Contents

Handling very large event trees by means of Binary Decision Diagrams

F. Ducamp & S. Planchon
Institut de Protection et Sreté Nucléaire, DES/SERS B.P. 6, 92265 Fontenay-aux-Roses Cedex FRANCE
{ducamp,planchon}@uranie.ipsn.fr
A. Rauzy & P. Thomas
CNRS-LaBRI, Université Bordeaux I, 351, cours de la Libération F-33405 Talence cedex FRANCE
{rauzy,thomas}@labri.u-bordeaux.fr

Abstract

This article is about the various pre-processing techniques, computation algorithms and approximation schemes the authors develop to assess very large event trees coming from nuclear PSA. As a core algorithm, we use Binary Decision Diagrams.

1. Introduction

Since about twenty years and the WASH-1400 report, event trees and fault trees techniques are widely used for nuclear risk assessment studies. The joint use of these techniques is nowadays well understood by practitioners as a modelling tool. However, problems still remain to exploit it fully because of the computational complexity of the underlying problems (minimal cutsets computation, probabilistic assessments).

Most of the computer codes implement MOCUS like algorithms. Several approximations are associated with this class of algorithms. On the one hand, these approximations are necessary to reduce the exponential blow up. On the other hand, no one is able to ensure that the results obtained in this way are accurate.

As an alternative, Bryant's Binary Decision Diagrams (BDDs) [1] have been recently introduced in the reliability fields [2]. They have proved to be a very powerful tool to assess Boolean models (see for instance [3]). Not only they make it possible to compute efficiently minimal cutsets and top event probabilities, but also the results are exact (no approximation is performed). However, problems

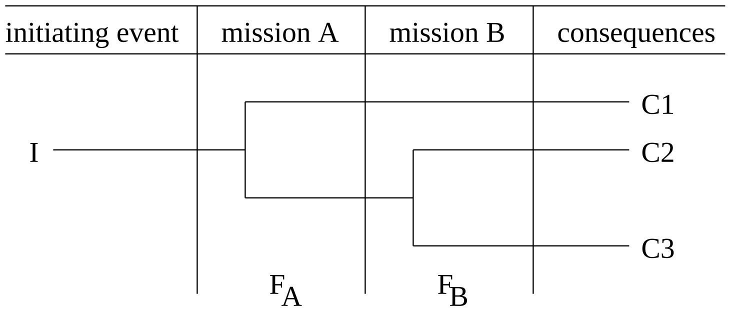

Fig. 1. An event tree

still remain to assess very large models because BDDs are subject to exponential blow up as well. However, the specific nature of event trees coming from PSA makes room for specific improvements of the method.

This work is a part of the Aralia/Hévéa project, a collaboration of several institutions and companies, including the French nuclear regulatory institution (IPSN) and the French center for scientific research (CNRS). It aims to develop a BDD based computer code to assess event trees. Aralia is a BDD package extended with many PSA features, while Hévéa is the event tree manager.

2. Event Trees

A sequence of an event tree starts from an initiating event, goes through a number of fork and ends with a consequence. Each fork corresponds to a mission, i.e. the response of a safety system. The failures of such a system are described by means of a fault tree. In the nuclear PSA framework, the fault trees are in general coherent, i.e. are monotone Boolean formulae. Moreover, basic event probabilities are low. The success of the safety system mission is modelled by the negation of the fault tree (its failure is modelled by the fault tree itself). A sequence is compiled into the conjunct of the Boolean formulae (fault trees or negation of fault trees) that model the branches it goes through. A set of sequences is compiled into the disjunct of the formulae encoding its members.

As an illustration consider the event tree pictured on Fig. 1.. It made of one initiating event I, two missions and the corresponding fault trees F_A and F_B, and three consequences $C1$, $C2$ and $C3$. Upper branches represent successes of missions. The three sequences $I-C1$, $I-C2$ and $I-C3$ are therefore encoded by the following conjuncts. $I-C1 = I.\bar{F_A}$, $I-C2 = I.F_A.\bar{F_B}$ and $I-C3 = I.F_A.F_B$.

The sequences of event trees we are dealing with are typically made of about 10 fault trees, 500 up to 1000 basic events and 500 up to 1500 gates. The corresponding formulae are very specific. This makes it possible to develop specific mathematical frameworks, algorithm schemes and heuristics to handle them.

3. Conventional approximation schemes

Most of the computer codes implement MOCUS like algorithms to assess event trees. Each sequence is considered in turn. First, minimal cutsets of the sequence are computed (the MOCUS algorithm works in a top-down way, but bottom-up algorithms are sometimes used as well). Then, probabilistic quantities are assessed by means of the Sylvester-Poincare development: let $\pi_1, \ldots, \pi_n$ be the minimal cutsets obtained during the first step. Then, the probability $p(top)$ of the top-event is as follows:

$$p(top) \quad = \quad \sum_{i=1,n} p(\pi_i) - \sum_{i=1,n-1,j=i+1,n} p(\pi_i.\pi_j) + \ldots \tag{1}$$

This algorithmic scheme induces three kinds of approximations.
– In case of non-monotone logic, a function is not equivalent to the disjunction of its minimals cutsets. Consider for instance, the function $F = ab + \bar{a}c$. Minimal cutsets of F are ab and c, and therefore $F \not\equiv ab + c$ (see [4] for detailed discussion on this topics). Event tree sequences are non-monotone for they include negations.
– Since it is too costly to determine all of the minimal cutsets only the most important ones (from a probabilistic point of view) are considered. This may induce significant deviations.
– Since it is too costly to apply the full Poincare development, only the very first terms are taken into account.

These three approximation schemes are in general safe for probabilities of basic events are low. However, there is no way to ensure that the error is under a given bound. Efficient methods to compute exact results do exist, as examplified by Binary Decision Diagrams.

4. Binary Decision Diagrams

Bryant's Binary Decision Diagrams (BDDs) [5, 1] are the state-of-the-art data structure to encode and to manipulate Boolean functions. Since their introduction in the reliability analysis framework (by one of the author among others [2]), BDDs have proved to be the most efficient technique to assess fault trees.

The BDD associated with a formulae is a compact encoding of the truth table of this formula. This representation is based on the Shannon decomposition. Let F be a Boolean formula that depends on the variable v, then there exists

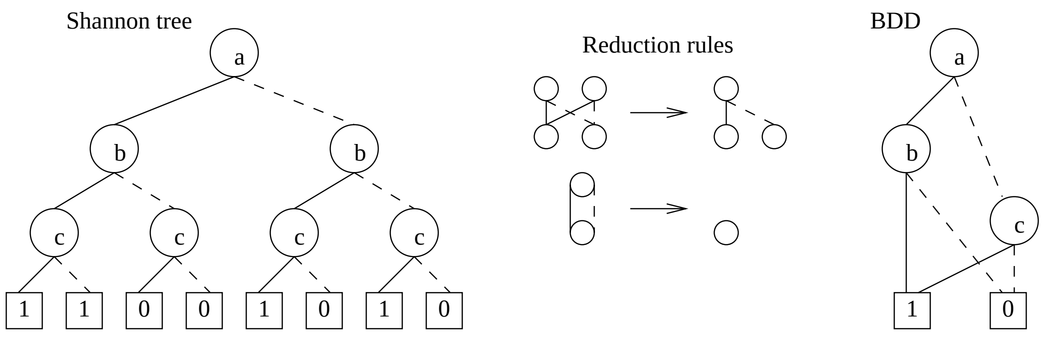

Fig. 2. From the Shannon tree to the BDD encoding $ab + \bar{a}c$.

two formulae F_1 and F_0 not depending on v such that $F = v.F_1 + \bar{v}.F_0$. By choosing a total order over the variables and applying recursively the Shannon decomposition, the truth table of any formula can be graphically represented as a binary tree. The nodes are labeled with variables and have two outedges (a *then*-outedge, pointing to the node that encodes F_1, and a *else*-outedge, pointing to the node that encodes F_0). The leaves are labeled with either **0** or **1**. The value of the formula for a given variable assignment is obtained by descending along the corresponding branch of the tree. The Shannon tree for the formula $ab + \bar{a}c$ and the lexicographic order is pictured Fig. 2. (dashed lines represent *else*-outedges).

Indeed such a representation is very expensive. It is however possible to shrink it by means of the following two reduction rules.

– Isomorphic subtrees merging. Since two isomorphic subtrees encode the same formula, at least one is useless.

– Useless nodes deletion. A node with two equal sons is useless since it is equivalent to its son $(v.F + \bar{v}.F = F)$.

By applying these two rules as far as possible, one get the BDD associated with the formula. A BDD is therefore a directed acyclic graph. It is unique, up to an isomorphism. This process is illustrated on Fig. 2..

The full binary tree is never built: logical operations (and, or, xor, ...) can be directly performed on BDDs. They are polynomial in the sizes of their operands. Therefore, the BDD encoding a formula is obtained from the BDDs that encode its subformulae. A complete implementation of a BDD package is described in [1]. The reader interested in details should thus refer to this article.

Given the BDD $F = \langle v, F_1, F_0 \rangle$ encoding a fault tree (or a event tree), the probability of the top event can be assessed in linear time (w.r.t. the size of the BDD) by applying the Shannon decomposition $p(F) = p(v).p(F_1) + [1 - p(v)].p(F_0)$ (see [2] for more details).

Minimal cutsets can be computed as well. The principle is to compute a

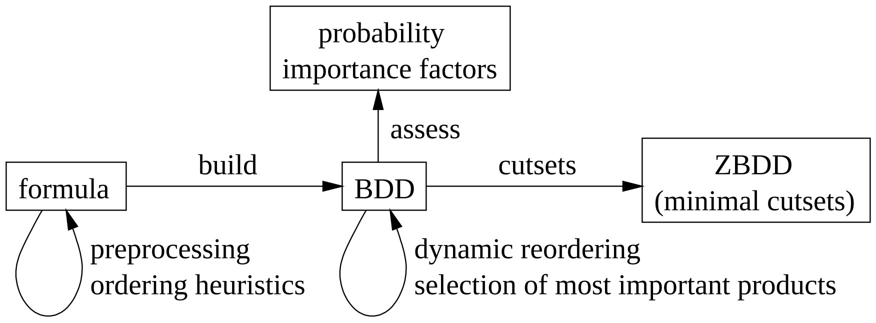

Fig. 3. Road map

second BDD that encodes the cutsets (see [2, 6]). This process is illustrated by the upper part of Fig. 3..

5. Algorithms and heuristics

As the other methods, the efficiency of BDDs is sensitive to a number of factors such as the way the formula is written, the order chosen among and variables. Below, we list a number of techniques we use in order to improve the efficiency of the method on the particular class of formulae we are dealing with.
Pre-processing of the formulae: the formula encoding a set of sequences is a disjunction of conjunctions of literals, where each literal is either of fault tree or its negation. It is possible to factorise the formula in order to reduce the number of auxiliary computations, and therefore to achieve substantial memory and time savings. It appears also that substantial improvements can be achieved by permuting the arguments of associative/commutative connectives (see [7] for a discussion on that topics).
Variable ordering heuristics: BDDs require to chose an order among the variables of the formula under study. Here again, the specific nature of the formulae makes it possible to develop specific variable ordering heuristics.
Dynamic variable reordering: it is sometimes of interest to change dynamically the variable ordering. This technique has proved to be very powerful in the case of circuit analyses [8]. In our case, this technique seems to too time consuming to be used. However, as noted in [7], a good strategy consists in drawing more or less at random a number of rewritings (and therefore variable orderings) of the formula and to restart the computations from scratch until a good rewriting is found.
Approximations: in [4], we showed how the concept of Boolean sub-algebrae could be used to perform approximated computations. We show that this technique is of a special interest in the case of nuclear PSA event tree. We show that it makes

it possible to tune the approximations.

The combination of the above techniques makes it possible to improve by orders of magnitude the efficiency of BDD assessments and, in some cases, to handle event trees that would be not tractable otherwise.

References

[1] K. Brace, R. Rudell, and R. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference.* IEEE 0738, 1990.

[2] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering & System Safety*, 05(59):203–211, 1993.

[3] S. Combacon, Y. Dutuit, A. Laviron, and A. Rauzy. Comparison between two tools (Aralia and ESCAF) applied to the Study of the Emergency Shutdown System of a Nuclear Reactor. In A. Mosleh and R.A. Bari, editors, *Proceedings of the International Conference of Probabilistic Safety Assessment and Management, PSAM'4*, volume 2, pages 1019–1024, New-York, 1998. Springer Verlag.

[4] Y. Dutuit and A. Rauzy. A guided tour of minimal cutsets handling by means of binary decision diagrams. In *Proceedings of Probabilistic Safety Assessment conference, PSA'99*, volume 2, pages 55–62. American Nuclear Society, 1999. ISBN 0-89448-640-3.

[5] R. Bryant. Graph Based Algorithms for Boolean Fonction Manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[6] Y. Dutuit and A. Rauzy. Exact and Truncated Computations of Prime Implicants of Coherent and non-Coherent Fault Trees within Aralia. *Reliability Engineering and System Safety*, 58:127–144, 1997.

[7] M. Bouissou, F. Bruyère, and A. Rauzy. BDD based Fault-Tree Processing: A Comparison of Variable Ordering Heuristics. In C. Guedes Soares, editor, *Proceedings of European Safety and Reliability Association Conference, ESREL'97*, volume 3, pages 2045–2052. Pergamon, 1997. ISBN 0-08-042835-5.

[8] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. In *Proceedings of IEEE International Conference on Computer Aided Design, ICCAD'93*, pages 42–47, November 1993.