Zero-Suppressed Binary Decision Diagrams Resilient to Index Faults

Anna Bernasconi, Valentina Ciriani

Zero-Suppressed Binary Decision Diagrams Resilient to Index Faults*

Anna Bernasconi[1] and Valentina Ciriani[2]

[1] Dipartimento di Informatica, Università di Pisa, Italy, annab@di.unimi.it
[2] Dipartimento di Informatica, Università degli Studi di Milano, Italy, valentina.ciriani@unimi.it

Abstract. This paper discusses the error resilience of Zero-Suppressed Binary Decision Diagrams (ZDDs), which are a particular family of Ordered Binary Decision Diagrams used for representing and manipulating combination sets. More precisely, we design a new ZDD canonical form, called index-resilient reduced ZDD, such that a faulty index can be reconstructed in time $O(k)$, where k is the number of nodes with a corrupted index.

1 Introduction

Algorithms and data structures resilient to memory faults [11–13] are able to perform the tasks they were designed for, even in the presence of unreliable or corrupted information. The design of resilient algorithms and data structures is a fundamental issue, as fast, large, and cheap memories in computer platforms are characterized by non-negligible error rates [14].

Several error models have been proposed for designing resilient data structures [22]. A fault model in which any error is detectable via an error message when the program tries to reach the faulty object is proposed in [2]. The authors assume that an error denies access to an entire node of the structure. A model with higher granularity, called *faulty-RAM*, is presented in [9, 10, 13]. In faulty-RAM an adversary can corrupt any memory word and it is impossible to determine a priori if a memory area is corrupted or not. Such a scenario is realistic since an error can be induced by an external source, perhaps temporary, which can change any memory location that can not be discovered a priori. Moreover, faulty-RAM model has an extreme granularity: any memory location (from a single bit, the single data, or an entire structure) can be affected by a fault. Another interesting error model is the *single-component model* described in [22], which focuses on single attributes of an item at a time and assumes that each error affects one component of one node of the storage structure, e.g., a pointer, a count, an identifier field.

* 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

The purpose of this paper is to discuss the error resilience of a data structure called *Zero-Suppressed Binary Decision Diagrams* (ZDDs) [17], that are a particular type of Ordered Binary Decision Diagrams (OBDDs). OBDDs are a fundamental family of data structures for Boolean function representation and manipulation [5]. They have been originally studied for circuit design and formal verification. Recently, the area of application of OBDDs has widened including representation and manipulation of combination sets in different research fields as data mining [17–19], bioinformatics [20, 21, 23], data protection [6]. The growing interest in these data structures is evidenced by the fact that in 2009 Knuth dedicated the first fascicle in the volume 4 of "The Art of Computer Programming" to OBDDs [15].

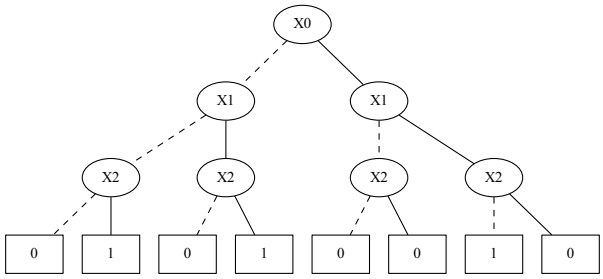
OBDDs are DAG representations of Boolean functions, where each internal node N is labeled by a Boolean variable x_i and has exactly two outgoing edges: 0-edge and 1-edge. Terminal nodes (leaves) are labeled 0 or 1. OBDDs can be constructed by applying some reduction rules to a Binary Decision Tree, and depending on the set of reduction rules, different representations can be derived. For example, Figures 1(b), 1(c), and 1(d) are different decision diagrams, derived by the decision tree in Figure 1(a), representing the same Boolean function. In particular, Reduced OBDDs (ROBDDs) [5] are typically used for the representation of general Boolean functions, while *Zero-Suppressed* BDDs (ZDDs) are used for representing family of subsets of combination sets [15, 17]. Indeed, ZDDs can be used to describe and manipulate solutions to combinatorial problems as, in this framework, they are much more compact than ROBDDs. For instance, the family of subsets $\{\{x_1, x_2\}, \{x_3, x_4\}, \{x_1\}\}$ of the set $\{x_1, x_2, \ldots, x_{10}\}$ needs a ROBDD representation with 10 variables, while the ZDD representation uses only the four variables included in the subsets.

Security aspects of implementation techniques of ROBDDs have been discussed in [7], and an error resilient version has been proposed in [3, 4]. In this paper we study error resilience of ZDDs. We exploit the single-component error model and we assume that errors are reported when the program tries to use the fault component of a node. We consider, as component of a node N in a ZDD, the index i of the variable x_i associated to the node. In particular, we design a ZDD, called *index-resilient reduced ZDD*, such that a faulty index can be reconstruct in time $O(k)$, where k is the number of nodes with a corrupted index. Moreover, the proposed index-resilient ZDD is a canonical form.

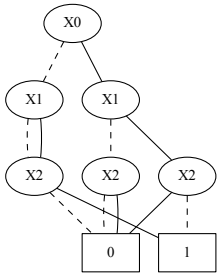
The paper is organized as follows. Preliminaries on OBDDs and ZDDs are described in Section 2. In Section 3 we discuss the error resilience of the standard ZDD structure, and in Section 4 we introduce and study index-resilient ZDDs. Section 5 concludes the paper.

2 OBDDs and Zero-Suppressed BDDs

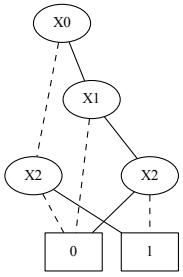
A *Binary Decision Tree* (BDT) on a set of Boolean variables $\{x_0, x_1, \ldots x_{n-1}\}$ is a rooted binary tree, where each non-terminal (internal) node N is labeled by a Boolean variable x_i and has exactly two outgoing edges: 0-edge and 1-edge.



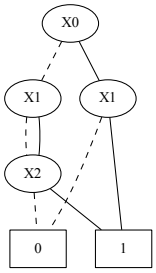
(a) BDT



(b) QR-BDD (m-rule)



(c) ROBDD (m-rule and r-rule)



(d) ZDD (m-rule and z-rule)

Fig. 1. Example of transformations of a BDT using the reduction rules.

Terminal nodes (leaves) are labeled 0 or 1 (e.g., see Figure 1(a) where dashed, rep., solid, lines represent 0-edges, resp., 1-edges). Without loss of generality, we can assume that each node containing the variable x_i (with $0 \leq i \leq n-1$) lyes on the i-th level of the tree. Thus, the variable x_0 is the root of the BDT and the leaves are on level n (see for example the BDT in Figure 1(a)).

BDTs are typically used to represent completely specified Boolean functions (i.e., any function $f : \{0,1\}^n \rightarrow \{0,1\}$). The leaves represent the constants 0 and 1 and the root represents the entire Boolean function f. The value of f on the input $x_0, \ldots, x_{n-1}$ is found by following the path indicated in the BDT by the values of $x_0, \ldots, x_{n-1}$ on the edges: the value of $f(x_0, \ldots, x_{n-1})$ is the label of the reached leaf. For example, the BDT in Figure 1(a) represents the Boolean function $f : \{0,1\}^3 \rightarrow \{0,1\}$ such that $f(0,0,0) = 0, f(0,0,1) = 1, \ldots, f(1,1,1) = 0$.

In order to give a more compact description of Boolean functions, a BDT can be compressed in an acyclic graph (called BDD) that represents the same function. In particular, a *Binary Decision Diagram* (BDD) on 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, 0-edge and 1-edge, pointing to two nodes called 0-child and 1-child of node N, respectively. Terminal nodes (leaves) are labeled 0 or 1. A 0-*parent* (resp., 1-*parent*) of a node N is a node M such that N is a 0-child (resp, 1-child) of M. For instance, the decision diagrams in Figures 1(b), 1(c), and 1(d) are examples of BDDs.

A BDD is *ordered* (OBDD) if there exists a total order $<$ over the set X of variables such that if an internal 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 $x_i < x_{i_1}$. Hereafter we will consider ordered BDDs only.

In order to obtain an OBDD starting from a BDT we can apply several *reduction rules*:

- **m-rule:** (or merge rule) if M and N are two distinct nodes that are roots of isomorphic subgraphs, then N is deleted, and all the incoming edges of N are redirected to M (N and M are called *mergeable*);
- **r-rule:** (or redundant rule) a node N that has both edges pointing to the same node M is deleted and all its incoming edges are redirected to M (N is called *redundant node* or *r-node*);
- **z-rule:** (or zero-suppress rule) a node N that has the 1-edge pointing to the constant leaf 0 is deleted and all its incoming edges are redirected to the subgraph pointed by the 0-edge (N is called *z-node*).

A *zr-node* is a redundant z-node, i.e., is a node with both edges pointing to the constant leaf 0.

There are different *reduced BDD forms* that derive from the use of one or two reduction rules:

- **QR-BDD:** (*Quasi-Reduced BDD*) [16] is the OBDD derived from a BDT repeatedly applying the m-rule until it is no longer applicable (see Figure 1(b));
- **ROBDD:** (*Reduced Ordered BDD*) [1, 5, 8, 15] is the OBDD derived from a BDT repeatedly applying the m-rule and r-rule until they are no longer applicable (see Figure 1(c));
- **ZDD:** (*Zero-suppressed BDD*) [15, 17] is the OBDD derived from a BDT repeatedly applying the m-rule and z-rule until they are no longer applicable (see Figure 1(d)).

QR-BDDs, ROBDDs and ZDDs are *canonical forms*. In particular, given a function f and a variable ordering $<$, there is exactly one QR-BDD, one OBDD, and one ZDD with variable ordering $<$ that represent f. Thus, once we have fixed the variable ordering, we can compute the QR-BDD, the ROBDD and the ZDD starting from a BDT repeatedly applying the corresponding reduction rules in any order. Moreover, it is possible to first build a QR-BDD (applying the m-rule) and then transform it in a ROBDD (resp., ZDD) using the r-rule (resp., z-rule) on it. In fact, starting from a QR-BDD, the r-rule and the z-rule cannot create new mergeable nodes (as shown in [4] for the r-rule, and in

Section 4 for the z-rule). The interpretation of a QR-BDD as a Boolean function is equivalent to the interpretation of a BDT since the m-rule simply merges isomorphic subgraphs resulting in an OBDD that has all the paths from the root to the leaves containing *all* the variables in X. For ROBDDs and ZDDs we have to give a correct interpretation of possibly missing nodes (in a path), which have been deleted using the r-rule or the z-rule. In particular, a missing variable in a path of a ROBDD means that the variable can have any value (0 or 1). For example, in Figure 1(c), the path "x_0 0-edge x_2 1-edge 1", where x_1 is missing, represents two possible input values (i.e., 001, 011) on which the function takes the value 1. On the other hand, the interpretation of a missing variable x_i in a path of a ZDD means that if $x_i = 1$ the function outputs 0, otherwise (i.e., if $x_i = 0$) the function outputs the value obtained following the path. For example, in Figure 1(d), the path "x_0 1-edge x_1 1-edge 1", where x_2 is missing, means that $f(1, 1, 1) = 0$ and $f(1, 1, 0) = 1$.

3 Index Reconstruction Cost

In this section we discuss error resilient indexes in ZDDs, we analyze the cost of the reconstruction of a corrupted index, and study the impact of the ZDD reduction rules on this cost. This study gives us the knowledge to describe in Section 4 a new index resilient version of ZDDs.

Monitoring the work on error resilient OBDDs [3, 4], we give some definitions useful to describe error resilient ZDDs. Without loss of generality, let us assume that the chosen variable ordering is $x_0 < x_1 < \ldots < x_{n-1}$, so that the index of a variable in a node is the *level* of the node in the corresponding ZDD. In order to facilitate the index reconstruction of a faulty node N we define the range of indexes that contains the original index of the node. Let N be an internal node in a ZDD Z, the *node range* $I_N = [i_P + 1, i_C - 1]$ is the range containing all the possible levels for N in Z, where i_P is the maximum index of N's parents in Z, and i_C is the minimum index of its children, where the leaves have "index" n, and if N is the root, i.e., N has no parent, $i_P = -1$.

Obviously, by definition of ZDD, the index of node N belongs to its range I_N. Thus, in presence of an error in the index i of N we have a lower and an upper bound for the reconstruction of i given by $i_P + 1$ and $i_C - 1$, respectively. In particular, if $i_P + 1 = i_C - 1$, then i is $i_C - 1$.

Let us now examine which characteristics make a ZDD more suitable to the reconstruction of a corrupted index. To this aim, we introduce a metric to measure the cost of the reconstruction of a corrupted index of a ZDD node in the worst case. The *index reconstruction cost* $C(N)$ of the faulty index i in the node N is given by the number of indexes that are candidate to be the correct one in N.

If we consider the case of one fault only in node N, we have that $C(N)$ is at most $|I_N|$. In particular, $C(N) = |I_N|$ whenever there is no additional knowledge on the structure of the ZDD. In the rest of this section, we therefore

assume that $C(N) = |I_N|$. Instead, in Section 4 we will study ZDDs with a particular structure implying that $C(N) \leq |I_N|$.

In the best case, for each node N of a ZDD we have that $C(N)$ is 1, meaning that any index can be reconstructed in constant time (considering one single error). This condition is obviously satisfied by BDTs. In fact, in a BDT, all paths from the root to the terminal nodes contain exactly n nodes, where n is the number of input variables. Thus, for each node N, $C(N) = |I_N| = 1$. It is interesting to notice that the optimal cost $C(N) = 1$ can also be reached by reduced ZDDs.

Recalling that a reduced ZDD can be constructed from a BDT by iteratively applying the reduction rules (m-rule and z-rule) and noticing that a BDT has optimal cost, we can study how the node range can increase using the two reduction rules.

We first consider the merge rule, i.e., the rule that is also used for the reduction of OBDDs. In the OBDD context, Theorem 1 in [3] shows that this rule does not increase the index reconstruction cost of the nodes. In fact, the merge of isomorphic subgraphs does not change the node range of the involved nodes. In other words, each node N in a QR-BDDs is such that $C(N) = |I_N| = 1$.

On the other hand, the second reduction rule (z-rule), that distinguishes ZDDs from OBDDs, can increase the index reconstruction costs. For example, consider the ZDD in Figure 1(d), that can be obtained applying the z-rule to the QR-BDD in Figure 1(b). While each node of the QR-BDD has cost 1, node x_1 on the right of the ZDD has cost 2, since its range is increased by the z-rule. We finally note that not always the z-rule increases the node cost.

4 Index-Resilient Reduced ZDDs

The analysis of the previous section shows that, while the merge rule never increases the overall index reconstruction cost, the application of the z-rule could increase it. In this section, we describe a new reduced ZDD model where we maintain some z-nodes in the diagram, in order to guarantee a constant index reconstruction cost for each node. In particular we will define a ZDD, called *index-resilient reduced ZDD*, satisfying the following properties:

1. the index reconstruction cost of each node N is $C(N) = 1$;
2. in presence of k nodes with a corrupted index in an index-resilient ZDD, the cost needed to reconstruct a faulty index is $O(k)$;
3. starting from a QR-BDD, the construction of the corresponding index-resilient reduced ZDD is linear in time;
4. the index-resilient reduced ZDD is canonical.

Due to space limitations, formal proofs are omitted and will be discussed in the extended version of this paper.

We note that, since the z-rule can increase the index reconstruction cost, we could decide not to apply this rule during the reduction of a ZDD. In this way, we only use the m-rule and obtain a QR-BDD that has a cost $C(N) = 1$ for each

node N. Recall that an important property of QR-BDD is that each node at level i has all parents at level $i-1$ and all children on level $i+1$. QR-BDDs are still a compact representation and could represent a convenient and canonical trade-off between memory saving, reduction time and error reconstruction time.

However, the use of the z-rule does not always increase the index reconstruction cost. In other words, it is still possible to delete some z-nodes in a QR-BDD guaranteeing that, in the final OBDD, the index reconstruction cost of each node N is still $C(N) = 1$. Most importantly, as we will show in this section, it is possible to apply the z-rule to some z-nodes, and derive a *canonical* OBDD, *more compact* than a quasi-reduced one, and with a cost $C(N) = 1$ for each node N. We will call these OBDDs *index-resilient ZDDs*.

Definition 1 (Index-Resilient ZDD). *An* Index-Resilient ZDD *is an OBDD obtained from a QR-BDD applying several times, possibly never, the z-rule guaranteeing that each internal node N on level i has at least one child on level $i+1$, for any level of the OBDD.*

In particular, a QR-BDD is an index-resilient ZDD where each node on level i has *all* parents on level $i-1$ and *all* children on level $i+1$.

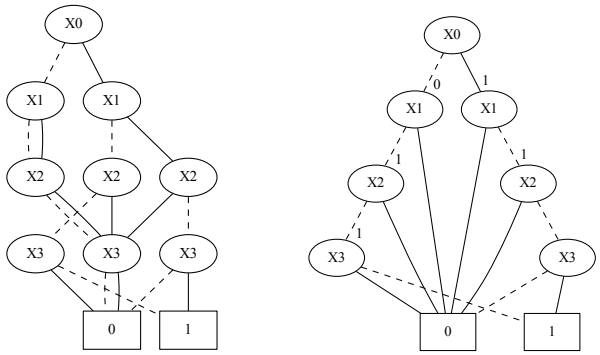
Observe that the index reconstruction cost for any node N in an index-resilient ZDD is $C(N) = 1$, since the variable index of a node N is directly given by $i = \min\{i_0, i_1\} - 1$, where i_0 and i_1 are the indexes of the 0- and 1-child of N. Note that, for any internal node N in a ZDD, the number of children of N is 2, but the number of parents of N can be $O(m)$, where m is the total number of nodes in the ZDD, and, in the worst case, $m \in \Theta(2^n/n)$ [16]. Note also that for the reconstruction of the index of N we do not need to know the indexes of its parents (whose number can be exponential in the number of variables), but only the indexes of its children. In fact, as shown below, we can define a structure where any node containing a variable x_i must have at least one child containing the variable x_{i+1}.

To compute a compact index-resilient ZDD, we start from a QR-BDD deleting some z-nodes while preserving the index-resilient property. For this purpose we first observe that in a QR-BDD there are at most one zr-node and one non-redundant z-node. These nodes are on level $n-1$. See, for example, the QR-BDD depicted in Figure 1(b).

We first consider the zr-node N_{zr}, if existing, in the QR-BDD. We can note that the removal of N_{zr} can generate new z-nodes, i.e., the 1-parents of N_{zr} (see Proposition 1). In particular, if N_{zr} has an r-node parent M, the removal of N_{zr} transforms M in a zr-node but the ZDD is not index-resilient since M is at level $n-2$ and has both children (the 0 constant) at level n. If we remove the zr-node M, we can generate again new z-nodes and one possible zr-node that has reconstruction cost greater than 1. We can, therefore, consider the entire chain of r-nodes that ends with a zr-node defined as follows:

Definition 2 (zr-chain). *A* zr-chain *in an index-resilient ZDD is a chain $C = N_1, N_2, \ldots, N_k$ (with $k \geq 1$) of nodes such that:*

1. *N_1 has no redundant parents,*



(a) A zr-chain (b) Two z-chains

Fig. 2. Examples of zr-chain and z-chains. The value $P_C(N)$ is depicted over each node N in the z-chains. The z-chain on the left is removable.

2. *N_i, with $i \in [1, \ldots, k-1]$, is an r-node and its unique child is N_{i+1},*
3. *N_k is a zr-node.*

The node N_1 is called head *of the chain, and the leaf* 0 *is the* child *of the chain.*

When $k = 1$ the chain corresponds to the zr-node N_1. As already observed, the deletion of a zr-chain generates new z-nodes in the obtained index-resilient ZDD:

Proposition 1. *Let C be a zr-chain in an index-resilient ZDD Z. If C is removed from Z, then any node in Z that is a 1-parent of a node in C becomes a z-node.*

For example, consider the QR-BDD in Figure 2(a) that contains a zr-chain of three nodes. The removal of the zr-chain will produce two new z-nodes (the nodes with indexes x_2 that are not part of the chain).

If we remove the entire zr-chain, the resulting OBDD is still an index-resilient ZDD, as proved in the following proposition. Moreover, in a QR-BDD the zr-chain is unique and, once deleted, the resulting index-resilient ZDD does not contain a new zr-chain.

Proposition 2. *Let B be a QR-BDD containing a zr-chain C. We have that:*

1. *C is the unique zr-chain in B,*
2. *the OBDD B' derived by deleting C from B is an index-resilient ZDD,*
3. *B' does not contain any zr-chain,*
4. *B' does not contain any mergeable node.*

We can observe that, after the deletion of a zr-chain in a QR-BDD, each node N at level i in the resulting index-resilient ZDD has both children at level

$i+1$, or one child at level n (the 0 leaf) and a child at level $i+1$. Moreover, at level $n-1$ there exists at most one single z-node (i.e., the node that have the terminal 0 as 1-child and the terminal 1 as 0-child). The index-resilient ZDD obtained after the deletion of the zr-chain, can still contain z-nodes that can be removable. In order to efficiently test whether we can delete a z-node N, we first need the following parameter that counts the number of parents of N whose index reconstruction cost is affected by the deletion of N.

Definition 3 (P_C). *Let N be a z-node in an index-resilient ZDD resulting by the deletion of the zr-chain from a QR-BDD. $P_C(N)$ is the number of parents P of N satisfying* at least one *of the following properties:*

1. *Both children of P are z-nodes (possibly, the same z-node if P is redundant) and N is the* 1*-child of P;*
2. *P has another child $N' \neq N$ on a level strictly greater than $i+1$, where i is the level of P (i.e., N' is the terminal node 0).*

Note that if N is the root, then $P_C(N) = 0$. We can observe that Definition 3 derives from the fact that the cost $C(P) = 1$, of a node P at level i, is not increased by the deletion of its z-node child N if P has the other child $N' \neq N$, on level $i+1$ and N' cannot be removed. The child N' is not removed in two possible cases: 1) N' is not a z-node; 2) N' is a z-node (like N) but is the 1-child of P. This second criterion is an arbitrary choice due to the necessity of deleting one of the two z-nodes that are children of P while maintaining the index reconstruction cost and the canonicity of the representation. More precisely, when a node P has two children that are z-nodes, one of them can be removed without increasing the cost of P. In this paper we always remove the 0-child of P in order to guarantee that the resulting index-resilient ZDD is canonical (see Theorem 2). The choice of removing the 1-children is similar.

For example, see the index-resilient ZDD in Figure 2(b). Each z-node N in the figure has a value that corresponds to $P_C(N)$. We note that $P_C(N)$ can be efficiently computed with a simple visit of a index-resilient ZDD obtained after the deletion of the zr-chain from the QR-BDD.

When the QR-OBDD is constructed, the zr-chain (if existing) deleted, and P_C is computed, we can define chains of z-nodes (*z-chains*) and we can characterize the z-chains that can be removed, maintaining equal to 1 the index reconstruction cost of each remaining node. We therefore introduce the concept of *removable z-chain*.

Definition 4 (Removable z-chain). *A* removable z- chain *in an index-resilient ZDD, which does not contain zr-chains, is a chain $C = N_1, N_2, \ldots, N_k$ (with $k \geq 1$) of z-nodes such that:*

1. *N_i, with $i \in [2, \ldots, k]$, is the 0-child of N_{i-1},*
2. *$P_C(N_1) = 0$,*
3. *$\forall i \in [2, \ldots, k]$, $P_C(N_i) = 1$,*
4. *if M is a z-node then $P_C(M) > 1$, where M is the the 0-child of N_k.*

The node N_1 is called head *of the chain, and M is called* child *of the chain.*

The second requirement states that the head of the chain N_1 can be removed without affecting the reconstruction cost of its parents, as detailed in Proposition 4. Note that this requirement implies that all parents of N_1 are not z-nodes or r-nodes. The third requirement states that any other node N_i $(1 < i \leq k)$ of the chain affects only the reconstruction cost of its parent in the chain. The last requirement guarantees that the removable z-chain is maximal. When the chain is composed by a single z-node node N, we have that N is removable when $P_C(N) = 0$. Consider, for example, see the index-resilient ZDD in Figure 2(b). While, the z-chain on the left is removable, the z-chain on the left is not removable since the first node N_1 has $P_C(N_1) = 1$.

In an index-resilient ZDD there are no "crossing" z-chains, i.e., a node cannot be part of two different z-chains. This is a direct consequence of the following property.

Proposition 3. *In an index-resilient ZDDs there are no nodes with two or more z-nodes as parents.*

The following proposition shows that the deletion of all removable z-chains in an index-resilient ZDD Z does not change the overall index reconstruction cost, i.e., after the removal of the chains, each internal node on level i still has at least a child on level $i + 1$, for any level i in Z.

Proposition 4. *Let $C = N_1, N_2, \ldots, N_k$, $k \geq 1$, be a removable z-chain in an index-resilient ZDD, which does not contain a zr-chain. The OBDD resulting from the deletion of C is still an index-resilient ZDD.*

Observe that once removable z-chains have been deleted, we are left with an index-resilient ZDD that can still contain some z-nodes: those that do not form a removable chains.

We can now propose a new OBDD reduction algorithm that, starting from a QR-BDD, deletes first the zr-chain and then all the removable z-chains. The following Theorem 1 shows that the deletion of removable z-chains does not construct new removable z-chains. Therefore, after the removal of the zr-chain, we can detect (and than delete) all the removable chains at the same time.

The reduction algorithm is based on a constant number of visits starting from a quasi-reduced OBDD. The first visit is a breadth first search used to detect and remove the zr-chain, if exists. Another visit is used to compute the parameter P_C for each z-node; then with a breadth first visit, all removable z-chains are identified and their nodes are removed with a final visit of the OBDD, executed by a simple recursive depth first visit that deletes from the OBDD all nodes identified as removable.

The correctness of the new reduction algorithm is proved in the following theorem.

Theorem 1. *Let B be a quasi-reduced OBDD. The reduction algorithm computes an index-resilient ZDD Z equivalent to B that contains neither removable z-chains nor a zr-chain.*

The cost of the algorithm is linear in the size of the quasi-reduced OBDD in input, as it consists in a constant number of visits of the data structure.

We now formally introduce the concept of *Index-Resilient Reduced ZDD*.

Definition 5 (Index-Resilient Reduced ZDD). *An index-resilient ZDD is* reduced *if it contains neither a zr-chain nor removable z-chains.*

Theorem 2. *Let Z be an index-resilient reduced ZDD obtained with the reduction algorithm. Then*

1. *for each node N in Z, $C(N) = 1$;*
2. *Z does not contain mergeable nodes;*
3. *Z is canonical, i.e., given a function f and a variable ordering $<$, Z is the only index-resilient reduced ZDD with variable ordering $<$ that represents f.*

The index reconstruction cost remains limited even in presence of more than one error on the indexes, as stated and proved in the following theorem, that shows a result similar to the one obtained for OBDDs in [4].

Theorem 3. *The reconstruction cost of a node N on level i in an index-resilient reduced ZDD Z affected by k errors on the indexes is $O(\min(k, 2^{n-i}))$.*

Finally observe that, even if in our analysis we have implicitly assumed that a ZDD is constructed correctly, and that memory faults occur when the data structure is in use, this assumption can be completely removed for index-resilient reduced ZDDs. Indeed, their construction starts from a binary decision tree that is transformed into a QR-BDD, and in both models each node has all children on the level immediately below. Moreover, during the execution of the reduction algorithm, we always guarantee that each node has at least one child on the level below, thus a faulty index can be immediately detected and restored.

5 Conclusion

This paper has proposed a new ZDD canonical form that is resilient to errors in indexes. This form can be derived in linear time starting from a quasi-reduced OBDD. Future work on this subject includes the analysis of error in edges. Indeed, this problem is part of the more general problem of designing an error resilient DAG structure. Furthermore, it could be interesting to design error resilient algorithms for standard operations on ZDDs, like union, intersection, and set difference.

References

1. Akers, S.: Binary Decision Diagrams. IEEE Transactions on Computers 27(6) (1978)
2. Aumann, Y., Bender, M.: Fault Tolerant Data Structures. In: 37th Annual Symposium on Foundations of Computer Science (FOCS) (1996)

3. Bernasconi, A., Ciriani, V., Lago, L.: Error Resilient OBDDs. In: IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS). pp. 246–249 (2013)
4. Bernasconi, A., Ciriani, V., Lago, L.: On the Error Resilience of Ordered Binary Decision Diagrams. Technical Report arXiv:1404.3919 (2014)
5. Bryant, R.: Graph Based Algorithm for Boolean Function Manipulation. IEEE Transactions on Computers (1986)
6. Ciriani, V., di Vimercati, S.D.C., Foresti, S., Livraga, G., Samarati, P.: An obdd approach to enforce confidentiality and visibility constraints in data publishing. Journal of Computer Security 20(5), 463–508 (2012)
7. Drechsler, R.: Verifying integrity of decision diagrams. In: Computer Safety, Reliability and Security (1998)
8. Ebendt, R., Fey, G., Drechsler, R.: Advanced BDD Optimization. Springer Verlag (2005)
9. Finocchi, I., Grandoni, F., Italiano, G.: Designing reliable algorithms in unreliable memories. Computer Science Review 1(2), 77–87 (2007)
10. Finocchi, I., Italiano, G.: Sorting and Searching in Faulty Memories. Algorithmica (2008)
11. Finocchi, I., Grandoni, F., Italiano, G.F.: Optimal resilient sorting and searching in the presence of memory faults. Theor. Comput. Sci. 410(44), 4457–4470 (2009)
12. Finocchi, I., Grandoni, F., Italiano, G.F.: Resilient dictionaries. ACM Transactions on Algorithms 6(1) (2009)
13. Italiano, G.: Resilient Algorithms and Data Structures. In: Algorithms and Complexity (2010)
14. Jacob, B., Ng, S., Wang, D.: Cache, DRAM, Disk. Morgan Kaufmann (2008)
15. Knuth, D.: The Art of Computer Programming Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams. Addison-Wesley Professional (2009)
16. Liaw, H.T., Lin, C.S.: On the OBDD-representation of general Boolean functions. IEEE Transactions on Computers (1992)
17. Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In: ACM/IEEE 30th Design Automation Conference (DAC). pp. 272–277 (1993)
18. Minato, S.: Data Mining Using Binary Decision Diagrams. In: Progress in Representation of Discrete Functions, chap. 5, pp. 97–109. Morgan & Claypool (2010)
19. Minato, S.: Techniques of bdd/zdd: Brief history and recent activity. IEICE Transactions 96-D(7), 1419–1429 (2013)
20. Minato, S., Kimihito, I.: Symmetric item set mining method using zero-suppressed bdds and application to biological data. Information and Media Technologies 2(1), 300–308 (2007)
21. Requeno, J.I., Colom, J.M.: Compact representation of biological sequences using set decision diagrams. In: Rocha, M.P., Luscombe, N., Fdez-Riverola, F., Rodrguez, J.M.C. (eds.) 6th International Conference on Practical Applications of Computational Biology & Bioinformatics, Advances in Intelligent and Soft Computing, vol. 154, pp. 231–239. Springer Berlin Heidelberg (2012)
22. Taylor, D.: Error models for robust storage structures. In: 20th International Symposium on Fault-Tolerant Computing (1990)
23. Yoon, S., Nardini, C., Benini, L., De Micheli, G.: Discovering coherent biclusters from gene expression data using zero-suppressed binary decision diagrams. IEEE/ACM Trans. Comput. Biol. Bioinformatics 2(4), 339–354 (Oct 2005)