Lecture Notes in Computer Science 531

Edited by G. Goos and J. Hartmanis

Advisory Board: W. Brauer D. Gries J. Stoer

E. M. Clarke R. P. Kurshan (Eds.)

Computer-Aided Verification

2nd International Conference, CAV '90
New Brunswick, NJ, USA, June 18-21, 1990
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Volume Editors

Edmund M. Clarke
School of Computer Science, Carnegie-Mellon University
Pittburgh, PA 15213, USA

Robert P. Kurshan
AT&T Bell Laboratories, Mathematical Science Research Center
Murray Hill, NJ 07974, USA

CR Subject Classification (1991): C.2.2, C.3, D.2.4, F.3-4

ISBN 3-540-54477-1 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54477-1 Springer-Verlag New York Berlin Heidelberg

Typesetting: Camera ready by author
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Preface

This volume is devoted to the proceedings of the second workshop on **Computer-Aided Verification** (the first to use this specific title). The motivation for a workshop in computer aided verification is to bring together researchers working on effective algorithms or methodologies for formal verification (as distinguished, say, from attributes of logics or formal languages). The considerable interest generated by the first workshop (held in Grenoble, June 1989; see LNCS 407) motivated this meeting. As interest continued to grow, it was decided to hold CAV on an annual basis, and for that purpose the CAV Steering Committee was formed (see below). In view of this, we take the opportunity here to state the focus of CAV and briefly sketch the history of research in formal verification, as it relates to CAV.

It is the intention of the CAV Steering Committee that future CAV meetings will continue the current focus on the problem of making formal verification feasible for various models of computation. Present emphasis is on models associated with distributed programs, protocols, and digital circuits. A good test of algorithm feasibility is to embed it into a verification tool and exercise that tool on realistic examples. Thus, we have promoted special sessions for the demonstration of new verification tools. For the technical sessions, we seek theoretical results that lead to new or more powerful verification methods. We expect there will be less emphasis at CAV meetings on purely theoretical results in program logics – not because fundamental results of this type are unimportant but because this research is adequately covered in other conferences. Since we expect that a number of the results presented at CAV actually will be used by hardware and software designers, we seek to maintain a balance between presentations by researchers and practitioners.

Proofs of correctness of algorithms such as the Euclidean algorithm go far back into history. The importance of such proofs in computing apparently was realized by Turing. However, it was not until the 1960s and early 1970s that provably correct computation began to attract much attention as a self-contained area of research. Fundamental contributions in this period established the vehicles through which formal proofs of program correctness could be constructed from axioms and rules of inference in the same way that proofs in mathematics are constructed.

The first proofs of program correctness were hand constructed and, therefore, quite short and easy to follow. As a general approach, however, manually constructed proofs of correctness were beset by two fundamental problems. First was the problem of scalability: real programs tend to be long and intricate (compared with the statement of a mathematical theorem), so a proof of correctness could be expected to be correspondingly long. Under these circumstances, it was unclear to what extent a methodology based upon manually constructed proofs could be expected to be successful. The second problem was credibility: unlike published mathematics which may be expected to undergo extensive peer review, proofs of programs are more likely to be read only by the author. Much interesting work has continued in this direction, however, and through the mid-1980s most of the research on formal verification (as this area of research became known) remained focused upon manual proofs of correctness. Applications of the work did not overcome these two fundamental problems.

The purpose of CAV is to feature research specifically directed at overcoming these two problems of scalability and credibility. Presently, this thrust has become synonymous with computer-aided verification.

Initially, researchers thought that computer programs for theorem-proving could be used in automatic program verification. Logics emerged as a mechanism to formalize the manipulation of the properties to be proved. Out of fundamental work in logic and mathematics, automatic theorem-proving advanced rapidly. Automated theorem-proving had its own problems, however, stemming largely from its non-algorithmic nature, and basic problems of tractability and decidability. These difficulties seemed to provide obstacles which were much too difficult for early theorem-provers to overcome. Many of the pioneers in program verification became disillusioned and moved into other research areas where progress was more rapid.

Recent advances and the maturation of theorem-provers (and more specifically, proof-checkers) has renewed interest in their application to practical tasks such as hardware verification. Currently, this direction has demonstrated applicability for proving properties of data paths in hardware designs. Theorem-proving has been less successful in verifying properties related to *control*, particularly when concurrency and process synchronization are involved.

Together with the early disillusionment in theorem-provers emerged an intense interest in restricted logics for which formula satisfiability is decidable. Pre-eminent among these logics was propositional temporal logic.

However, this work had two significant deficiencies of its own. First, these logics invariably constituted a substantial abstraction of a restricted class of programs. In fact, abstraction was so great that formulas in the logic lost their connection to the programs they were meant to abstract. Second, as a purely computational matter, decision procedures still were largely intractable, being exponential in the size of the formulas.

This second problem was undercut in 1980 through the introduction of model-checking as an alternative to checking formula satisfiability. Not only was linear-time model-checking demonstrated (for branching-time temporal logic), but perhaps the first computer implementations of practical formal verification algorithms were produced, as well. Computational complexity nonetheless remained an issue: while model-checking could be done in time linear in the size of the model, the model itself grows exponentially in the number of model components; for 'real' models, complexity still was the gating issue.

This problem and the problem of bridging the gap from model to implementation were addressed soon after through the introduction of homomorphic reduction. This permitted checking complex models indirectly through checks on reductions which are relative to the respective properties under test. Homomorphism also served as a mechanism for stepwise refinement, relating implementations to design models. This led to compositional and hierarchical verification, as well as specialized reduction methods involving certain types of induction. Complexity still remained an issue, however, as homomorphic reductions may be difficult (or impossible) to produce, especially in the case of large data path models; even small data path models with many inputs are not made readily tractable by homomorphic reduction. The same difficulties applied in some degree to induction.

Significant inroads into these difficulties have been made through the work on symbolic model checking using binary decision diagrams (BDDs), as introduced at the first session of this workshop in Grenoble (1989). While not scalable (effective use of BDDs currently appears to be limited to around 150 binary variables, while applications often require thousands), use of BDDs in conjunction with homomorphic reduction and

induction appears extremely promising, and has generated considerable excitement. Introduced at this same meeting was work on real-time verification, which continues to generate much interest, as well.

The current workshop (CAV'90) presents some very substantial progress using BDDs. In addition, this workshop introduces reductions based upon partial order representations; these reductions offer new potential for dealing with the tractability problem inherent in state-based models. Other papers presented here explore theorem-proving and proof-checking in controller verification, and the remaining papers present much vital work which continues and extends current verification methods.

We have witnessed a migration from theorem-proving to model-checking, and a recent renewed interest in proof-checking. This may represent the start of a swing back toward theorem-proving, especially through the combination of model-checking and symbolic techniques. If so, we may expect more general theorem-proving to become integrated into existing verification tools, providing a basis for static and dynamic reasoning from the same platform.

For example, verification of a (dynamic) property of a model through expansion of the model state space or BDD evaluation may be simplified by exploiting a symmetry or inductive property in the model; the symmetry or inductive property upon which the simplification is based may be verified through a (static) syntactic check on the model specification, using theorem-proving techniques.

Whatever the future may hold, our perception is that computer-aided verification has emerged from adolescence into a very exciting and promising adulthood.

Three more CAV workshops have been planned; the next will be held in Aalberg, Denmark in July 1991 under the direction of Kim Larsen.

These proceedings are derived from *Computer-Aided Verification '90*, E. M. Clarke, R. P. Kurshan (eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science 3, American Mathematical Society/Association for Computing Machinery, 1991, which contains the full versions of the papers originally presented at CAV'90.

We would like to thank the other members of the Steering Committee and the members of the CAV'90 Program Committee for their invaluable help in making this workshop a

success. We are appreciative to DIMACS for their sponsorship of the workshop and the contribution of their facilities at Rutgers University for the duration of the workshop (June 18-21). In particular, we thank Pat Toci of DIMACS for her extensive work organizing and managing the entire on-site logistics of the workshop, and we thank Thelma Pickell of AT&T Bell Laboratories for administrative management of the meeting.

Rutgers University
New Brunswick, NJ
May 1991

E. M. Clarke
R. P. Kurshan
Program Co-Chair
CAV'90

Contents