IJE TRANSACTIONS A: Basics Vol. 27, No. 1, (January 2014) 29-32

International Journal of Engineering

Journal Homepage: www.ije.ir

RESEARCH NOTE

Complexity Reduction in Finite State Automata Explosion of Networked System Diagnosis

M. Ghasemzadeh*

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

PAPER INFO

Paper history:
Received 06 April 2013
Received in revised form 31 May 2013
Accepted 20 June 2013

Keywords:
Complexity
Finite State Automata
Networked System Diagnosis
ROBDD

ABSTRACT

This research puts forward the rough finite state automata which have been represented by two variants of BDD (Binary Decision Diagram) 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.v

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.

Diagnosablity 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 Diagnosablity 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?

*Corresponding Author Email: m.ghasemzadeh@yazd.ac.ir (M. Ghasemzadeh)

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 labeled 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 $f^1,\ldots, f^m$ 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 labeled by 0 or 1.
3) All non-sink nodes of G, which are also called *internal* nodes, are labeled 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 π, 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 labeled by x_j before a node labeled 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].

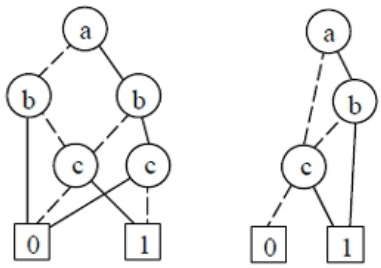

Figure 1. The left is ROBDD representation and the right is ZBDD representation of {{a,b},{a,c},{c}}. [2]

3. NETWORK SYSTEM MODEL

The system which is supposed to be verified is modeled as an automaton $A=(X_A,\Sigma,\delta_A,x_{0_A})$. First, we reduce the finite state automaton A by eliminating the non-accessible states. Let $P(X_A)$ denote the power set of X_A and $E\subseteq P(X_A)$. Here, we define two finite state automata LA and UA to be: $LA=(P(X_A),\Sigma,\delta_{LA},\{x_{0_A}\})$ and $UA=(P(X_A),\Sigma,\delta_{UA},\{x_{0_A}\})$.

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 $\delta_{LA},\delta_{UA}: P(X_A)\times\Sigma\to P(X_A)$ are partial functions. In detail,

$$\delta_{LA}(X,\sigma)=\cup\{e\in E\mid e\subseteq \cup_{x\in X}(\delta_A(x,\sigma))\},\ \delta_{UA}(X,\sigma)=\cup\{e\in E\mid e\cap\cup_{x\in X}(\delta_A(x,\sigma))\neq\varnothing\} \quad (1)$$

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

$$e_\sigma=\{x\mid\sigma\in\Sigma,\exists x'\in X_A,\delta_A(x',\sigma)=x\},\quad e''_\sigma=\{x\mid\sigma\in\Sigma,\exists! x'\in X_A,\delta_A(x',\sigma)=x\},\quad e'_\sigma=e_\sigma-e''_\sigma,\ E=\{e\mid e=e_\sigma\vee e\in P(e'_\sigma)\}. \quad (2)$$

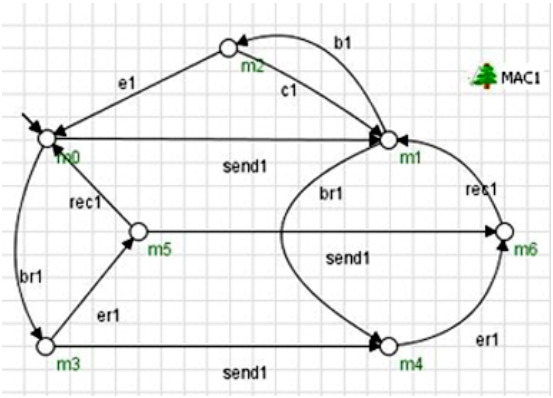

Figure 2. The CSMA/CD protocol specification model

3. 1. Example: CSMA/CD Protocol In order to better understand, first we introduce the CSMA/CD protocol model. For simplicity, we will consider the case in which two identical stations are connected. The service which is provided by MAC is an error free bidirectional communication. The protocol specification for the MAC consists of two MAC entities one in each station. Figure 2 shows one of these entities. Initially, a message from LLC (*send*) may be accepted. The MAC initiates transmission (*b*), unless a message is in the process of being received. If the transmission is successfully terminated (*e*), then a new message may be accepted and the process is repeated. If a collision occurs (*c*) before termination, then MAC attempts to retransmit it after a short time of waiting. In some states, a message may be received (*br*). Following such an event, MAC may not start any transmission until all of the messages have been received (*er*) and the LLC has been notified (*rec*). However, MAC may accept a transmission request from LLC (*send*) in any state, unless such a request is pending.

3. 2. Networked System Diagnosis with OBDD and ZBDD In fact, we are interested to abate the exponential need for memory space using BDD or its variants. Since ZBDD is a brilliant data structure in set representation, we formulate the problem to benefit from this property.

3. 2. 1. Complexity Reduction in Model of System by ZBDD We construct a family including some sets over $X_A \cup \Sigma \cup X'_A$, where X and Σ are as usual. And X' is a set including target states of transitions which is an isomorphism of X. All subsets contain only three elements which are source state, event and target state. So, we have

$$A_{ZBDD} = \cup\{\{x_{i_A}, \sigma, x'_{i_A}\} \mid \forall x_{i_A}, x''_{i_A} \in X_A, \forall \sigma \in \Sigma, \delta_A(x_{i_A}, \sigma) = x''_{i_A}, X'(x''_{i_A}) = x'_{i_A}\} \quad (3)$$

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 *Subset1* on A_{ZBDD} and *CountMinterm*. Subset1 function computes the positive cofactor of the object with respect to the input. *CountMinterm* function returns the number of sets in the family. Then, for all x'_{i_A} if $(A_{ZBDD}.Subset1(x'_{i_A})).CountMinterm()$ is less than 1, it must be added to e''_σ, and if it is equal or greater than 1, it must be added to e_σ. Now, *LA* and *RA* can be computed. For this purpose, we apply an implicit depth first search, stating with $\{x_{0_A}\}$. 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 $X_A \cup \Sigma \cup X'_A$, like before. All minterms contain only three positive variables which are source state, event and target state and others are negative. So, we have

$$A_{ROBDD} = \bigvee_{\forall x_{i_A}, x''_{i_A} \in X_A, \forall \sigma \in \Sigma, \delta_A(x_{i_A}, \sigma) = x''_{i_A}, X'(x''_{i_A}) = x'_{i_A}} \left(\bigwedge_{x_j \neq x_i} (\overline{x_j}) \wedge \bigwedge_{\sigma' \neq \sigma} (\overline{\sigma'}) \wedge \bigwedge_{x'_j \neq x'_i} (\overline{x'_j}) \wedge (x_i \wedge \sigma \wedge x'_i) \right) \quad (4)$$

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 $3 \times (|\Sigma| \times (|X_A| + 1))$ 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

$|\Sigma|\times(|X_A|+1)\times(|\Sigma|+(2\times|X_A|))$ 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_{ZBDD}, 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

1. Lin, F., "Diagnosability of discrete event systems and its applications", ***Discrete Event Dynamic Systems***, Vol. 4, No. 2, (1994), 197-212.
2. Minato, S.-i., "Zero-suppressed bdds and their applications", ***International Journal on Software Tools for Technology Transfer***, Vol. 3, No. 2, (2001), 156-170.
3. Mishchenko, A. *Extra library for zdd.* Available from: http://www.ee.pdx.edu/ alanmi/research/extra.htm.
4. Somenzi, F. *Cudd package, realease 2.3.1.* Available from: http://vlsi.colorado.edu/ fabio/cudd/cuddIntro.html.

Complexity Reduction in Finite State Automata Explosion of Networked System Diagnosis

RESEARCH NOTE

M. Ghasemzadeh

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

P A P E R I N F O

Paper history:
Received 06 April 2013
Received in revised form 31 May 2013
Accepted 20 June 2013

Keywords:
Complexity
Finite State Automata
Networked System Diagnosis
ROBDD

چکیده

در این پژوهش با بکارگیری یک گونه خاص از نمودار دودویی تصمیم برای نمایش ماشین حالت محدود نشان داده می شود که چگونه می‌توان این ساختار را به طور مفیدی برای عیب‌یابی سیستم‌های شبکه‌ای به کار برد. متد پیشنهادی این مزیت را در اختیار می‌گذارد تا درگیر تعداد نمایی از ترکیب‌های ممکن نشویم. برای پیاده‌سازی متد پیشنهادی از بسته استاندارد ارائه شده توسط دانشگاه کلورادو بهره برده‌ایم. یک اثبات ریاضی نیز برای این ادعا که بکارگیری نمودار دودویی تصمیم مورد نظر می‌تواند بسیار بهتر عمل کند ارائه شده است.

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