Form Methods Syst Des (2018) 52:227–228
https://doi.org/10.1007/s10703-018-0318-y

Special issue: program equivalence

Ofer Strichman[1]

Published online: 23 March 2018

Program equivalence, while generally being undecidable, is arguably one of the most important problems in formal verification. While program functional correctness, which is also undecidable, has always been the 'holy grail' of formal verification, program equivalence is easier to solve in many realistic cases, which makes it a lower-hanging fruit and hence more attractive as a topic for research and tool development. Despite the fact that the latter can be reduced to the former, a simple reduction misses the opportunities to exploit the similarity between the two compared programs if it exists. One of the articles in this issue (*Automating Regression Verification of Pointer Programs by Predicate Abstraction* by Klebanov et al.) indeed shows techniques that exploit such similarities for improving the success rate of the verification engine.

Interestingly, the topic of program equivalence has attracted the interest of several research communities, which have little overlap in their conferences, and hence are largely unaware of one another. These communities range from the more theoretical field of denotational semantics, to the field of software verification, which combines theory with tool construction. The latter promotes research on methods for automated verification when possible, with tool construction (typically called software model checkers) and even annual competitions that measure their success with hundreds of test programs.

In the last 2 years there is an ongoing attempt to create a joint research community around the topic of program equivalence, based on researchers from the above-mentioned communities. So far this effort resulted in the *Workshop on Program Equivalence* that was held in *Queen-Mary university* of London in April 2016, a plan for another workshop in *Dagstuhl*, to be held in April of 2018, and this special issue on program equivalence. This issue contains three articles, that in a sense form a rather representative sample of the frontier in research on program equivalence.

The article by Klebanov, Rümmer and Ulbrich, *Automating Regression Verification of Pointer Programs by Predicate Abstraction*, is focused on a new way to perform *regression verification* (the specific case of program equivalence, where the two programs are assumed

✉ Ofer Strichman
ofers@ie.technion.ac.il

[1] Information Systems Engineering, IE, Technion, Haifa, Israel

to be structurally similar), leveraging recent progress in software verification based on solving sets of Horn formulas, and the implementation of this method in various tools. Given two functions (in a restricted fragment of ANSI-C, which is rich enough to accommodate most programs, including programs that manipulate the heap) that they wish to prove equivalent, they translate them to weakest-liberal-preconditions (wlp), and from there to Horn clauses over uninterpreted predicates. Modern SMT solvers like Eldarica or MS's Z3 that support solving constrained Horn clauses can then attempt to find a solution for the uninterpreted predicates. Z3 attempts to solve such formulas with either the Property-Directed Reachability (PDR) or the Duality strategies, which are designed originally for proving program functional correctness. One of the strategies used in Eldarica is based on *predicate abstraction*. This is an iterative 'abstraction-refinement' procedure, which begins with some seed of predicates (typically those that appear in the program), and then finds additional ones based on analyzing abstract counterexamples. If those counterexamples cannot be 'concretized', then a Craig interpolant can be computed, which serves as a new predicate. The success of such solvers highly depend on their ability to identify sufficiently strong predicates. It turns out that it can be made more effective by seeding it with various predicates that refer to the variables of the two compared functions—the authors call them *coupling predicates*—such as equivalences between mapped variables in the two compared functions. Such predicates, as they prove empirically, can expedite significantly the run time, and also increases the percentage of cases that the verification engine terminates successfully.

Akroun and Salaün, in their article *Automated Verification of Automata Communicating via FIFO and Bag Buffers*, focus on distributed programs represented as labeled transition systems. The asynchronous communication between the nodes is modeled with a FIFO queue (this model is appropriate when the order of sending and receiving messages is the same), or a bag buffer (this model is appropriate when the orderly delivery of the messages is not guaranteed). They answer fundamental questions, such as under which conditions the problem of determining the equivalence of two such asynchronous systems is decidable. They also study different types of equivalence, such as strong, weak and branching bisimulation, and trace equivalence.

Finally, the article by Murawski and Tzevelekos, *Algorithmic games for full ground references*, focuses on decidability questions related to equivalence of programs that can be modelled with a specific variant of an ML-like language that permits *ground storage*. This means that the language permits absolute addresses of variables and other addresses, and allows access to the content of a given address, like the de-referencing operator in the C programming language. It turns out that under specific restrictions on the grammar, equivalence of terms in the language becomes decidable.

With this special issue we hope to disseminate ideas, encourage collaborations between researchers that currently consider themselves as being in separate communities, and draw the attention of the larger verification community to this challenging problem and the research activity that it attracts.