Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

129

Brent T. Hailpern

Verifying Concurrent Processes Using Temporal Logic

Springer-Verlag
Berlin Heidelberg New York 1982

Author

Brent T. Hailpern
IBM Thomas J. Watson Research Center
P.O.Box 218, Yorktown Heights, NY 10598, USA

CR Subject Classifications (1981): 5.21 5.24

ISBN 3-540-11205-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-11205-7 Springer-Verlag New York Heidelberg Berlin

Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

Abstract

Concurrent processes can exhibit extremely complicated behavior, and neither informal reasoning nor testing is reliable enough to establish their correctness. In this thesis, we develop a new technique for the verification of parallel programs. The technique is stated in terms of axioms and inference rules, and it is used to prove safety and liveness properties of parallel programs. Safety properties are assertions that must be satisfied by the system state at all times; they are analogous to partial correctness. Liveness properties refer to events that will occur in the future, such as program termination or the eventual receipt of a message. In addition to the formal proof rules, we present several heuristics to aid in the preparation of correctness proofs.

We model a parallel program as a set of interacting modules (processes and monitors), and we exploit this modularity in the verification process. First we prove properties of the low-level modules directly from their code. We then combine the specifications of the low-level modules to prove properties of higher-level modules, without again referring to the code. Eventually, we prove properties of the entire program.

We discuss the application of this verification technique to two classes of parallel programs: network protocols and resource allocators. Most previous approaches to verifying network protocols have been based upon reachability arguments for finite-state models of the protocols. Only protocols of limited complexity can be verified using the finite-state model, because of the combinatorial explosion of the state space as the complexity of the protocol increases. In contrast, our approach allows us to abstract information from the details of the implementation, so that the proof need not grow unmanageably as the protocol size increases.

The discussion of resource allocation centers around Hoare's structured paging system, which is a complex hierarchical program. With this example, we demonstrate that many of the techniques used in program verification can be used for specification as well.

The thesis also describes a number of tools that have been useful in proving concurrent programs. Two of the most important are history variables and temporal logic. We employ history variables to record the interaction between the modules that constitute a program. Temporal logic serves as a convenient notation for stating and proving liveness properties.

Acknowledgments

Years ago, my parents taught me to love learning. I thank them for that lesson and for their love; for without either one, this thesis would never have been written.

Many people at Stanford contributed in one way or another to this thesis. John Hennessy and Gio Wiederhold, as members of my reading committee, provided me with many useful comments and suggestions. John Gilbert, Jim Boyce, and Shel Finkelstein helped me understand the beauty of formal logic in general and temporal logic in particular. Robert Tarjan guided me through my first year at Stanford with his wisdom and his inexhaustable common sense. To these people and to the faculty, staff, and students of the Stanford Computer Science Department, I express my gratitude.

I want give special thanks to two of my dearest friends: David Wall and Richard Pattis. They provided constant professional and emotional support during my entire graduate career. The two of them were always there when I needed advice, someone to listen to me complain, a sounding board, criticism, or companionship. They contributed greatly to this thesis by reading various drafts and providing numerous comments.

There are no words to express my gratitude to Susan Owicki, my advisor. She introduced me to program verification and to temporal logic. Her ideas, comments, advice, and suggestions form an integral part of this thesis. It has been a great honor to have known her and to have worked with her.

I dedicate this thesis to my wife, Susan. Her love, support, and understanding gave me the strength to write this thesis. I am in her debt for the many hours I spent working when we could have been together, and I look forward to spending the rest of my life discharging that debt.

My research was supported by a number of agencies, and I gratefully acknowledge their generosity. My graduate education was funded primarily by fellowships from the National Science Foundation and the Fannie and John Hertz Foundation. I received additional support from teaching assistantships in the Stanford Computer Science Department and Computer System Laboratory and from research assistantships with the S-1 project and my advisor. (The S-1 project is supported at Lawerence Livermore Laboratory of the University of California by the Department of the Navy via ONR Order No. N00014-78-F0023. Research with my advisor was supported by the Joint Services Electronics Project, under contract N-00014-75-C-0601. JSEP also funded my travel expenses in connection with this research.)

Table of Contents

List of Figures