Reliability Engineering and System Safety 111 (2013) 206–216

Contents lists available at SciVerse ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Solving dynamic flowgraph methodology models using binary decision diagrams

Kim Bjorkman

VTT Technical Research Centre of Finland, Systems Research, P.O. Box 1000, FI-02044 Espoo, Finland

ARTICLE INFO

Article history:
Received 18 January 2012
Received in revised form
2 November 2012
Accepted 7 November 2012
Available online 22 November 2012

Keywords:
Binary decision diagram
Dynamic flowgraph methodology
Dynamic reliability analysis
Multi-valued logic
Prime implicant

ABSTRACT

Dynamic flowgraph methodology (DFM) is a computationally challenging approach to the reliability analysis of dynamic systems with feedback loops. To improve the computational efficiency of DFM modelling, we propose a new approach, based on binary decision diagrams (BDDs), to solving DFM models. The objective of DFM analysis is to identify the root causes of a postulated top event. The result is a set of prime implicants that represent system faults resulting from diverse combinations of software logic errors, hardware failures, human errors and adverse environmental conditions. Two approaches to solving prime implicants have been implemented in software called YADRAT. The first approach is based on meta-products, and the second on zero-suppressed BDDs (ZBDD). Both approaches have been used previously in fault tree analysis. In this work, the ideas of prime implicant computations are adapted to a dynamic reliability analysis approach combined with multi-valued logic. The computational efforts required for the two approaches are compared by analysing three example systems. The results of the comparison show that BDDs are applicable in DFM computation and that in particular the ZBDD-based approach can solve moderately sized DFM models in a reasonable time.

1. Introduction

One of the most widely used methodologies for reliability modelling, particularly in the nuclear domain, has been the static event tree/fault tree (ET/FT) approach. The inability of the ET/FT approach to capture time-dependent dynamic behaviours has made it somewhat impractical in assessing the reliability of dynamic systems. Dynamic methodologies can provide a more accurate representation of probabilistic system evolution in time than the ET/FT approach.

Numerous dynamic reliability approaches are available, such as dynamic flowgraph methodology (DFM) [1–3], Markov/cell-to-cell mapping technique (CCMT) [4,5], Petri Nets [6], Bayesian approaches [7–9], test-based approaches [10], Boolean logic driven Markov process (BDMP) [11], and black box approaches [12,13]. Some of these approaches are reviewed in, for example, [10,14,15] and Markov CCMT were ranked in [10] as the two top dynamic reliability modelling approaches, with the most positive features and fewest negative features. The Markov/CCMT and the DFM approaches are not compensatory methods; rather they should be used in a complementary fashion. It is suggested in [4] that DFM should be used to identify possible failure sequences or initiating events that lead to a specified event. In contrast, Markov/CCMT should be used to guarantee the completeness and verification of the quantification of the failure sequences that may require more detailed modelling.

DFM is an approach to modelling and analysing the behaviour of dynamic systems for reliability assessment. DFM can be used to identify how certain top events (typically a system failure) may occur in a system. The result is a set of prime implicants that represents system faults resulting from diverse combinations of software logic errors, hardware failures and adverse environmental conditions. A DFM analysis corresponds to a minimal cut set search of a fault tree and prime implicants are similar to the minimal cut sets of fault tree analysis.

The modelling approach used by DFM is promising, due to the simplicity of its formalism, the possibility of modelling time dependencies and loop dependencies and the possibility of modelling multi-state logic and incoherent reliability structures. The main drawback and limitation of DFM is scalability. Realistic modelling easily causes a combinatorial explosion as the number of states in the decision tables increases.

The main of objective of this paper is to present a new approach to solving DFM models with the aim of improving scalability. The novelty of the approach is that it employs binary decision diagrams (BDD) [16,17] to represent a DFM model. The BDD was chosen as the underlying data structure, because BDDs are state-of-the-art data structures that have been used in several applications, and several efficient open-source BDD packages are available.

E-mail address: kim.bjorkman@vtt.fi

0951-8320/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ress.2012.11.009

The two different BDD-based approaches have been implemented for the exact computation of prime implicants in software called YADRAT. The first approach is based on meta-products [18] and the second on zero-suppressed BDD (ZBDD) [19]. Both approaches have been used previously in static failure tree analysis [20,21]. In this work, the ideas for prime implicant computation are adapted to a dynamic reliability analysis approach combined with the multi-valued logic of DFM.

The remainder of this paper is structured as follows: Section 2 introduces binary decision diagrams and the dynamic flowgraph methodology. Section 3 reviews related work. Section 4 discusses the algorithms in YADRAT. Section 5 compares two BDD-based algorithms and, also, compares the YADRAT with another DFM software called DYMONDA [22]. The aim of the latter comparison is not to compare computational efficiency but to demonstrate that BDD-based approaches can be used to solve DFM models correctly. Section 6 concludes the paper.

2. Preliminaries

2.1. Boolean algebra

A Boolean function of n variables is a function on B^n into B, where B is the set $\{0,1\}$, n is a positive integer, and B^n denotes the n-fold Cartesian product of the set B [23]. A literal is a Boolean variable x or its complement $\overline{x}$. A product term p is a single literal or a logical product of two or more literals. P_n is the set of products that can be built out of a set of variables $\{x_1, \ldots, x_n\}$ [18].

Let V be a set of Boolean variables. An assignment ξ is any mapping from V to [0,1]. An assignment satisfies a formula F if $\xi[F]=1$, otherwise it falsifies F [20]. A logical function $G(x_1, \ldots, x_n)$ implies a Boolean function $F(x_1, \ldots, x_n)$ if any assignment that satisfies G also satisfies F. A prime implicant of a Boolean function $F(x_1, \ldots, x_n)$ is a normal product term $G(x_1, \ldots, x_n)$ that implies F, such that if any variable is removed from G, then the resulting product term does not imply F.

A set of prime implicants that is logically equivalent to a function is called cover [24]. In the scope of this paper two types of covers are of interest: irredundant and complete. A set of products P is a prime cover of a Boolean function F if it is made of prime implicants of F. Cover P is irredundant if there is no proper subset of P that is a prime cover of F. The complete cover includes all the possible prime implicants of F.

2.2. Binary decision diagram

A binary decision diagram [16,17] is a data structure used to represent Boolean functions. The BDD is based on the repeated application of the classic Shannon expansion formula

$$F = x \cdot F|_{x=1} + \overline{x} \cdot F|_{x=0} \tag{1}$$

A BDD is a rooted, directed acyclic graph consisting of decision nodes with two edges the 1-edge and 0-edge, and two terminal nodes the 1-terminal and the 0-terminal representing the Boolean functions 0 and 1. A variable assignment for which the represented Boolean function is true is represented by a path from the root node to the 1-terminal node.

An ordered binary decision diagram (OBDD) is a BDD with the constraint that the input variables are ordered and every decision node to terminal node path in the OBDD visits the input variables in ascending order. By reducing the OBDD, the reduced ordered binary decision diagram (ROBDD) is obtained. Bryant [16] demonstrated how a BDD could be modified to an ROBDD to enable the creation of a canonical representation a Boolean function. In the rest of this paper, a BDD is understood to mean the reduced and ordered form of a BDD.

The ite (If–Then–Else) connective is generally used for the representation of Boolean functions as BDDs. The ite connective can be used to implement all two-variable Boolean functions. It is defined as follows:

$$ite(F,G,H) = (F \cdot G) + (\overline{F} \cdot H) \tag{2}$$

where F, G, and H are Boolean functions.

The performance of the ite connective can be improved by introducing a computation table that maintains the results of previous computations [16]. When the algorithm is used for two nodes, first the computation table is checked. If the table contains an entry of the computation then the result can be returned immediately.

Another type of BDD is the zero-suppressed BDD (ZBDD) introduced by Minato [19]. A ZBDD is a BDD with different semantics and reduction rules. ZBDDs are more efficient than regular BDDs when the Boolean function handled is very sparse, i.e. when a formula is zero almost everywhere. Set-operations on ZBDDs are of polynomial worst-case complexity, as for logical operations on BDDs. Basic operations on ZBDDs are presented in [19].

The size of a BDD is heavily dependent on the chosen variable order. Generally, there are two different approaches to determining the variable ordering used: static heuristics and dynamic heuristics [25]. One of the most used heuristics is depth-first left-most heuristic (DFLM). In DFLM, variables are numbered on the basis of the depth-first left-most traversal of a formula. In practice, DFLM heuristics gives rather good results compared with other approaches suggested in the literature. However, some questions have been raised regarding the goodness of the heuristics [26].

2.3. Dynamic flowgraph methodology

Dynamic flowgraph methodology is an approach to modelling and analysing the behaviour of dynamic systems for reliability assessment and verification [1]. In DFM models, the logic of the system is expressed in terms of causal relationships between physical variables and states of the system. The time aspects of the system associated with, for example, the execution of control commands or the dynamics of the process are represented as a series of discrete state transitions and time delays. DFM can be used to identify how certain postulated events may occur in a system. DFM has been used to assess the reliability of nuclear power plant control systems [27], space rockets [3] and chemical batch processes [28].

DFM models are directed graphs that are analysed at discrete time instances. A node represents a variable that can be in one of a finite number of predefined states. The state of a node can change at discrete time instances. The state of the node is determined by the states of its input nodes at a single instance of time and the time lag that specifies how many time instances it takes for an input to affect the state of the present node. The state of a node, as a function of the states of its input nodes, is determined by a decision table. A decision table is an extension of the truth table in which each variable can be represented with any finite number of states.

A DFM model can be analysed in two different modes: deductive and inductive. In inductive analysis, event sequences are traced from causes to effects, corresponding to simulation of the model. In deductive analysis, event sequences are traced backwards from effects to causes. In this paper, only deductive analysis is considered.

A deductive analysis begins with the identification of a certain system condition of interest (a top event), corresponding typically