Electronic Notes in Theoretical Computer Science 56 (2001)
http://www.elsevier.nl/locate/entcs/volume56.html 190 pages

Formal Verification based on Boolean Expression Diagrams

Poul Frederick Williams

Abstract

This dissertation examines the use of a new data structure called Boolean Expression Diagrams (BEDs) in the area of formal verification. The recently developed data structure allows fast and efficient manipulation of Boolean formulae. Many problems in formal verification can be cast as problems on Boolean formulae. We chose a number of such problems and show how to solve them using BEDs.

Equivalence checking of combinational circuits is a formal verification problem which translates into tautology checking of Boolean formulae. Using BEDs we are able to preserve much of the structure of the circuits within the Boolean formulae. We show how to exploit the structural information in the verification process.

Sometimes combinational circuits are specified in a hierarchical or modular way. We present a method for verifying equivalence between two such circuits. The method builds on *cut propagation*. Assuming that the two circuits are given identical inputs, we propagate this knowledge through the circuits from the inputs to the outputs. The result is the knowledge of how the outputs of the two circuits correspond, e.g., are the outputs of the two circuits pairwise equivalent? The circuits and the movements of cuts can be described using Boolean formulae.

Symbolic model checking is a technique for verifying temporal specifications of finite state machines. It is well known how finite state machines and the evaluation of the temporal specifications can be expressed using Boolean formulae. We show how to do these manipulations using BEDs. We concentrate on examples which are hard for standard symbolic model checking methods.

Determining whether a formula is satisfiability is a problem which occurs in verification of combinational circuits and in symbolic model checking. Often satisfiability checking is associated with detecting errors. We examine how satisfiability checking can be done using the BED data structure.

Finally, we take a look at how it is possible to extend the BED data

structure. Among other operations, we introduce an operator for computing minimal *p-cuts* in fault trees. A fault tree is a Boolean formula expressing whether a system fails based on the condition ("failure" or "working") of each of the components. A minimal *p-cut* is a representation of the most likely reasons for system failure. This method can be used to calculate approximately the probability of system failure given the failure probabilities of each of the components.

As part of this research, we have developed a BED package. The appendix describes the package from a user's point of view.

Note Added in Print

This book is a slight revision of the author's Ph.D. thesis [Wil00].