Annals of Nuclear Energy 90 (2016) 22–25

Contents lists available at ScienceDirect

Annals of Nuclear Energy

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

Efficient reduction and modularization for large fault trees stored by pages

Shanqi Chen [a,b], Jin Wang [a], Jiaqun Wang [a], Fang Wang [a,*], Liqin Hu [a,b]

[a] Key Laboratory of Neutronics and Radiation Safety, Institute of Nuclear Energy Safety Technology, Chinese Academy of Sciences, Hefei, Anhui 230031, China
[b] University of Science and Technology of China, Hefei, Anhui 230031, China

ARTICLE INFO

Article history:
Received 16 July 2015
Received in revised form 18 September 2015
Accepted 3 November 2015
Available online 17 December 2015

Keywords:
Fault tree
Simplification
Modularization
Probabilistic Risk Assessment

ABSTRACT

Fault Tree Analysis (FTA), an indispensable tool used in Probabilistic Risk Assessment (PRA), has been used throughout the commercial nuclear power industry for safety and reliability analyses. However, large fault tree analysis, such as those used in nuclear power plant requires significant computer resources, which makes the analysis of PRA model inefficient and time consuming. This paper describes a fault tree pre-processing method used in the reliability and probabilistic safety assessment program RiskA that is capable of generating minimal cutsets for fault trees containing more than 10,000 gates and basic events. The novel feature of this method is not only that Boolean reduction rules are used but also that a new objective of simplification is proposed. Moreover, since the method aims to find more fault tree modules by the linear-time algorithm, it can optimize fault tree modularization, which further reduces the computational time of large fault tree analysis.

1. Introduction

Fault Tree Analysis (FTA), an indispensable tool in Probabilistic Risk Assessment (PRA), has been used throughout the commercial nuclear power industry for safety and reliability analyses. However, large fault tree analysis for nuclear power applications require significant computer resources, which makes the analysis of PRA model inefficient and time consuming. Fault tree pre-processing mainly includes fault tree simplification and fault tree modularization, which are performed before qualitative and quantitative fault tree analysis. Pre-processing significantly influences the calculation speed and is also important in the overall process of fault tree analysis.

The primary purpose of fault tree simplification is to simplify the fault tree structure by pruning the redundant nodes or sub-trees. Simplification reduces the size of large fault tree, which reduces the calculation complexity. The earliest algorithm (Bengiamin et al., 1976) proposed to simplify the fault tree, which eliminates repeated events inputted to OR gates, was very simple. The algorithm based on Program Package for Evaluation of Fault Trees (FAUNET) rules (Platz and Olsen, 1976) was one of the widely used traditional simplification approaches. Its' process used several Boolean reduction rules to reduce the fault tree size without changing the logic structure. Additional rules, such as elimination (Sun and Andrews, 2004) were added by later scholars. A new fault tree reduction methodology in Integrated Reliability and Risk Analysis System (IRRAS) algorithms (Russell and Rasmuson, 1993) was proposed by using some bottom-up techniques, which took advantage of several optimization methods to restructure and prune the fault tree including independent sub-trees, probability pruning, and coalescing of gates. Fault tree optimization algorithms used in the Finnish Centre for Radiation and Nuclear Safety (STUK) PSA (SPSA) code (Niemela, 1994) did not use any Boolean reduction rule, since the fault tree itself contains simplification rules.

Modularization (Chatterjee, 1975) of fault trees is performed on the simplified fault tree structure. It further reduces the computing complexity significantly by dividing a large fault tree into small pieces which can be translated into cutsets independently. Many improvements of modularization were performed (Wilson, 1985; Camarinopoulos and Yllera, 1986; Kohda et al., 1989). After a linear-time algorithm to find fault tree modules was developed by Dutuit and Rauzy (1996), modularizing the fault tree itself requires little processing time in comparison with the tree processing time. However, different structures of the same fault tree would result in different accounts of modules. Use of this methodology will results in modules that are either too large or too small; therefore, it is difficult to obtain appropriate sizes of modules.

All of the simplification methods discussed above do not consider the effect of fault tree structure with respect to

* Corresponding author.
E-mail address: fang.wang@fds.org.cn (F. Wang).

http://dx.doi.org/10.1016/j.anucene.2015.11.002
0306-4549/

modularization. A simplification algorithm, based upon process of paging stored fault trees and a heuristic decision of when to reconstruct fault tree pages in order to obtain more modules, is presented by this paper. The novel feature of the proposed algorithm is not only that Boolean reduction rules were used but also that a new objective of simplification was proposed. The objective is aimed on obtaining more modules. Therefore, this pre-processing algorithm can help both the fault tree simplification and the modularization, which significantly reduces the computation time.

2. Methodology

2.1. A brief review of Boolean rules

A brief review of Boolean rules is presented here, which usually consist of four rules.

(1) Contraction: subsequent gates of the same type are contracted to form a single gate. This reconstructs the fault tree as an alternating sequence of AND gates and OR gates.
(2) Extraction: the purpose is to identify the common factor.
(3) Factorization: pairs of events that always occur together under the same type of gates are identified and combined to form a single complex event.
(4) Elimination: using the Boolean law of absorption, like: $a + (a * b) = a$, $a * (a + b) = a$.

The application of the above rules is to simplify a fault tree to its minimal form; that is, the fault tree cannot be simplified further by using the above simplification rules. However, it does not work very well with the fault tree structure stored in pages.

2.2. Paging storage

Fault tree structure can be imported from the code database or models in other codes, like the FTP format model used by CAFTA (Jim et al., 1986), RSA format model used by RiskSpectrum (Berg, 1990), and XML format model used by XFTA (Rauzy, 2012). When the fault tree model is very large similar to those developed for nuclear power plant, it requires large amount computer resources (RAM) and computation time, so it is difficult to be imported and stored directly.

In order to optimize use of computer memory and to improve the importing speed, fault trees are stored by pages which are connected by transfer gates. This results in the same fault tree structure existing in more than one place which can be stored independently as a single page. This single page is linked by a transfer gate which replaces the position of the top gate of the structure. For example, Fig. 1 shows a fault tree that is not stored by pages, and Fig. 2 depicts the fault tree as two pages which are connected by a TRANSFER gate T_G2.

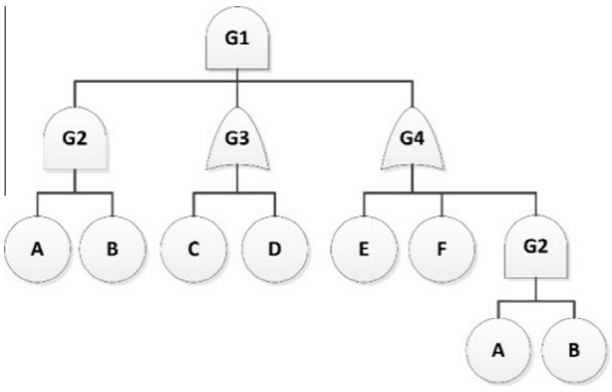

Fig. 1. Fault tree structure without paging storage.

2.3. Pre-processing method

After importing the fault tree structure, simplification will be performed in order to reduce the fault tree to its most concise form without changing its logic functions.

In order to simplify the paging stored fault trees to its minimal form, a heuristic fault tree reduction method is proposed here. In this method, three supplements are provided in order to make the reviewed Boolean rules of simplification more suitable for paging stored fault trees.

(1) TRANSFER gate is treated as the same type of basic event while implementing those rules.
(2) Simplify every page from bottom to top to its minimal form first, and the last page that is the whole fault tree is simplified last.
(3) A new rule is added after those four rules, and iterations of these five rules will continue until the fault tree structure converges. This important rule is to rename each gate according to the sub-tree structure where its top event is this gate, then to restore the whole fault tree as pages according to the rules mentioned above. That is, all of the sub-trees with a same certain structure will be represented by a TRANSFER gate with a unique name according to its structure characteristics. And this TRANSFER gate will be connected to a single sub-tree page which stores the certain fault tree structure.

Figs. 3 and 4 are examples which show the importance of the three supplements. The fault tree page linked by TRANSFER gate A can be simplified to its minimal form by supplement two, and TRANSFER gate A can be looked as a basic event based on supplement one, then G4 will disappear by the elimination rule. After all rules of simplification, both G3 and G5 have the same structure representing (A + B) which is stored as a page after renaming them as the same gate from supplement three.

A fault tree module is a sub-tree which is completely independent of the rest of the tree, which means one event cannot appear both inside and outside of the module. In the practice of the large fault tree analysis for many nuclear power plants, it is found that the simplification had a substantial effect on the modularization which significantly influences the computational speed of the large fault tree analysis. Therefore, a heuristic reconstruction method is proposed here in order to obtain more modules. In this method, pages will be broken down in the modularization process of the fault tree simplified based on paging stored structures.

Consider the fault tree shown in Fig. 5 for example, which has three pages. According to the simplification rules described above,

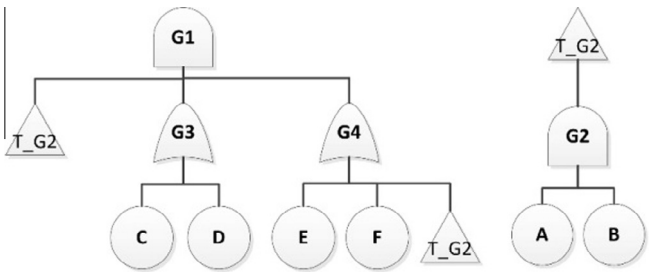

Fig. 2. Fault tree structure with paging storage.