



Formal Software Verification: How Close Are We?

Gerard Holzmann

► **To cite this version:**

Gerard Holzmann. Formal Software Verification: How Close Are We?. John Hatcliff; Elena Zucca. Joint 12th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS) / 30th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE), Jun 2010, Amsterdam, Netherlands. Springer, Lecture Notes in Computer Science, LNCS-6117, pp.1, 2010, Formal Techniques for Distributed Systems. .

HAL Id: hal-01055211

<https://hal.inria.fr/hal-01055211>

Submitted on 11 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

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 the 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.