Scalable Techniques for Formal Verification

Sandip Ray

Scalable Techniques for Formal Verification

Dr. Sandip Ray
Department of Computer Sciences
University of Texas, Austin
University Station 1
78712-0233 Austin Texas
MS C0500
USA
sandip@cs.utexas.edu

ISBN 978-1-4419-5997-3 e-ISBN 978-1-4419-5998-0
DOI 10.1007/978-1-4419-5998-0
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010927798

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To Anindita
Who showed me that sometimes dreams can come true

Preface

This book is about *formal verification*, that is, the use of mathematical reasoning to ensure correct execution of computing systems. With the increasing use of computing systems in safety-critical and security-critical applications, it is becoming increasingly important for our well-being to ensure that those systems execute correctly. Over the last decade, formal verification has made significant headway in the analysis of industrial systems, particularly in the realm of verification of hardware. A key advantage of formal verification is that it provides a mathematical guarantee of their correctness (up to the accuracy of formal models and correctness of reasoning tools). In the process, the analysis can expose subtle design errors. Formal verification is particularly effective in finding corner-case bugs that are difficult to detect through traditional simulation and testing. Nevertheless, and in spite of its promise, the application of formal verification has so far been limited in an industrial design validation tool flow. The difficulties in its large-scale adoption include the following (1) deductive verification using theorem provers often involves excessive and prohibitive manual effort and (2) automated decision procedures (*e.g.*, model checking) can quickly hit the bounds of available time and memory.

This book presents recent advances in formal verification techniques and discusses the applicability of the techniques in ensuring the reliability of large-scale systems. We deal with the verification of a range of computing systems, from sequential programs to concurrent protocols and pipelined machines. Each problem is different, bringing in unique challenges for verification. We show how to ameliorate verification complexity by applying formal analysis judiciously within a hierarchical, disciplined reasoning framework.

The research documented here originally appeared as the author's Ph.D. dissertation. The research owes its existence more to skepticism about the possibility of mathematical analysis at this scale than to anything else. Coming from a background in theory of computation and armed with traditional results in computability and complexity theory, I approached the area with extreme suspicion: how can techniques that are so severely handicapped with undecidability, incompleteness, and (in the better scenarios) lower bounds of computational complexity be successful in practice? The answer to the paradox that I found in my own research is that formal reasoning should not be viewed as a stand-alone tool but rather an integral part of the system design process. In particular, formal reasoning thrives when

it serves to make explicit the intuitions already available to the system designers, helping them effectively articulate the informal justifications behind optimizations and design tweaks. A well-designed reasoning framework permits specification of a problem at a level that matches the intuition of the designer and this turns the analysis into a vehicle for making the intuitive justifications rigorous. Intuitive justification and mathematical rigor must go hand-in-hand, each clearing the way of the other.

Most treatises and research articles on formal verification do not focus on the role of analysis as part of the system design process. Indeed, verification research today is essentially single tracked: the mantra is to develop algorithms and heuristics for providing more automation in the analysis of larger and larger systems. Automation is certainly critical to large-scale system verification. However, with design complexity is increasing faster than automation heuristics can catch up, improvement in raw automation alone is insufficient: one must additionally have disciplined, hierarchical, and customized reasoning infrastructures for different domains. Such infrastructures must permit (1) decomposition of a complex problem into tractable pieces, (2) effective management and control of the overall verification and decomposition, and (3) the ability of different, customized, automated tools to bear upon each individual piece. Furthermore, the act of developing a reusable framework forces thinking about a verification problem at an appropriate level of abstraction. Note that an "abstraction" *does not* mean a "toy." Rather, the abstraction must capture the essence of the class of systems being analyzed (often by generalization), while eliminating irrelevant complexities. Admittedly, developing a generic, compositional framework for a specific target domain is not easy. Thinking about a problem in a generic manner requires creativity, experience, and practice. However, the cost of doing so is ameliorated by the fact that the same framework can be used over and over on different problems. I have found the act of developing careful, generic frameworks for different problem domains to be both instructive and fruitful, as it exposes the underlying structure of the problems, helps spot connections between various subproblems, and thereby leads to the identification of the verification strategy most suited for the problem at hand. The research presented in this book shows numerous examples of generic verification strategies for different problem domains. The infrastructures described here are intended to serve as a guide to formal verification researchers and practitioners, who intend to pursue mathematical reasoning for large-scale systems. The book is also intended to show system designers how mathematical reasoning can serve as a helpful practical tool for ensuring reliable execution of the systems they design.

This research owes its existence to a number of people, including my teachers, collaborators, friends, and family. Perhaps the biggest debt goes to my Ph.D. supervisor J. Strother Moore for giving me the right balance of freedom, encouragement, and direction to guide the course of this research. Some other key players are (in alphabetical order) Jeff Golden, Warren A. Hunt, Jr., Matt Kaufmann, John Matthews, Erik Reeber, Fei Xie, and Thomas Wahl.

The research presented in this monograph has been supported over the years in part by funding from the National Science Foundation under Grants ISS-041741,

CCF-0916772, and CNS-0910913, by the Defense Advanced Research Projects Agency and National Science Foundation under Grant CNS-0429591, and by the Semiconductor Research Corporation under Grants 02-TJ-1032 and 08-TJ-1849. I am grateful to these agencies for the support. However, any opinions, findings, conclusions, or recommendations expressed herein are mine and do not necessarily reflect the views of any of these agencies.

Finally, I thank my wife Anindita for bearing with me through all the late hours that went behind the book. This book would not have seen the light of the day without her understanding and support.

Austin, TX Sandip Ray
November 2009

Contents