Costas Courcoubetis (Ed.)

# Computer Aided Verification

5th International Conference, CAV '93
Elounda, Greece, June 28-July 1, 1993
Proceedings





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

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
D-76131 Karlsruhe, FRG

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Volume Editor

Costas Courcoubetis
Department of Computer Science, University of Crete
and
Institute of Computer Science, FORTH
P. O. Box 1385, GR-71110 Heraklion, Crete, Greece

CR Subject Classification (1991): F.3, D.2.4, B.7, C.2.2

ISBN 3-540-56922-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-56922-7 Springer-Verlag New York Berlin Heidelberg


Printed in Germany

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

# Preface

This volume contains the proceedings of the Fifth Conference on Computer-Aided Verification (CAV'93), held in Elounda, Crete, Greece, from June 28 to July 1, 1993.

The objective of the CAV conferences is to bring together researchers and practitioners interested in the development and use of methods, tools and theories for the computer-aided verification of concurrent systems. The conferences provide an opportunity for comparing various verification methods and tools that can be used to assist the applications designer. Emphasis is placed on new research results and the application of existing methods to real verification problems.

Of the 84 submitted papers, 37 were accepted for presentation. Invited talks were given by B. Brayton (UC Berkeley), M. Gordon (Cambridge University), and P. Varaiya (UC Berkeley). The first day of the conference was dedicated to tutorials on real-time formalisms by R. Alur (AT&T Bell Laboratories), D. Dill (Stanford University), T. Henzinger (Cornell University), and partial order verification methods by P. Wolper (University of Liege). Besides the areas of real-time verification (which this year included results about the new formalism of hybrid systems) partial order methods and hardware verification where the conference has been traditionally strong in the past, there were some new themes which emerged in this year's conference. These themes are of vital importance for advancing the state-of-the-art in computer-aided verification and include the combination of model-checking with theorem proving techniques, and the exploitation of symmetry in verification methodologies. There were sessions devoted to hardware verification, theorem proving, real-time formalisms, process algebras and calculi, partial order methods, the exploitation of symmetry, and other verification methods and tools.

Financial support is provided among others by the Institute of Computer Science, FORTH, the University of Crete, and mainly by the Commission of the European Communities, Directorate-General XIII, ESPRIT program. Many research projects funded by ESPRIT Basic Research contributed a large number of high-quality papers to this conference.

The Program Committee was very active in reviewing and shaping the final program. The Steering Committee, consisting of E.M. Clarke (Carnegie Mellon University), R.P. Kurshan (AT&T Bell Laboratories), A. Pnueli (Weizmann Institute), and J. Sifakis (VERIMAG), took part in the reviewing process as well and offered council at appropriate moments. This year, the Program Committee members were: R. Alur (AT&T Bell Labs), G. Bochmann (U. Montreal), R. Brayton (UC Berkeley), E. Brinksma (U. Twente), R. Cleaveland (North Carolina State U.), W. Damm (Oldenburg U.), R. de Simone (INRIA), D. Dill (Stanford U.), A. Emerson (UT Austin), O. Grumberg (Technion), N. Halbwachs (VERIMAG), H. Hiraishi (Kyoto Sangyo U.), G. Holzmann (AT&T Bell Labs), K. Larsen (Aalborg U.), P. Loewenstein (Sun), L. Paulson (Cambridge

U.), D.K. Probst (Concordia U.), A. Sangiovanni-Vincentelli (UC Berkeley), B. Steffen (TU Aachen), C. Stirling (Edinburgh U.), P. Wolper (U. Liege) and T. Yoneda (Tokyo Inst. of Tech.).

Costas Courcoubetis is General and Program Chair. MITOS SA is responsible for the local arrangements, registration, and the treasurer functions. Liana Kefalaki is the Conference Secretary and Magda Hadzaki is the assistant to the Program Chair. P. Godefroid (Liege U.) assisted in the preparation of the tutorial on partial order verification methods. R. Schapire (AT&T Bell Labs) provided his program for compiling the electronic scorecards of the reviewers to produce the final reports.

A partial list of additional referees (besides the steering and program committee members) is the following: T. Henzinger (Cornell U.), D. Peled (AT&T Bell Labs), D. Long, X. Zhao, W. Marrero, S. Jha, S. Campos (Carnegie Mellon U.), O. Bernholz (Technion), G. Bruns, H. Huttel, R. Kaivola, F. Moller (Edinburgh U.), J. C. Madre (Bull), P. Godefroid (Liege U.), M. Kaltenbach, P. Attie (UT-Austin), T. Nipkow (TU Munich), F. Balarin, T. Shiple, A. Aziz, R. Hojati, S. Krishnan, Y. Kukimoto (UC Berkeley), N. Ip (Stanford U.), A. Bouajjani, S. Yovine, A. Olivero, S. Graf, L. Mounier, J-C. Fernandez (VERIMAG), O. Burkart, A. Kindler, C. Weise (RWTH Aachen), K. Hamaguchi (Kyoto U.), S. Kimura (UAIST, Nara), H. Eertink, P. Kars, R. Langerak, L. Pires (U. Twente), A. Ingolfsdottir, J. C. Godskesen, A. Skou (Aalborg U.).

Heraklion, May 1993 — Costas Courcoubetis

# Table of Contents

**Session 7: Process Algebras and Calculi**



**Session 8: Partial Orders**



**Session 9: Exploiting Symmetry**

# Logic Synthesis and Design Verification

Robert K. Brayton

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720



Much work has been done over the last 7-8 years on the automatic synthesis of combinational logic. This work has found its way into the software of successful companies whose programs are found almost everywhere digital circuits are designed. More recently, research has focused on the automatic synthesis of sequential networks.

In combinational logic, the model of computation used is a Boolean network which is a multi-rooted DAG where each node is an arbitrary logic function. An arc is drawn from node $i$ to node $j$ if node $j$ explicitly uses the result of the node $i$ computation, node $i$'s output. For sequential logic, the model of computation is a FSM network, an arbitrary graph where each node is a FSM with symbolic inputs, outputs, and internal states. One nodes outputs are another's inputs.

One of the paradigms for logic synthesis is to focus on a single node and to simplify it as much as allowed. Each node is then iteratively simplified until there is no change at any node. At this point, the logic is at some minimal representation. Such a paradigm works quite well for combinational logic now, although it took many years to work out the theory and practice of using the full capabilities of node minimization at each node. One of the problems was to simply derive and represent all the flexibility that is allowed at any node, i.e. all the "permissible" functions at a node.

Recently, these ideas have been extended to FSM networks. It has been shown that the "permissible" FSM's at a node can be represented with a single nondeterministic FSM, where a permissible FSM is a completely specified deterministic FSM that is simulated by the NDFSM. Then one form of node minimization is to select the permissible machine with the least number of states - a hard problem, but possibly one where good heuristics can be developed. Unfortunately, the derivation of the NDFSM involves a subset construction, so here again heuristics need to be employed.

The FSM network model is exactly the same one that is used in some forms of design verification - e.g. L-processes. Indeed in the FSM network it is allowed to have nodes with nondeterministic behavior, perhaps representing part of the environment. Design verification deals with proving that the FSM network satisfies certain properties, $P_1, \ldots, P_n$. Using L-automata, each of these (if $\omega$-regular) can be represented by a finite set of deterministic L-automata, $P_{i1}, \ldots, P_{ik}$. If "complete", the entire set can be taken as our specification. This is analogous to the "observability relation" used in combinational logic, which specifies what input-output behavior is allowed. Such an observability relation is then used to derive a local environment for a single node in the Boolean network. One can

speculate that a similar theory should hold for the FSM network. Other useful parallels hold between the two areas of logic synthesis and design verification.

The purpose of this talk is to survey these parallels, with the idea that an improved understanding of both areas leads to common ideas and tools and gives guidelines about fruitful avenues of research to pursue.

# Efficient Verification with BDDs using Implicitly Conjoined Invariants *

Alan J. Hu and David L. Dill

Department of Computer Science, Stanford University

**Abstract.** Many researchers have reported that using Boolean decision diagrams (BDDs) greatly increases the size of hardware designs that can be formally verified automatically. Our own experience with automatic verification of high-level aspects of hardware design, such as protocols for cache coherence and communications, contradicts previous results; in fact, BDDs have been substantially inferior to brute-force algorithms that store states explicitly in a table.
We believe that new techniques are needed to realize the potential advantages of BDD-based verification at the protocol level. Here, we isolate several common causes of BDD-size blowup and show how these problems can be alleviated by a new verification approach based on partially evaluating the invariant being checked into an implicit conjunction of small BDDs. We describe the new method and give several examples of its application.

## 1 Introduction

With the increasing cost and complexity of hardware designs and protocols, formal verification techniques become ever more attractive. Boolean decision diagrams (BDDs) [3] have enabled much progress in this area, from the early work applying BDDs to verification [1, 6, 5, 8, 19] through the current work of numerous researchers.

Most of the current research on automatic formal hardware verification has focused on gate and transistor-level design. We believe that automatic formal verification also has an important role at the very high-level of design, for example, in checking communications and consistency-maintenance protocols in a very large system (e.g. a multiprocessor). Verification of high-level, abstract specifications can catch *conceptual errors* early in the design cycle, when they are easier and cheaper to correct.

We have developed the Mur$\varphi$ verification system specifically for this application domain, refining our design by verifying large, real examples (e.g. industrial multiprocessor cache coherence and link-level protocols) in addition to the usual academic examples (dining philosophers, alternating bit protocol, etc.) [10]. Mur$\varphi$ encompasses both a C++-based verifier and a high-level BDD-based verifier, called Ever, which supports integer and enumeration types, arrays, and records, a wide range of arithmetic, logical, and relational operators, and high-level imperative control structures (sequence, if-then-else) as well as non-determinism directly using standard BDDs [13]. (Although similar in spirit and motivation, Ever differs in this respect from other approaches to

* This research was supported by the Defense Advanced Research Projects Agency (contract number N00014-87-K-0828) and by a gift from Mitsubishi Electronics. The first author was supported by an ONR Graduate Fellowship. Most of this work was done using equipment generously donated by Sun Microsystems.

higher-level BDD-based verification, like MDDs [18, 15] or EVBDDs [16], which are extensions to the basic BDD data structure.)

To our initial surprise, the BDD-based verification method has been disappointing — a method which stores all of the reachable states explicitly in a hash-table substantially outperforms the BDD methods on our real examples. Moreover, existing techniques for efficient verification with BDDs hold no particular promise for alleviating these problems.

There are several sources of BDD-size blowup in our examples, each of which must be addressed if we are to have efficient verification of invariants. We have found a unified view of many sources of BDD-size blowup — that a single BDD must represent the conjunction of many small BDDs in such a way that the BDD for the conjunction is huge, regardless of variable-ordering — and propose a unified framework that eliminates BDD-size blowup in these cases by only partially evaluating the property being verified into a list of small BDDs that is *implicitly* conjoined, rather than explicitly building the large BDD for the conjunction, and maintaining these implicitly conjoined lists of small BDDs throughout the verification process.



## 2 Theoretical Basis

In modeling the system being verified, we generally assume an interleaving model of concurrency, both because such a model translates easily and efficiently into our underlying representation and also because that is the most intuitive model for the high-level distributed protocols we want to verify. The high-level Ever description is translated automatically into the underlying model of computation [13], which is a single non-deterministic finite-state machine with set of states $Q$, transition relation $\delta : Q \times Q \rightarrow \{0, 1\}$, and set of start states $S \subseteq Q$. The verification task is, given the set of states $I \subseteq Q$ that satisfies the invariant property being verified, to determine if there exists a path starting from a state in $S$ and leading to a state not in $I$, and, if there is such a path, to output that path as a counterexample to the property being verified.

The usual approach to such a verification task is to compute the set of states reachable from $S$ and to check that the set of reachable states is a subset of $I$ (e.g. [8, 5, 7, 19, 4]). This approach entails computing the set of reachable states as the fixed-point $\mu Z.\lambda u.S(u) \vee \exists v[Z(v) \wedge \delta(v, u)]$, which is the smallest set $Z$ such that $S \subseteq Z$ and any state that is a successor under $\delta$ of a state in $Z$ is also in $Z$ [6]. We will call this approach "forward traversal."

The expression $\exists v[Z(v) \wedge \delta(v, u)]$ is generally called the image of set $Z$ under transition relation $\delta$ [9, 19], which we will denote by Image$(\delta, Z)$. Also commonly used is the image on the domain of $\delta$ of a subset of the codomain given by $\exists v[Z(v) \wedge \delta(u, v)]$, which we denote by PreImage$(\delta, Z)$. We will also be using the expression $\forall v[\delta(u, v) \Rightarrow Z(v)]$, which we will denote by BackImage$(\delta, Z)$. The BackImage operator behaves in some sense as the inverse of the Image operator because BackImage$(\delta, Z)$ is the largest set such that Image$(\delta,$ BackImage$(\delta, Z)) \subseteq Z$. (The PreImage is sometimes called the inverse image [19]. PreImage, however, has this "inverse of Image" behavior only when $\delta$ is a total function (deterministic), in which case PreImage and BackImage are the same.)

An important property of these image operators is that if $\delta$ is described in Ever (using high-level data structures, imperative semantics, and non-determinism) and the set $Z$ is represented as a BDD (either built from an Ever description or computed by the