Formal Software Verification: How Close Are We?

Gerard J. Holzmann

Laboratory for Reliable Software
Jet Propulsion Laboratory, California Institute of Technology
M/S 301-230, 4800 Oak Grove Drive, Pasadena, CA 91109
gholzmann@acm.org

Abstract. Spin and its immediate predecessors were originally designed for the verification of data communication protocols. It didn't take long, though, for us to realize that a data communications protocol is just a special case of a general distributed process system, with asynchronously executing and interacting concurrent processes. This covers both multi-threaded software systems with shared memory, and physically distributed systems, interacting via network channels.
The tool tries to provide a generic capability to prove (or as the case may be, to disprove) the correctness of interactions in complex software systems. This means a reliable and easy-to-use method to discover the types of things that are virtually impossible to detect reliably with traditional software test methods, such as race conditions and deadlocks.
As initially primarily a research tool, Spin has been remarkably successful, with well over one million downloads since it was first made available by Bell Labs in 1989. But our goal is te development of a tool that is not only grounded in foundational theory, but also usable by all developers of multi-threaded software, not requiring specialized knowledge of formal methods.
In this talk we try to answer the question how close we have come to reach these goals, and where especially we are still lacking. We will see that our understanding has changed of what a verification tool can do – and what it *should* do.

Key words: software verification, software analysis, concurrency, model checking, software testing, static source code analysis.