Theoretical Computer Science 595 (2015) 11–33

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the error resilience of ordered binary decision diagrams ☆

Anna Bernasconi [a], Valentina Ciriani [b,*], Lorenzo Lago [b]

[a] Dipartimento di Informatica, Università di Pisa, Italy
[b] Dipartimento di Informatica, Università degli Studi di Milano, Italy

ARTICLE INFO

Article history:
Received 17 December 2013
Received in revised form 17 April 2015
Accepted 28 May 2015
Available online 9 June 2015
Communicated by G.F. Italiano

Keywords:
Binary decision diagrams
Error resilient data structures

ABSTRACT

An Ordered Binary Decision Diagram (OBDD) is a data structure that is used in an increasing number of fields of Computer Science (e.g., logic synthesis, program verification, data mining, bioinformatics, and data protection) for representing and manipulating discrete structures and Boolean functions. The purpose of this paper is to study the error resilience of OBDDs and to design a resilient version of this data structure, i.e., a self-repairing OBDD. In particular, we describe some strategies that make reduced ordered OBDDs resilient to errors in the indices, that are associated to the input variables, or in the pointers (i.e., OBDD edges) of the nodes. These strategies exploit the inherent redundancy of the data structure, as well as the redundancy introduced by its efficient implementations. The solutions we propose allow the exact restoring of the original OBDD and are suitable to be applied to classical software packages for the manipulation of OBDDs currently in use. Another result of the paper is the definition of a new canonical OBDD model, called *Index-Resilient Reduced OBDD*, which guarantees that a node with a faulty index has a reconstruction cost $O(r)$, where r is the number of nodes with corrupted index. Experimental results on a classical benchmark suite validate the proposed approaches.

1. Introduction

Ordered Binary Decision Diagrams (OBDDs) are the state-of-the-art data structure for Boolean function representation and manipulation. Indeed, they are widely used in logic synthesis, CAD of integrated circuits and in many safety critical applications, like verification (see [7] and [9], and [3,4,6] for more recent applications of OBDDs to logic synthesis). A binary decision diagram (BDD) over a set of Boolean variables $X = \{x_0, x_1, \ldots, x_{n-1}\}$ is a rooted, connected direct acyclic graph, where each non-terminal node is labeled with a variable of X, and each terminal node is labeled with a value in $\{0, 1\}$. Each non-terminal node has exactly two outgoing edges, 0-edge and 1-edge, pointing to two nodes called 0-child and 1-child of the node. A BDD is *ordered* if there exists a total order $<$ over the set X of variables such that if a non-terminal node is labeled by x_i, and its 0-child and 1-child have labels x_{i_0} and x_{i_1}, respectively, then $x_i < x_{i_0}$ and $x_i < x_{i_1}$.

BDDs were first introduced by Lee [19] and Akers [1], and developed by Bryant who proposed a canonical representation in [8]. Besides digital-system design, nowadays BDDs are applied for representing and manipulating discrete structures in other research fields, as for instance data mining [21–23], bioinformatics [24,25,29], and data protection [10]. The growing

☆ This work was supported in part by the Italian Ministry of Education, University, and Research (MIUR) under PRIN 2012C4E3KT national research project "AMANDA" Algorithmics for MAssive and Networked DAta.

* Corresponding author.
E-mail addresses: anna.bernasconi@unipi.it (A. Bernasconi), valentina.ciriani@unimi.it (V. Ciriani), lorenzo.lago@studenti.unimi.it (L. Lago).

http://dx.doi.org/10.1016/j.tcs.2015.05.050
0304-3975/

interest in BDDs is also evidenced by the fact that in 2009 Knuth dedicated the first fascicle in the volume 4 of "The Art of Computer Programming" to this data structure [18].

However, despite their popularity, error resilient versions of BDDs have not yet been proposed. We are aware only of a paper where security aspects of implementation techniques of OBDDs are discussed, and methods to verify the integrity of OBDDs are presented [11]. In particular, a recursive checksum technique for on-line and off-line checks is proposed and experimentally evaluated: the on-line check verifies the correctness of the node during each access, so that errors can be detected very early; while the off-line check (usually performed by a depth-first-search algorithm starting from the root of the OBDD) is used to verify the integrity of the whole data structure. However, [11] only deals with the problem of error detection, and does not consider error correction, which is instead the main goal of our paper.

Nowadays, the resilience of algorithms and data structures to memory fault is a very important issue [14–16]: fast, large, and cheap memories in today's computer platforms are characterized by non-negligible error rates, which cannot be underestimated as the memory size becomes larger [17]. Computing in the presence of memory errors is therefore a fundamental task in many applications running on large, fast and cheap memories, as the correctness of the underlying algorithms may be jeopardized by even very few memory faults.

The scientific community has studied the problem in two different frameworks: *(i)* fault tolerant hardware design and *(ii)* development of error resilient algorithms and data structures. While fault tolerant hardware has been widely studied even in the past, the design of algorithms and data structures resilient to memory faults, i.e., algorithms and data structures that are able to perform the tasks they were designed for, even in the presence of unreliable or corrupted information, has become much more attractive only recently (for a survey on the subject refer to [16]).

The purpose of this paper is precisely to discuss the error resilience of OBDDs and to design a resilient version of this data structure.

In particular, we describe some strategies that make reduced OBDDs resilient to errors in the indices, that are associated to the input variables, or in the pointers (i.e., OBDD edges) to the nodes. These strategies exploit the inherent redundancy of this data structure, as well as the redundancy introduced by its efficient implementations. The solutions we propose *(i)* allow the exact restoring of the original OBDD and of the associated function f, and *(ii)* are suitable to be applied to classical software packages for the manipulation of OBDDs currently in use, as for instance the CUDD library. Indeed, our first goal is to be able to efficiently reconstruct via software the corrupted OBDD without changing the data structure.

However, to reach this goal we first assume that the unique table, i.e., a hash table used by most software implementation of OBDDs to facilitate their reduction (see Section 2 for more details), is fault free. More precisely, we assume that the unique table is either implemented using error resilient linked lists [2] or it is stored in a safe memory area not affected by errors. The last one could be seen as a strong requirement, but fortunately we are able to remove this assumption completely still guaranteeing a very efficient reconstruction of all corrupted indices in the OBDD. Indeed, the main contribution of the paper is the definition of a new canonical OBDD model, called *Index-Resilient Reduced OBDD*, which guarantees, by construction, that a node with a faulty index has a reconstruction cost $O(r)$, where r is the number of nodes with corrupted index. As the new model does not exploit the unique table to restore all corrupted indices, we do not need a fault-free unique table anymore. Instead, we will only require that the two terminal nodes (the leaves of the OBDD) are always uncorrupted, and therefore that they are memorized in a safe memory or duplicated. We also show how index-resilient reduced OBDDs can be constructed starting from binary decision trees or by applying Boolean operations to index-resilient reduced OBDDs. Both construction methods can be implemented with error resilient algorithms, i.e., algorithms capable of dealing with errors (in the data structures) occurring during their execution.

Finally, we describe some methods for dealing with errors on edges. We can consider two possible strategies: we use safe unique tables, implemented with perfect hash functions, or we can use hash tables with error resilient linked lists [2]. While the first approach guarantees a full error correction at the expense of the strong assumption on the fault freeness of the unique tables, the second strategy does not require safe unique tables, but can fail in some error corrections due to collisions. The experimental results indicate some setting for the hash tables that can limit the percentage of failed recoveries.

The paper is an extended version of the conference paper [5] and is organized as follows. Definitions of the error models and preliminaries on OBDDs and their implementations are described in Section 2. In Section 3 we propose an efficient index reconstruction algorithm, and in Section 4 we introduce and study index-resilient OBDDs. Section 5 discusses how index-resilient OBDDs can be dynamically computed through a sequence of binary Boolean operators (as AND, OR, EXOR) applied to other index-resilient OBDDs using the standard algorithm Apply (reviewed in Appendix A). Section 6 describes strategies for broken edge reconstruction. Experimental results for validating the proposed strategies are reported in Section 7. Section 8 concludes the paper.

2. Preliminaries

2.1. Reduced ordered binary decision diagrams

A *Binary Decision Diagram* (BDD) over a set of Boolean variables $X = \{x_0, x_1, \ldots, x_{n-1}\}$ is a rooted, connected direct acyclic graph, where each non-terminal (internal) node N is labeled by a Boolean variable x_i and has exactly two outgoing edges, the 0-edge and the 1-edge, pointing to two nodes called the 0-child and the 1-child of node N, respectively. N is called