Complexity Reduction in Finite State Automata Explosion of Networked System Diagnosis

M. Ghasemzadeh*

Electrical and Computer Engineering Department, Yazd University, Yazd, Iran

ABSTRACT

This research puts forward the rough finite state automata which have been represented by two variants of Binary Decision Diagram (BDD) called ROBDD and ZBDD, for networked system diagnosis. Using the suggested data structures can help us overcome the combinatorial explosion which usually occurs in system diagnosis. In implementations and analysis of our experimental results, we used CUDD-Colorado University Decision Diagram package. A mathematical proof for the claimed complexity is provided which shows that ZBDD representation has superiority in space and time complexity to ROBDD representation.

doi: 10.5829/idosi.ije.2014.27.01a.04

1. INTRODUCTION

This paper addresses complexity reduction in explosion of networked system diagnosis. Networked systems have been modeled as discrete event systems (DESs). DESs have discrete states and events. When an event takes place, state of the DES is changed.

Diagnosability in discrete event systems was first investigated by Lin [1]. Briefly, it is concerned with the sequence of events to determine whether a system is operating correctly or a failure has already occurred. Since Diagnosability is crucially important, especially in large complex systems, it has received considerable attentions in science and industry.

For example, diagnosis in distributed telecommunication systems is a major concern, where the communication protocols and processes may be modeled as DESs.

When modeling a DES by finite state automata, usually an explosion in required memory and processing time happens which prevents every practical effort in diagnosis and verification?

2. BINARY DECISION DIAGRAMS

A binary decision diagram (BDD) is a data structure that is used to represent a Boolean function. BDDs can also be used to present compressed representation of sets or relations. A Boolean function can be represented as a rooted, directed, acyclic graph, which consists of decision nodes and two terminal nodes called 0-terminal and 1-terminal. Each decision node is labelled by a Boolean variable and has two children nodes called low child and high child. The edge from a node to a low (high) child represents an assignment of the variable to 0(1). Such a BDD is called ordered if different variables appear in the same order on all paths from the root. A BDD is said to be reduced if the following two rules have been applied to its graph: 1- Merge any isomorphic subgraphs. 2- Eliminate any node whose two children are isomorphic (non-effective nodes). The advantage of Reduced Ordered Binary Decision Diagram (ROBDD) is that it is canonical (unique) for a particular function. This property makes it useful in functional equivalence checking.

Most of the times when we address BDD, we mean ordered BDD or OBDD. An OBDD is a graphic description of an algorithm for the computation of a Boolean function. The following definition describes
the syntax of OBDD, i.e., the properties of the underlying graph. The semantics of OBDD, i.e., the functions represented by OBDD, is specified by the following definition.

2. 1. Definition 1. An OBDD $G$ representing the Boolean functions $f^1, \ldots, f^n$ over the variables $x_1, \ldots, x_n$ is a directed acyclic graph with the following properties:
1) For each function $f^i$ there is a pointer to a node in $G$.
2) The nodes without outgoing edges, which are called sinks or terminal nodes, are labelled by 0 or 1.
3) All non-sink nodes of $G$, which are also called internal nodes, are labelled by a variable and have two outgoing edges, a 0-edge and a 1-edge.
4) On each directed path in the OBDD, each variable occurs at most once as the label of a node.
5) There is a variable ordering $\pi$, which is a permutation of $x_1, \ldots, x_n$, and on each directed path, the variables occur according to this ordering. This means, if $x_i$ is arranged before $x_j$ in the variable ordering, then it must not happen which on some path, there is a node labelled by $x_j$ before a node labelled by $x_i$.

Zero-suppressed Binary Decision Diagram is a variant of BDD (in short ZBDD or ZDD). In this case, similar subgraphs merge as in BDDs [4]; the major difference is in the eliminating rule. In ZBDD, instead of non-effective nodes, the nodes which their high child is connected to 0-terminal are removed. In fact, ZBDD is introduced for representing and doing operations on sets and combinatorics. In this paper, we use ZBDD features in sets class. The 0-terminal node shows $\{\}$, and 1-terminal node shows $\{\}$. Each ZBDD shows a family of sets.

A path from the root to the 1-terminal represents a variable assignment for which the represented subset is determined. If one element of reference set exists, it is represented by 1 (high child), otherwise it is 0 (low child). In Figure 1, the class $\{\{a, b\}, \{a, c\}, \{c\}\}$ is shown by ZBDD and ROBDD. The advantage of ZBDD in nodes count, leads to reduction in time and space complexity. It is proved [2] that ZBDDs are more suitable for representing sets of sets than using OBDD for this purpose.

2. 2. The CUDD Package CUDD (Colorado University Decision Diagram) is a package which provides functions for manipulation of Binary Decision Diagrams (BDDs), Algebraic Decision Diagrams (ADDs), and Zero-suppressed Binary Decision Diagrams (ZDDs).

BDDs are used to represent switching functions; ADDs are used to represent functions from $\{0,1\}^n$ to an arbitrary set. ZDDs represent switching functions like BDDs. However, they are much more efficient than BDDs when the functions which being represented are characteristic functions of cube sets, or in general, when the ON-set of the function is very sparse. They are inferior to BDDs in other cases. The package provides a large set of operations on BDDs, ADDs, and ZDDs, functions to convert BDDs into ADDs or ZDDs and vice versa, and a large assortment of variable reordering methods.

The CUDD package can be used in three ways: As a black box, as a clear box and through an interface [3, 4].

![Figure 1](image-url) The left is ROBDD representation and the right is ZBDD representation of $\{\{a,b\},\{a,c\},\{c\}\}$. [2]

3. NETWORK SYSTEM MODEL

The system which is supposed to be verified is modeled as an automaton $A=(X, \Sigma, \delta, x_0)$. First, we reduce the finite state automaton $A$ by eliminating the non-accessible states. Let $P(X, \sigma)$ denote the power set of $X$, and $E \subseteq P(X_{\sigma})$. Here, we define two finite state automata $LA$ and $UA$ to be: $LA=(P(X, \Sigma, \delta_{LA}, x_{LA})$ and $UA=(P(X, \Sigma, \delta_{UA}, x_{UA})$.

It is obvious that the state spaces of the above automata are finite and the input symbol set of them is the same as the symbol set of $A$.

The transition functions $\delta_{LA}, \delta_{UA} : P(X) \times \Sigma \rightarrow P(X)$ are partial functions. In detail,

$$\delta_{LA}(X, \sigma) = |\{e \in E | e \subseteq \cup_{x \in X} (\delta_{A}(x, \sigma))\} \cup \{e \in E | e \cap \cup_{x \in X} (\delta_{A}(x, \sigma)) \neq \emptyset\}$$ (1)

In the second step, we try to construct $E$ to be;

$$e_{o} = \{x | \sigma \in \Sigma, \exists x' \in X_{\sigma}, \delta_{A}(x', \sigma) = x\},$$

$$e_{o'} = \{x | \sigma \in \Sigma, \exists x' \in X_{\sigma}, \delta_{A}(x', \sigma) = x\}$$

$$e_{e} = e_{o} - e_{o'}, \ E = \{e | e = e_{o} \lor e \in E(e_{o})\}.$$ (2)
In the following, we list the CUDD C++ interface code [3, 4] which adds a set of corresponding functions with a transition into the model of the system:

```
ZDD newTrans = (*mgr).zddOne(NumberOfVariables);
newTrans = ((newTrans.Change(source)).Change(symbol)).Change(target);
totalTrans = totalTrans.Union(newTrans);
```

We start with an empty set {}, then compute all states and transitions of the system and add them to our model, if it has not been added previously.

Next, we have to compute \( E \), which is provided by applying \( \text{Subset1} \) on \( \mathcal{A}_{\text{ZBDD}} \) and \( \text{CountMinterm} \). \( \text{Subset1} \) function computes the positive cofactor of the object with respect to the input. \( \text{CountMinterm} \) function returns the number of sets in the family. Then, for all \( x_i \) if \( (\mathcal{A}_{\text{ZBDD}}.\text{Subset}(x_i)).\text{CountMinterm}() \) is less than 1, it must be added to \( e'_{\mathcal{Z}} \), and if it is equal or greater than 1, it must be added to \( e_{\mathcal{Z}} \). Now, \( LA \) and \( RA \) can be computed. For this purpose, we apply an implicit depth first search, starting with \( \{0\} \). In each step, based on value of \( E \), a set is added to the family representing the transition functions of \( LA \) and \( UA \).

**3. 2. 2. Complexity Reduction in the Model of System by ROBDD** Here, we construct a Boolean function including some minterms over \( X \cup \Sigma \cup X' \), like before. All minterms contain only three positive variables which are source state, event and target state and others are negative. So, we have

\[
\mathcal{A}_{\text{ROBDD}} = \bigvee_{x_i \in X \cup \Sigma \cup X'} \bigwedge_{x_i' \in X} \bigl( -\overline{x_i} \bigr) \bigwedge_{x_i' \in \Sigma} (\overline{x_i'} \bigwedge \overline{x_i} \bigwedge x_i') \bigwedge_{x_i' \in X'} x_i' (4)
\]

**3. 2. 3. ZBDD vs. ROBDD** For studying complexity of the approach in worst case, we estimate the higher bound. In computing system model, the largest produced ZBDD data structure is for book keeping of transitions. Since each transition is a triple-element set and with assumption of having a deterministic machine, in worst case by ignoring ZBDD reduction rule about similar subgraphs, we will have \( 3 \times (|\Sigma| \times (|X| + 1)) \) nodes. This estimation is too pessimistic, because the appreciable superiority of ZBDD in representing sparse combination set is ignored. The largest ROBDD is also for book keeping of its transitions, which consists of
terms. This bound is also too pessimistic; here the merging and eliminating rules in construction of ROBDD are ignored. For example in our experimental data in CSMA/CD protocol, we had 21 nodes in $A_{ZBDD}$, 20 nodes in $LA_{ZBDD}$, 20 nodes in $LA_{ROBDD}$, comparing to 64 nodes in $A_{ROBDD}$, 58 nodes in $LA_{ROBDD}$, and 38 nodes in $LA_{ROBDD}$. In an experimental large grid networked system, we had 56712 nodes in $A_{ZBDD}$, comparing with 237845 nodes in $A_{ROBDD}$. We can see from formal analysis as well as from experimental results that ZBDDs have advantages in complexity reduction in finite state automata explosion of networked system diagnosis and verification.

4. CONCLUDING REMARKS

Using traditional methods in networked system diagnosis leads to exponential space and time algorithms which have no practical use except for some of very small networks. This research shows how we can benefit from ROBDD and ZDD which are powerful data structures to overcome this dilemma.

5. REFERENCE