how to correct the specific problem, further errors may be detected, or even be introduced while previous ones are corrected.

Given a formal specification for the system, it is natural to try to automatically generate 'correct-by-design' code. This does not diminish the need to apply formal methods, as a complete set of properties may not be available. However, some complicated programming problems, for example mutual exclusion, have complete specifications, and the automatic synthesis of code can lift a difficult burden from the programmer.

The problem of *realizability*, i.e., the automatic synthesis from specification can be traced back to Church. Seminal solutions were provided by Rabin, and by Pnueli and Rosner. In its classical form, one is interested in constructing a system that interacts with an environment; the environment is uncontrolled and capable of making selections between different choices of response. The goal is that the system will behave in such a way that, whatever choices are made by the environment, the overall behavior satisfies the specification. One can view this as a *two player game*. The specification, often expressed in a formalism for languages, e.g.,

D. Peled (✉)
Department of Computer Science, Bar Ilan University, Ramat Gan, Israel
e-mail: doron.peled@gmail.com

S. Schewe (✉)
Department of Computer Science, University of Liverpool, Liverpool, UK
e-mail: sven.schewe@liverpool.ac.uk

temporal logic or as automata (over finite or infinite words), is the winning condition for the system. In essence, the realization of the specification is hence a *strategy* for the system to win the game.

Realization problems produce some intractability and undecidability results, which make the application of synthesis to programming challenging. New directions that makes synthesis practical are emerging. In addition to the classical synthesis problems, involving game theory, we start to see interesting new directions. The limitation of synthesis are now better understood, providing conditions under which software synthesis is plausible, and providing new algorithms. New synthesis methods are presented based on SAT solvers, heuristic search, and genetic programming. In addition, synthesis is applied to new domains, including probabilistic systems, systems with real time constraints, and systems with quantitative goals.

This volume of Acta Informatica contains four selected papers from the first SYNT workshop, held July 7th and 8th, 2012 in Berkeley California, and further contributions. SYNT 2012 was the first in a series of international workshops dedicated to the synthesis of software from formal specification. It was co-located with CAV 2012 conference. The goal of SYNT is to bring together researchers of various aspects of synthesis. The co-location with CAV brings together the formal methods and synthesis communities. The papers selected here present different aspects of synthesis: for quantitative constraints, real-time constraints, robustness conditions and formal languages. SYNT 2013 was held in St. Petersburg, and SYNT will be co-located with CAV 2014 as part of the 2014 Vienna Summer of Logic.

Doron Peled and Sven Schewe

SYNT 2012 organizers and editors of Acta Informatica Special Issue on Synthesis