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.

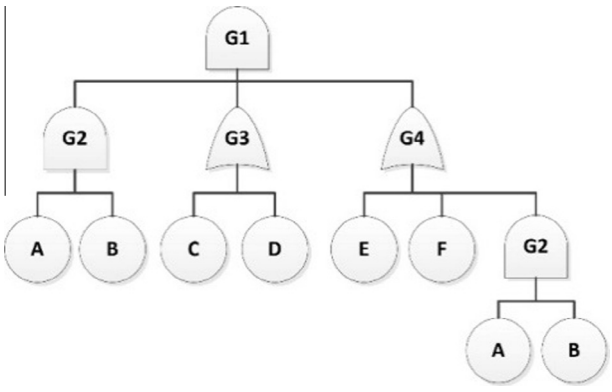

Fig. 1. Fault tree structure without paging storage.

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,

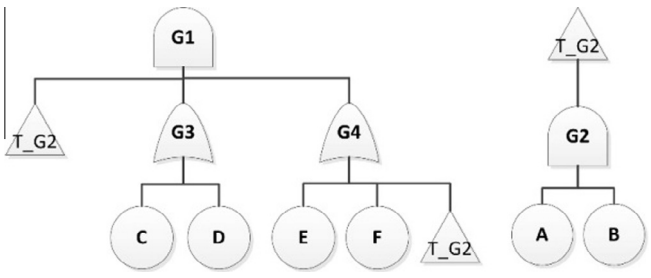

Fig. 2. Fault tree structure with paging storage.

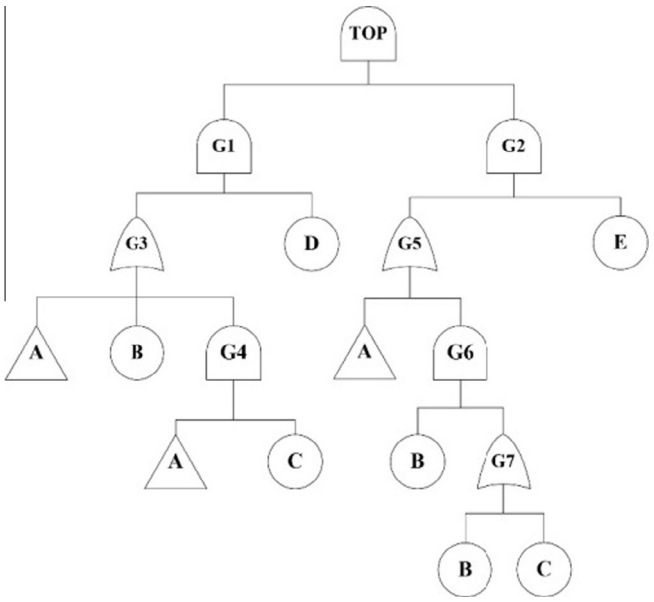

Fig. 3. Fault tree structure before simplification with three supplements.

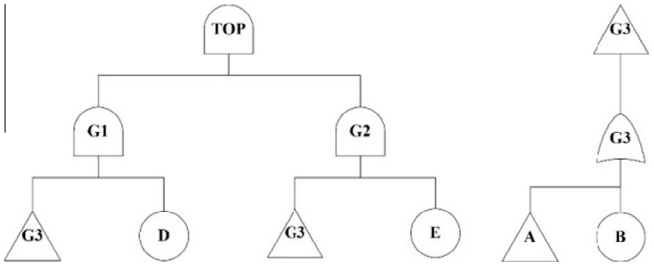

Fig. 4. Fault tree structure after simplification with three supplements.

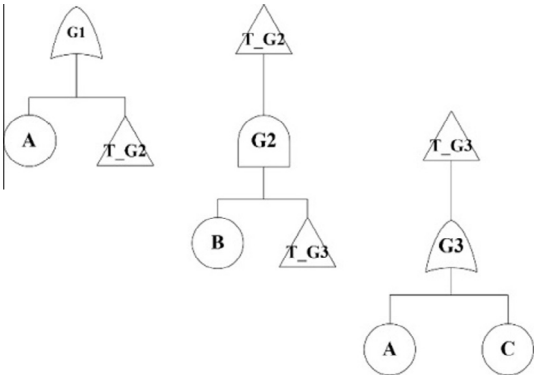

Fig. 5. Fault tree with paging storage after simplification.

this fault tree cannot be simplified further; therefore it can only be treated as a single module because the event A repeats. If this fault tree is not stored in pages, it would be simplified as depicted in Fig. 6, which can be divided as two smaller modules. These pages will be broken down which allows the fault tree to be reconstructed. Therefore, more modules are developed when detecting such cases which will hinder further simplification.

3. Results and discussion

The proposed method has been implemented in reliability and probabilistic safety assessment program RiskA (Wu et al., 2007, 2011; Wu and FDS Team, 2015; Wang et al., 2015; Yin et al., 2014). RiskA has been successfully applied in FDS series Fusion Power Plants (Wu and FDS Team, 2006, 2008; Hu and Wu, 2006), the International Thermonuclear Experimental Reactor (ITER) Test Blank Module (Wu and FDS Team, 2007), Fusion-Driven Hybrid System (Wu et al., 2006, 2011) and China LEAd-based Reactor CLEAR-I (Wu et al., 2014).

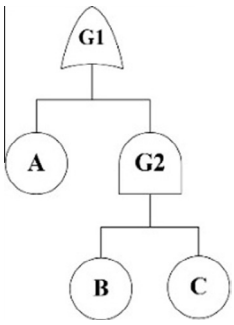

Fig. 6. Fault tree without paging storage after simplification.

Table 1
Fault tree node counts in different data structures.

No.	Node counts		Percentage (%)
	Directly stored	Paging stored	
1	377,945	1655	0.44
2	51,280	1122	2.19
3	100,437	2995	2.98
4	2,737,058	4270	0.16
5	2,686,621	4465	0.17
6	287,584	3053	1.06
7	287,584	3053	1.06
8	133,363	2756	2.07
9	165,226	3141	1.90
10	98,416	2248	2.28
11	10,712	939	8.77

Table 2
Description of 8 test cases.

No.	Node Counts		Truncation		Cutset number
	Event	Gate	Rank	Probability	
1	1616	1030	6	10^{-10}	2541
2	1616	1030	8	10^{-10}	2540
3	1616	1030	6	10^{-12}	8668
4	1616	1030	8	10^{-12}	8868
5	6963	5723	6	10^{-10}	22,827
6	6963	5723	8	10^{-10}	33,723
7	6963	5723	6	10^{-12}	407,612
8	6963	5723	8	10^{-12}	899,788

After the implemented of this method, RiskA is capable of generating minimal cutsets for fault trees containing more than 10,000 gates and basic events faster. Fault trees for Third Qinshan Nuclear Power Plant were used to test the proposed method, and the testing environment was Intel Core(TM) 2 Duo CPU E7200 @ 2.53 GHz, 2.00 GB Memory.

The first test was designed to prove the effectiveness of paging storage. Fault tree node counts of different data structures are described in Table 1 for 11 cases. From the "percentage" column, it shows that the node counts of the paging stored fault trees will be only 0.16–8.77% of the fault trees without paging storage.

The second test was designed to verify the efficiency of the simplification method for the paging stored fault trees. The description

Table 3
Test results.

No.	Before improving		After improving		Memory elimination	Speed comparison
	Memory (MB)	Time (ms)	Memory (MB)	Time (ms)		
1	13	5703	8	657	38.5%	8.7
2	13	5687	8	641	38.5%	8.9
3	21	7578	11	2500	47.6%	3.0
4	21	7640	11	2625	47.6%	2.9
5	134	269,422	60	24,703	55.2%	10.9
6	217	270,015	66	25,297	69.6%	10.7
7	546	391,016	234	146,875	57.1%	2.7
8	1487	470,563	486	226,110	67.3%	2.1

Table 4
Cases and test results.

No.	Node counts		Module counts before improving	Module counts after improving	Improvement (%)
	Event	Gate			
1	18,830	6822	784	1206	53.8
2	12,800	6419	478	865	80.9
3	16,905	6484	612	983	60.6

of 8 test cases is shown in Table 2, and results are presented in Table 3. From the results, it shows that the heuristic fault tree reduction method based on fault tree paging storage can eliminate 38.5–69.6% computer resources requirements and improve the computation speed by up to a factor of 2.1–10.9.

The third test was designed to show the improvement made by the heuristic reconstruction method. From the results shown in Table 4, it concludes that the heuristic reconstruction method is efficient in increasing the number of modules for the paging stored fault trees. The number of modules has been raised by 53.8–80.9%, which significantly enhances the computation speed of the fault tree analysis by reducing the occurrences of the extremely large modules due to the effect of modularization.

The proposed pre-processing method is suitable for all kinds of methods of FTA. However, it will be more helpful for the methods that is more sensitive to fault tree structure. For example, the methods based on Shannon decomposition such as Binary Decision Diagram method, the methods based on matrix such as Petri net method, and the traditional methods such as Fussell–Vesely method. The proposed method will be less helpful for the methods based on Boolean algebra and Set theory such as Branch-and-deduce method.

4. Conclusion

The fault tree pre-processing methods used in RiskA are presented, which include the fault tree paging storage, the fault tree simplification and the fault tree modularization. Moreover, three simplification supplement rules based on the paging storage were provided, and a method of breaking down a page for more efficient modularization is proposed. Test results shows that these methods can significantly reduce computer resources needs (RAM) and improve computation speed when solving large fault trees as typical for the PRA models of nuclear power plants.

Acknowledgments

This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (No. XDA03040000), the National Magnetic Confinement Fusion Science Program of China (Nos. 2014GB112000 and 2015GB116000), the Informatizational Special Projects of Chinese Academy of Sciences (No. XXH12504-1-09), Foundation of President of Hefei Institutes of Physical Science (No. YZJJ201327) and the Major / Innovative Program of Development Foundation of Hefei Center for Physical Science and Technology (No. 2014FXCX004). We further thank other members of FDS Team for their great help in this research.

References

Bengiamin, N.N., Bowen, B.A., Schenk, K.F., 1976. Efficient algorithm for reducing complexity of computation in fault tree analysis. IEEE Trans. Nucl. Sci. 23 (5), 1442–1446.
Berg, U., 1990. Reltree and risk spectrum – experience from and development of PSA software for PCS. Reliab. Eng. Syst. Safe. 30, 323–338.
Camarinopoulos, L., Yllera, J., 1986. Advanced concepts in fault tree modularization. Nucl. Eng. Des. 91 (1), 79–91.
Chatterjee, P., 1975. Modularization of fault trees: a method to reduce the cost of analysis. Reliab. Fault Tree Anal., 101–137
Dutuit, Y., Rauzy, A., 1996. A linear-time algorithm to find modules of fault trees. IEEE Trans. Reliab. 45, 422–425.
Hu, L., Wu, Y., 2006. Probabilistic safety assessment of the dual-cooled waste transmutation blanket for the FDS-I. Fusion Eng. Des. 81, 1403–1407.
Jim, Koren, John, Gaertner, Jeff, Riley, 1986. CAFTA plus: a comprehensive fault tree development work station. In: Proceedings – Annual Reliability and Maintainability Symposium. IEEE, Las Vegas, NV, USA, pp. 6–10.
Kohda, T., Henley, E.J., Inoue, K., 1989. Finding modules in fault-trees. IEEE Trans. Reliab. 38 (2), 165–176.
Niemela, I., 1994. On simplification of large fault-trees. Reliab. Eng. Syst. Safe. 44 (2), 135–138.
Platz, O., Olsen, J.V., 1976. FAUNET: a program package for evaluation of fault trees and network. Res. Establish. RIS Rep., 97–103
Rauzy, A.B., 2012. Anatomy of an efficient fault tree assessment engine. In: 11th Probabilistic Safety Assessment and Management, Helsinki. Probablistic Safety Assessment and Management (IAPSAM), Finland, pp. 3333–3343.
Russell, K.D., Rasmuson, D.M., 1993. Fault tree reduction and quantification – an overview of IRRAS algorithms. Reliab. Eng. Syst. Safe. 40 (2), 149–164.
Sun, H., Andrews, J.D., 2004. Identification of independent modules in fault trees which contain dependent basic events. Reliab. Eng. Syst. Safe. 86, 285–296.
Wang, J., Chen, S.Q., Wang, F., et al., 2015. A new decomposition algorithm for complex voting gates processing in fault tree analysis. J. Risk Reliab.
Wilson, J.M., 1985. Modularizing and minimizing fault-trees. IEEE Trans. Reliab. 34 (4), 320–322.
Wu, Y., Team, F.D.S., 2006. Conceptual design activities of FDS series fusion power plants in China. Fusion Eng. Des. 81 (23), 2713–2718.
Wu, Y., Team, F.D.S., 2007. Conceptual design and testing strategy of a dual functional lithium-lead test blanket module in ITER and EAST. Nucl. Fusion 47 (11), 1533–1539.
Wu, Y., Team, F.D.S., 2008. Conceptual design of the China fusion power plant FDS-II. Fusion Eng. Des. 83 (10), 1683–1689.
Wu, Y., Team, F.D.S., 2015. Development of reliability and probabilistic safety assessment program risk A. Ann. Nucl. Energy 83, 316–321.
Wu, Y., Zheng, S., Zhu, X., et al., 2006. Conceptual design of the fusion-driven subcritical system FDS-I. Fusion Eng. Des. 81 (8), 1305–1311.
Wu, Y., Liu, P., Hu, L., et al., 2007. Development of an integrated probabilistic safety assessment program. Chin. J. Nucl. Sci. Eng. 27 (3), 270–276.
Wu, Y., Hu, L., Li, Y., et al., 2011. Development of third qinshan nuclear power plant risk monitor. Chin. J. Nucl. Sci. Eng. 31 (1), 68–75.
Wu, Y., Jiang, J., Wang, M., 2011. A fusion-driven subcritical system concept based on viable technologies. Nucl. Fusion 51 (10), 532–542.
Wu, Y., Bai, Y., Song, Y., et al., 2014. Conceptual design of china lead-based research reactor CLEAR-I. Nucl. Sci. Eng. 34 (2), 201–208.
Yin, Y., Wang, J., Chen, S., et al., 2014. Calculation engine comparison of probabilistic safety analysis program risk A and risk spectrum. Nucl. Sci. Eng. 34 (2), 285–288.