Compositional Model Checking

Orna Grumberg
The technion, Haifa, Israel

Modern software, which is often concurrent and distributed, must be extremely reliable and correct. Model checking [3] is a technique for automating high-quality assurance of software. Given a finite state model of a system and a property, usually expressed as an automaton or a temporal logic formula, model checking systematically goes through all the possible system behaviors and checks them for conformance against the property. Despite its successes, the technique still suffers from the *state explosion problem*, which refers to the worst-case exponential growth of a program's state space with the number of concurrent components. *Compositional verification* techniques have shown promise in addressing this problem, by breaking-up the global verification of a program into local, more manageable, verification of its individual components.

Assume-Guarantee (AG) reasoning [7, 9] provides solutions to the problem of decomposing the verification of a large system into local verification steps of the system components. In the assume-guarantee paradigm we prove that whenever M_1 is part of a system satisfying an *assumption* A, then the system also guarantee the *property* P. We further prove that M_2 satisfies assumption A. We then conclude that $M_1 \| M_2$ satisfies property P.

The most challenging part of applying assume-guarantee reasoning, is coming up with appropriate assumptions to use in the application of the assume-guarantee rules. In [4, 8] *learning techniques* have been proposed for automating the generation of assumptions. The framework uses the L^* [2] automata-learning algorithm to iteratively compute assumptions in the form of deterministic finite-state automata.

Another important category of rules involve *circular reasoning* and use inductive arguments, over time, formulas to be checked, or both, e.g. [5, 6]. Such rules can naturally exploit the inherent circular dependency exhibited by the verified system and may result in smaller assumptions. In [1] a *circular* compositional verification rule was fully automated, by iteratively computing two assumptions g_1 and g_2 for M_1 and M_2, which are mutually dependent.

In this series of talks we describe frameworks for automating Assume-Guarantee (AG) rules. We first present the learning-based AG framework of [4]. We then describe the circular AG framework of [1].

References

[1] K. AbdElkader, O. Grumberg, C. S. Pasareanu, S. Shoham. *Automated Circular Assume-Guarantee Reasoning*. In: Procs of the FM; 2015.

[2] D. Angluin. *Learning Regular Sets from Queries and Counterexamples*. Inf. Comput.; Vol. 75(2); pp. 87–106; 1987.

[3] E. Clarke, O. Grumberg, D. Peled. *Model Checking*. MIT press; 1999.

[4] . M. Cobleigh, D. Giannakopoulou, C. S. Pasareanu. *Learning Assumptions for Compositional Verification*. In: Procs of the TACAS; pp. 331–346; 2003.

[5] K. L. McMillan. *Verification of an Implementation of Tomasulo's Algorithm by Compositional Model Checking*. In: Procs of the CAV; pp. 110–121; 1998.

[6] K. L. McMillan. *Circular Compositional Reasoning about Liveness.* In: Procs. of the CHARME; pp. 342–345; 1999.

[7] J. Misra, K. M. Chandy. *Proofs of Networks of Processes.* IEEE Trans. Software Eng.; Vol. 7(4); pp. 417–426; 1981.

[8] C. S. Pasareanu, D. Giannakopoulou, M. G. Bobaru, J. M. Cobleigh, H. Barringer. *Learning to Divide and Conquer: Applying the L^* Algorithm to Automate Assume-Guarantee Reasoning.* Formal Methods in System Design; Vol. 32(3); pp.175–205; 2008.

[9] A. Pnueli. *In Transition from Global to Modular Temporal Reasoning about Programs.* In: Logics and Models of Concurrent Systems; NATO ASI Series; 1985.