Managing Change in Formal Software Analysis: Two Research Challenges

Reiner Hähnle

Technische Universität Darmstadt
Department of Computer Science
`haehnle@cs.tu-darmstadt.de`

In the field of formal analysis of software systems dramatic progress could be witnessed in the last decade or so. Formal verification of simple safety properties often can be achieved in a fully automated manner for non-trivial, commercial code [3]. It can be expected that the verification of generic safety properties (for example, buffer under-/overflows, null pointer accesses) will soon become part of the compilation tool chain. Changing compilation targets tend to be managed simply by recompilation, so: where is the need for managing change in formal software analysis? Let us try to give an answer in the form of two research challenges.

Functional Verification. In contrast to checking of safety properties, functional verification still requires vast efforts. True, even highly complex system software can be formally verified when sufficient effort is spent, as is demonstrated (for example) by the L4.verified [5] and Verisoft [2] projects. However, this typically requires serious effort (ranging between several person months and person decades), involving formal specification and verification specialists. Even if some effort can be spent, it remains a central problem of current verification methods that they are not robust in presence of changes in the verification target: already small changes can invalidate large parts of an existing verification argument and might require to redo much of the verification effort. But, as clearly spelled out in the *FoMaC Manifesto*[6], change occurs continually during software development, and formal methods must cope with it, if they are to be relevant. Based on current technology, formal verification is simply far too disruptive in the context of agile development processes and can only be considered as a *post hoc* activtity. This renders functional verification impractical for any application outside extremely safety-critical systems. Therefore, we pose the following challenge:

> *Make formal specification and verification non-disruptive in the presence of changes in the verification target. Specifically, integrate formal verification with standard change management mechanisms such as version control, regression testing, etc.*

As one recent example of the line of work we have in mind, consider [4], which presents a technique for compositional, contract-based, formal verification in presence of constant evolutionary changes to the verification target.

Resource Analysis. Resource analysis is typically fully automatic and requires little or only generic specifications [1]. This makes it a highly interesting and useful alternative to full verification in case when the latter cannot be achieved or is too expensive. But resource analysis became a central issue also for a different reason: requirememts and code are not the only aspects of a system that can change. In recent years, the need to deal with dynamically changing computing environments, i.e., resources, became a pivotal issue. This is caused by three nearly parallel developments: the move from single- to multi-core architectures to compensate for the breakdown of Moore's law; the advent of cloud computing technology that renders available resources highly elastic; and the availability of sensing, computing, and networking capabilities in just any kind of technical artifact (*"cyber-physical system"*). As a consequence, modern software must be able to take advantage of different resource profiles that even might change dynamically during execution. To this end it is of prime importance to be able to analyze and optimize the resource consumption of software. This goes far beyond the classical topics of worst-case execution time and memory allocation, but includes parameters such as bandwidth, latency, degree of parallelism, and, of growing importance, energy consumption. It must be possible to connect the results of resource analysis with optimization of system configurations and, ultimately, with dynamic adaptation mechanisms. Hence, our second research challenge:

> *Extend resource analysis to a comprehensive set of environmental parameters including network and energy aspects. Make analysis methods incremental and change-aware. Use analysis results to generate optimized system configurations. Align these with dynamic software adaptation mechanisms with the aim of optimized resource consumption in a dynamically changing environment.*

References

1. E. Albert, P. Arenas, A. Flores-Montoya, S. Genaim, M. Gómez-Zamalloa, E. Martin-Martin, G. Puebla, and G. Román-Díez. SACO: static analyzer for concurrent objects. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 20th Intl. Conf., Part of the European Joint Conferences on Theory and Practice of Software, ETAPS, Grenoble, France*, volume 8413 of *LNCS*, pages 562–567. Springer, 2014.
2. E. Alkassar, M. A. Hillebrand, W. J. Paul, and E. Petrova. Automated verification of a small hypervisor. In G. T. Leavens, P. W. O'Hearn, and S. K. Rajamani, editors, *Verified Software: Theories, Tools, Experiments, Third Intl. Conference, VSTTE, Edinburgh, UK*, volume 6217 of *LNCS*, pages 40–54. Springer-Verlag, 2010.
3. D. Beyer. Status report on software verification - (competition summary sv-comp 2014). In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 20th Intl. Conf., Part of the European Joint Conferences on Theory and Practice of Software, ETAPS, Grenoble, France*, volume 8413 of *LNCS*, pages 373–388. Springer, 2014.

4. R. Bubel, R. Hähnle, and M. Pelevina. Fully abstract operation contracts. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, LNCS. Springer, Oct. 2014. In this proceedings.
5. G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: Formal verification of an operating system kernel. *Communications of the ACM*, 53(6):107–115, June 2010.
6. B. Steffen. LNCS transactions on foundations for mastering change: Preliminary manifesto. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium, ISoLA 2014, Corfu, Greece*, volume 8802 of *LNCS*, pages 509–512. Springer, Oct. 2014.