Equational Binary Decision Diagrams

Jan Friso Groote\textsuperscript{1,2} and Jaco van de Pol\textsuperscript{1}

\textsuperscript{1} CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
JanFriso.Groote@cwi.nl, Jaco.van.de.Pol@cwi.nl
\textsuperscript{2} Department of Computing Science, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. We allow equations in binary decision diagrams (BDD). The resulting objects are called EQ-BDDs. A straightforward notion of reduced ordered EQ-BDDs (EQ-OBDD) is defined, and it is proved that each EQ-BDD is logically equivalent to an EQ-OBDD. Moreover, on EQ-OBDDs satisfiability and tautology checking can be done in constant time.

Several procedures to eliminate equality from BDDs have been reported in the literature. Typical for our approach is that we keep equalities, and as a consequence do not employ the finite domain property. Furthermore, our setting does not strictly require Ackermann’s elimination of function symbols. This makes our setting much more amenable to combinations with other techniques in the realm of automatic theorem proving, such as term rewriting.

We introduce an algorithm, which for any propositional formula with equations finds an EQ-OBDD that is equivalent to it. The algorithm has been implemented, and applied to benchmarks known from literature. The performance of a prototype implementation is comparable to existing proposals.

1 Introduction

Binary decision diagrams (BDDs) \cite{Bryant86} are widely used for checking satisfiability and tautology of boolean formulae. Applications include hardware verification and symbolic model checking. Every formula of propositional logic can be efficiently represented as a BDD. BDDs can be reduced and ordered, which in the worst case requires exponential time, but for many interesting applications it can be done in polynomial time. The reduced and ordered BDD (OBDD) is a unique representation for boolean formulae, so satisfiability, tautology and equivalence on OBDDs can be checked in constant time.

Much current research is done on extending the BDD techniques to formulae outside propositional logic. In principle, the boolean variables can be generalized to arbitrary relations. The goal now is to check satisfiability or validity of quantifier free formulae in a certain theory. The main example is the logic of equality and uninterpreted function symbols (EUF) \cite{Ganzinger95}. Another example is the logic of difference constraints on integers or reals \cite{Ganzinger95}.

© Springer-Verlag Berlin Heidelberg 2000
EUF formulae have been successfully applied to the verification of pipelined microprocessors \cite{8,7} and of compiler optimizations \cite{16}. In these applications, functions can be viewed as black boxes that are connected in different ways. Hence the concrete functions can be abstracted from, by replacing them by uninterpreted function symbols (i.e., universally quantified function variables). It is clear that if the abstracted formula is valid, then the original formula is. However, the converse is not true, e.g. \(x + y = y + x\) is valid, but its abstract version \(F(x, y) = F(y, x)\) is not.

Two methods for solving EUF formulae exist. The first method is based on two observations by Ackermann \cite{1}. First, the function variables can be eliminated, essentially by replacing any two subterms of the form \(F(x)\) and \(F(y)\) by new variables \(f_1\) and \(f_2\), and adding functionality constraints of the form \(x = y \rightarrow f_1 = f_2\). The second observation is the finite domain property, which states that the resulting formula is satisfiable if, and only if, it is satisfiable over a finite domain. Given an upper bound \(n\) on this domain, each domain variable can be encoded as a vector of \(\lceil \log(n) \rceil\) bits. In this way the original problem is reduced to propositional logic, and can be solved using existing BDD techniques.

The second method extends the BDD data structure, by allowing equations in the nodes of a BDD, instead of boolean variables only. By viewing all atoms as distinct variables, the BDD algorithms can still be used to construct a reduced ordered BDD. Contrary to the propositional case, a path in these OBDDs can be inconsistent, for instance because it violates transitivity constraints. As a consequence, all paths of the resulting OBDD have to be checked in order to conclude satisfiability.

Ultimately, we are interested in the symbolic verification of distributed systems, using high-level descriptions. This involves reasoning about data types (specified algebraically) and control (described by boolean conditions on data). Properties of the system are described using large boolean expressions. We want to use BDD-techniques in order to prove, or at least simplify, boolean expressions containing arbitrary relation and function symbols. In this setting, abstraction doesn’t work, as it doesn’t preserve logical equivalence. Without abstraction, Ackermann’s function elimination cannot be applied, and the finite domain property doesn’t hold.

We therefore turn to the second method, allowing equations in the BDD nodes. We will give a new definition of “ordered”, such that in ordered BDDs all paths will be consistent. The advantage is that on ordered BDDs with equations, the satisfiability check can be done in constant time. The contribution of this paper is an intermediate step towards the situation where arbitrary relations and function symbols in BDDs are allowed. We restrict to the case of equations, without function symbols.

Technical Contribution. In Section \(2\) we introduce EQ-BDDs, which are BDDs whose internal nodes may contain equations between variables. We extend the notion of orderedness so that it covers the equality laws for reflexivity, symmetry, transitivity and substitution. The main idea is that in a (reduced) ordered EQ-