Nuclear Engineering and Technology 56 (2024) 2092–2098

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Zero-suppressed ternary decision diagram algorithm for solving noncoherent fault trees in probabilistic safety assessment of nuclear power plants

Woo Sik Jung

Sejong University, 209 Neungdong-Ro, Gwangjin-Gu, Seoul, 143-747, Republic of Korea

ARTICLE INFO

Keywords:
Zero-suppressed ternary decision diagram
Minimal cut sets
Prime implicants

ABSTRACT

Probabilistic safety assessment (PSA) plays a critical role in ensuring the safe operation of nuclear power plants. In PSA, event trees are developed to identify accident sequences that could lead to core damage. These event trees are then transformed into a core-damage fault tree, wherein the accident sequences are represented by usual and complemented logic gates representing failed and successful operations of safety systems, respectively. The core damage frequency (CDF) is estimated by calculating the minimal cut sets (MCSs) of the core-damage fault tree.

Delete-term approximation (DTA) is commonly employed to approximately solve MCSs representing accident sequence logics from noncoherent core-damage fault trees. However, DTA can lead to an overestimation of CDF, particularly when fault trees contain many nonrare events. To address this issue, the present study introduces a new zero-suppressed ternary decision diagram (ZTDD) algorithm that averts the CDF overestimation caused by DTA.

This ZTDD algorithm can optionally calculate MCSs with DTA or prime implicants (PIs) without any approximation from the core-damage fault tree. By calculating PIs, accurate CDF can be calculated.

The present study provides a comprehensive explanation of the ZTDD structure, formula of the ZTDD algorithm, ZTDD minimization, probability calculation from ZTDD, strength of the ZTDD algorithm, and ZTDD application results. Results reveal that the ZTDD algorithm is a powerful tool that can quickly and accurately calculate CDF and drastically improve the safety of nuclear power plants.

1. Introduction

1.1. Accident sequence logic

Coherent fault trees comprise basic events related to component failures and logic gates. By contrast, noncoherent fault trees comprise complemented basic events or complemented logic gates. Minimal cut sets (MCSs) are the minimal combinations of failures causing the top event of a coherent fault tree. Further, prime implicants (PIs) are the minimal combinations of failures and successes causing the top event of a noncoherent fault tree. In the probabilistic safety assessment (PSA) of nuclear power plants, both MCSs and PIs are usually denoted as MCSs.

The PSA of nuclear power plants is performed using event and fault trees. Each accident sequence in event trees comprises a logical combination of usual and complemented fault trees representing safety system failures and successes, respectively. Some of them lead to core damage in nuclear power plants. Because the general Boolean expression for an accident scenario $G_3G_4\ldots/G_7/G_8\ldots$ is identical to $G_3G_4\ldots/(G_7 + G_8 + \ldots)$, it can be expressed as G_1/G_2, where $G_1 = G_3G_4\ldots$ and $G_2 = G_7 + G_8 + \ldots$. Each accident scenario G_1/G_2 is inherently a noncoherent fault tree; thus, the expression should be solved using the appropriate algorithm. This study explains various ways to solve the accident sequence logic given by Eq. (1).

$$\begin{aligned} Top &= G_1/G_2 \\ G_1 &= bG_3 \\ G_2 &= bG_4 \\ G_3 &= a + c + e \\ G_4 &= c + d \end{aligned} \tag{1}$$

1.2. Approximate solutions by DTA

In PSA, the MCSs of G_1 and G_2 are separately generated using

E-mail address: woosjung@sejong.ac.kr.

https://doi.org/10.1016/j.net.2024.01.017
Received 4 August 2023; Received in revised form 8 January 2024; Accepted 13 January 2024
Available online 17 January 2024

Table 1
Delete-term approximation.

MCS of G_1	True events	G_1	G_2	G_1/G_2	MCS of G_1/G_2
ab	a and b	True	Indefinite	Indefinite	Yes
bc	b and c	True	True[a]	False	No
be	b and e	True	Indefinite	Indefinite	Yes

[a] G_2 is true when b and c are true.

traditional Boolean algebra [1] or the zero-suppressed binary decision diagram (ZBDD) algorithm [2–4]. Then, the approximate MCSs of G_1/G_2 are generated using delete-term approximation (DTA) [4], which compares the MCSs of G_1 and G_2 and deletes some nonlogical MCSs of G_1. Then, the core damage frequency (CDF) is calculated from the approximate MCSs. Here, MCSs and factorized MCSs are calculated using traditional Boolean algebra and the ZBDD algorithm, respectively.

The DTA procedure for calculating the approximate MCSs of G_1/G_2 in Eq. (1) is given by Eq. (2) and presented in Table 1. Among the MCSs of G_1, i.e., ab, bc, and be, bc is deleted since it results in false G_1/G_2, and ab and be are captured as final approximate MCSs of G_1/G_2 since they do not result in false G_1/G_2. In other words, DTA removes the subset MCSs in G_1 that have a superset MCS in G_2. The DTA is always performed one time with the MCSs of G_1 and G_2 at the top event. In other words, the DTA is not performed at the child gates of the top event.

$$\begin{aligned} &G_1 = ab + bc + be \\ &G_2 = bc + bd \\ &G_1/G_2 \cong Delterm(G_1, G_2) = ab + be \end{aligned} \tag{2}$$

1.3. Accurate solutions and probabilities

First, a solution of a binary decision diagram (BDD) [5–15] for G_1/G_2 can be directly calculated from the fault tree in Eq. (1) with the BDD algorithm [7–9], and the accurate probability $p(G1/G2)$ is calculated with this BDD. However, the BDD calculation frequently fails for large fault trees in PSA. The structure and algorithm of ternary decision diagram (TDD) [17–19] are variation of the BDD structure and algorithm (see Section 1.5).

Second, Boolean solutions of G_1/G_2 are generated as Eq. (3) from the fault tree in Eq. (1) without any approximation by the traditional Boolean algebra. However, the Boolean complement of G_2 frequently fails depending on the size of the fault tree for G_2.

$$\begin{aligned} &G_1 = ab + bc + be \\ &G_2 = bc + bd \\ &/G_2 = /(bc + bd) = (/b + /c)(/b + /d) = /b + /c/d \\ &G_1/G_2 = (ab + bc + be)(/b + /c/d) = ab/c/d + b/c/de \end{aligned} \tag{3}$$

Then, the solutions of G_1/G_2 in Eq. (3) are converted into a BDD [5–15] or sum-of-disjoint products (SDPs) [16], and the accurate probability $p(G_1/G_2)$ is calculated with the BDD or SDPs. Since the Boolean solutions in Eq. (3), $ab/c/d$ and $b/c/de$, are not disjoint, they are converted into SDPs of $ab/c/d$ and $/ab/c/de$ using Eq. (4). The probability $p(G1/G2)$ is calculated by simply adding the probabilities of SDPs.

$$\begin{aligned} &G_1/G_2 = ab/c/d + /(ab/c/d)b/c/de \\ &= ab/c/d + (/a + a/b + abc + ab/cd)b/c/de \\ &= ab/c/d + /ab/c/de \\ &p(G1/G2) = p(ab/c/d) + p(/ab/c/de) \end{aligned} \tag{4}$$

1.4. Comparison of approximate and accurate solutions

As listed in Table 2, three probabilities of rare event approximation (REA) probabilities, min-cut-upper bound (MCUB) probabilities, and SDP probabilities are calculated from $ab + be$ and $ab/c/d + b/c/de$ in Eqs. (2) and (3). As summarized by Eq. (5), Table 2 clearly shows that the three probabilities of G_1/G_2 in Eq. (2) are drastically overestimated when the accident sequence logic has nonrare events, such as seismic events. Thus, CDF can be extremely overestimated by DTA when fault trees have many nonrare events.

$$\begin{aligned} &p(ab + de) \cong p(ab/c/d + b/c/de) \text{ for rare events in internal PSA} \\ &p(ab + de) \gg p(ab/c/d + b/c/de) \text{ for nonrare events in seismic PSA} \end{aligned} \tag{5}$$

1.5. ZBDD, BDD, and TDD algorithms

Arbitrary Boolean equations can be encoded into ZBDD [2–4], BDD [5–15], or TDD [17–19] in Table 3. When a fault tree is solved in a bottom-up way, two ZBDDs, BDDs, and TDDs are combined with ZBDD [3,4], BDD [7,8], and TDD [17–19] algorithms, respectively. These ZBDD and BDD structures and algorithms provide efficient calculation of fault trees. As presented in Table 3, The TDD structure is a simple variation of BDD.

Bryant popularized the use of BDD by developing the BDD algorithm to efficiently construct and manipulate BDDs [7,8]. The BDD algorithm has been employed for reliability analysis [9], and the use of BDDs to solve large fault trees and importance measures has been investigated [10–14]. The BDD algorithm calculates an exact top event probability since it does not employ any approximations such as DTA. However, the BDD algorithm frequently fails to solve large fault trees. BDD truncation

Table 2
Comparison of $p(ab + de)$ and $p(ab/c/d + b/c/de)$.

Event probability[a]	Solution	REA probability	MCUB probability	SDP probability	Overestimation
1.0E-03	$ab + be$	2.00E-06[b]	2.00E-06[d]	2.00E-06[f]	0.2 %[h]
	$ab/c/d + b/c/de$	2.00E-06[c]	2.00E-06[e]	2.00E-06[g]	
1.0E-01	$ab + be$	2.00E-02	1.99E-02	1.90E-02	23.5 %
	$ab/c/d + b/c/de$	1.62E-02	1.61E-02	1.54E-02	
5.0E-01	$ab + be$	5.00E-01	4.38E-01	3.75E-01	300.0 %
	$ab/c/d + b/c/de$	1.25E-01	1.21E-01	9.38E-02	
9.0E-01	$ab + be$	1.62E-00	9.64E-01	8.91E-01	9900.0 %
	$ab/c/d + b/c/de$	1.62E-02	1.61E-02	8.91E-03	

[a] $p(a) = \ldots = p(e)$.
[b] $p(ab) + p(be)$.
[c] $p(ab/c/d) + p(b/c/de)$.
[d] $1 - (1 - p(ab))(1 - p(be))$.
[e] $1 - (1 - p(ab/c/d))(1 - p(b/c/de))$.
[f] $p(ab) + p(/abe)$ by $ab + be = ab + /(ab)be = ab + (/a + a/b)be = ab + /abe$.
[g] $p(ab/c/d) + p(/ab/c/de)$ by Eq. (4).
[h] $((f) - (g))/(g) * 100$ %.

Table 3
Comparison of BDD, TDD, and ZBDD algorithms.

Algorithm	Encoding	Solutions
ZBDD[a]	$< x, L, R >_{ZBDD} = xL + R$	Factorized MCSs
BDD[b]	$< x, L, R >_{BDD} = xL + /xR$	Disjoint solutions
TDD[c]	$< x, L, R, LR >_{TDD} = xL + /xR + LR$	Variation of BDD

[a] For mainly solving coherent fault trees.
[b] For mainly solving noncoherent fault trees.
[c] $< x, L, R, LR >_{TDD} = xL + /xR + LR = x(L + LR) + /x(R + LR) = xL + /xR = < x, L, R >_{BDD}$.

during the solving of a fault tree was impossible before the development of the BDD truncation algorithm [15].

1.6. Objectives and structure of the paper

The complemented logic gate of $/G_2$ can be solved using the Boolean complement of ZBDD in Eq. (6), and two ZBDDs for G_1 and $/\ G_2$ can be combined to generate the exact solutions of G_1/G_2. However, it is not practically used since the simultaneous encoding of x and $/\ x$ in ZBDD is very complex and drastically increases the computational burden depending on the number of complemented events. Thus, a noncoherent fault tree is solved using the ZBDD algorithm with DTA instead of the exact calculation of G_1/G_2. Owing to the use of DTA, the seismic CDF is drastically overestimated since the seismic PSA model has many nonrare events as explained in Section 1.4.

$$\begin{aligned} &/ < x, L, R >_{ZBDD} =< x, /L/R, < /x, /R, 0 > >_{ZBDD} \\ &/(xL + R) = /(xL)/R = (/x + x/L)/R = /x/R + x/L/R \end{aligned} \tag{6}$$

Although the ZBDD algorithm is a successful replacement of traditional Boolean algebra, generating the accurate solution of $G_1/\ G_2$ is a practically impossible task. This was a motivation for developing the zero-suppressed ternary decision diagram (ZTDD) algorithm [20]. The ZTDD algorithm was developed by the author of this paper, and it was very briefly introduced in Ref. 20.

This study introduces the ZTDD algorithm and its features that were developed for efficiently solving noncoherent fault trees in PSA. By using the ZTDD algorithm, (1) the approximate solution is calculated by DTA or (2) the accurate solution can be calculated without DTA.

The remainder of this paper is structured as follows. The ZTDD structure, formula of the ZTDD algorithm, deletion of the nonminimal solutions in ZTDD, DTA with the ZTDD algorithm, and probability calculation of ZTDD are explained in Section 2. The strength of the ZTDD Algorithm is summarized in Section 3. Furthermore, the results of applications to a sample and PSA fault trees that demonstrate the efficiency of the ZTDD algorithm are summarized in Sections 4 and 5. Finally, conclusions are provided in Section 6.

2. ZTDD structure and algorithm

In this section, the ZTDD algorithm is introduced. The ZTDD algorithm has two major strengths: (1) It can calculate a solution of $G_1/\ G_2$ without DTA (see Sections 2.2 and 2.3). (2) It can calculate approximate solutions of G_1/G_2 with DTA (see Sections 2.2 and 2.5).

2.1. ZTDD structure

ZTDD is newly defined for encoding the factorized MCSs or PIs that have complemented basic events. ZTDD has a Boolean structure that comprises recursively connected if-then-else connectives (ITEs) that have three terms of L, R, and N as given by Eq. (7). ZTDD encodes the Boolean equation $xL + /xR + N$ into three Boolean equations as xL, $/\ xR$, and N, where L, R, and N are child ZTDDs. The ZTDD can be interpreted as a factorized form of MCSs or PIs.

$$xL + /xR + N =< x, L, R, N > \tag{7}$$

The ZTDD in Eq. (7) can be encoded into BDD through Eq. (8) or converted into two connected ZBDDs through Eq. (9). Clearly, the ZTDD in Eq. (7) is much more intuitive and simpler than the BDD and ZBDD in Eqs. (8) and (9).

$$xL + /xR + N = xL + /xR + (x + /\ x)N = < x, L + N, R + N >_{BDD} \tag{8}$$

$$xL + /xR + N =< x, L, < /x, R, N > >_{ZBDD} \tag{9}$$

2.2. ZTDD algorithm

To solve a fault tree in a bottom-up way, two ZTDDs need to be combined in a logical manner. In this study, a set of ZTDD formulae are developed for combining two ZTDDs as given by Eq. (10). If x and y are two variables with a given variable ordering $x < y$, x is located at a higher position in ZTDD than y. Thereafter, the ZTDD combining operation with $G(x) =< x, L_1, R_1, N_1 >$ and $H(y) =< y, L_2, R_2, N_2 >$ is recursively performed from top to bottom ITEs following Eq. (10). Thus, a coherent or noncoherent fault tree can be solved in a bottom-up way using Eq. (10).

$$\begin{aligned} &G(x)\cdot H(x) =< x, (L_1L_2 + L_1N_2 + N_1L_2), (R_1R_2 + R_1N_2 + N_1R_2), N_1N_2 > \\ &G(x) + H(x) =< x, (L_1 + L_2), (R_1 + R_2), (N_1 + N_2) > \\ &G(x)\cdot H(y) =< x, L_1H, R_1H, N_1H > \\ &G(x) + H(y) =< x, L_1, R_1, (N_1 + H) > \end{aligned} \tag{10}$$

2.3. ZTDD for Boolean complement

Eq. (11) displays the Boolean complement of ZTDD. Here, the Boolean complements $/(xL) = /x + x/L$ and $/(/xR) = x + /x/R$ are applied instead of $/(xL) = /x + /L$ and $/(/xR) = x + /R$ to maintain disjoint solutions as much as possible. To calculate the accurate solutions of G_1/G_2, the ZTDD of G_2 is complemented into $/G_2$ by Eq. (11) and the two ZTDDs of G_1 and $/G_2$ are combined using Eq. (10).

$$\begin{aligned} &/ < x, L, R, N >=< x, /L/N, /R/N, 0 > \\ &/(xL + /xR + N) = (/x + x/L)(x + /x/R)/N = x/L/N + /x/R/N \end{aligned} \tag{11}$$

2.4. ZTDD minimization

When a fault tree is solved in a bottom-up way using Eq. (10), nonminimal solutions (subsets) are introduced in ZTDD, and they need to be deleted. These nonminimal solutions exist in the L and R of $< \alpha, L, R, N >$ since ZTDD is $\alpha L + /\alpha R + N$. The subsets in L and R are deleted if their minimal solutions (supersets) exist in N, through the *Subsume*(L, N) and *Subsume*(R, N) operations in Eq. (12).

$$\begin{aligned} &Subsume(G,H) = G\backslash H = \begin{cases} G\backslash N_2 & ,x > y \\ < x, L_1\backslash H, R_1\backslash H, N_1\backslash H > & ,x < y \\ < x, L_1\backslash (L_2 \text{ or } N_2), R_1\backslash (R_2 \text{ or } N_2), N_1\backslash N_2 > & ,x = y \end{cases} \\ &G(x) =< x, L_1, R_1, N_1 >= xL_1 + /xR_1 + N_1 \\ &H(y) =< y, L_2, R_2, N_2 >= yL_2 + /yR_2 + N_2 \end{aligned} \tag{12}$$

The term $L_1\backslash(L_2 \text{ or } N_2)$ denotes that each solution in L_1 is tested and

deleted if L_2 or N_2 has a superset.

2.5. ZTDD for DTA

To calculate the approximate solutions of G_1/G_2, DTA is employed. It is accomplished through the subsuming operation given by Eq. (13).

$$G_1 / G_2 \approx Delterm(G_1, G_2) = Subsume(G_1, G_2) \tag{13}$$

2.6. ZTDD probability calculation

First, the sum of PI probabilities is calculated by recursively calculating the probability given by Eq. (14) from the bottom to top of the ZTDD.

$$p(f) = p_x \times p(L) + (1 - p_x) \times p(R) + p(N) \tag{14}$$

Second, the MCUB probability of PIs can be optionally calculated by navigating all minimal solutions in the ZTDD. Third, the exact probability can be calculated from BDD by converting the ZTDD into BDD if necessary (see Eq. (8)).

3. Strength of the ZTDD algorithm

Both ZTDD and ZBDD algorithms can generate MCSs from coherent and noncoherent fault trees. However, the ZBDD algorithm aims to solve coherent fault trees. When a ZBDD algorithm solves a noncoherent fault tree, complemented basic events should be converted into temporary basic events such as $/x = x_{not}$ as listed in Table 4. Owing to this conversion, the Boolean equations $x/x = 0$ and $x + /x = 1$ cannot be applied during MCS generation of logic gates in a bottom-up way, and they are applied at the final stage after generating MCSs of a top event. For this reason, MCS generation becomes very explosive when a fault tree has many complemented events and logic gates. This is a serious limitation of a ZBDD algorithm.

As listed in Table 5, the Boolean equations of x/x, $x + /x$, and $/(xy)$ can be directly solved with the ZTDD algorithm without the substitution of $/x = x_{not}$ and $/y = y_{not}$. This is a great strength of the ZTDD algorithm over the ZBDD algorithm when a fault tree has many complemented events and logic gates.

4. ZTDD application to sample fault tree

4.1. ZTDD algorithm without DTA

The fault tree in Eq. (1) is solved in a bottom-up way with the ZTDD algorithm in Eq. (10) and Boolean complement in Eq. (11). The resultant ZTDDs for G_1, G_2, and $/G_2$ are depicted in Figs. 1–3. By combining the two ZTDDs of G_1 and $/G_2$, the ZTDD for G_1/G_2 is calculated as shown in Fig. 4.

Table 4
Examples of ZBDD Boolean operations.

Boolean equations	ZBDD Boolean operations (see Appendix A)
x/x	$< x,1,0 >< x_{not},1,0 > = < x,< x_{not},1,0 >,0 >$[a]
$x + /x$	$< x,1,0 > + < x_{not},1,0 > = < x,1,< x_{not},1,0 \gg$[a]
$/(xy)$	Impossible encoding

[a] $x/x = 0$ and $x + /x = 1$ cannot be applied during MCS generation.

Table 5
Examples of ZTDD Boolean operations.

Boolean equations	ZTDD Boolean operations (see Eqs. (10) and (11))
x/x	$< x,1,0,0 >< x,0,1,0 > = < x,0,0,0 > = 0$
$x + /x$	$< x,1,0,0 > + < x,0,1,0 > = < x,1,1,0 > = 1$
$/(xy)$	$/ < x,< y,1,0,0 >,0,0 > = < x,/ < y,1,0,0 >,1,0 >$ $= < x,< y,0,1,0 >,1,0 > = /x + x/y$

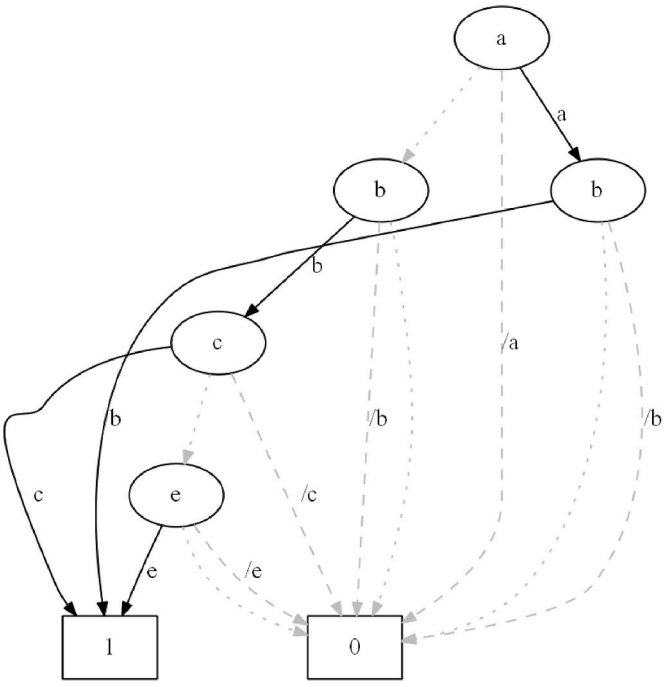

Fig. 1. ZTDD for G_1
$G_1 = < a,< b,1,0,0 >,0,< b,< c,1,0,< e,1,0,0 \gg,0,0 \gg$
Identical to $G_1 = ab + bc + be$ in Eq. (3).

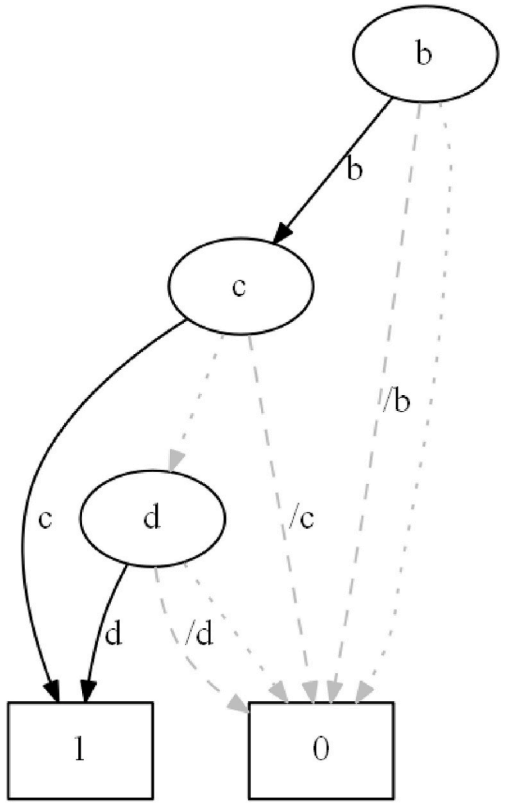

Fig. 2. ZTDD for G_2
$G_2 = < b,< c,1,0,< d,1,0,0 \gg,0,0 >$
Identical to $G_2 = bc + bd$ in Eq. (3).

The ZTDD shown in Fig. 4 can be converted into the BDD shown in Fig. 5. This BDD is identical to the SDPs in Eq. (4). In other words, the

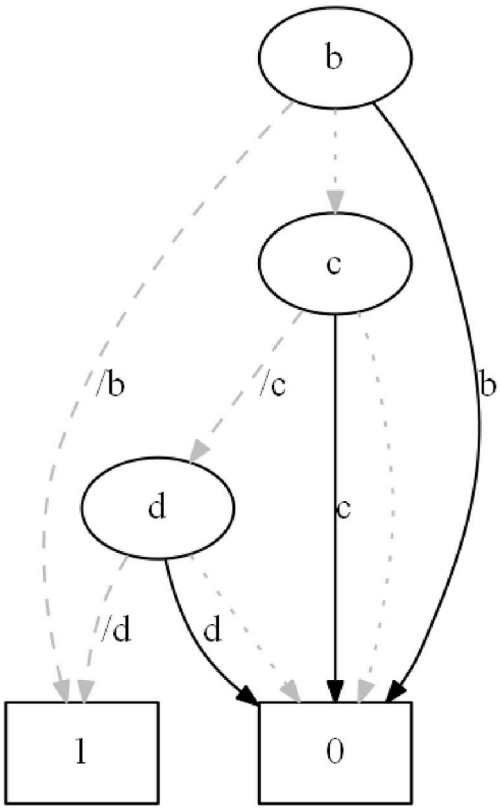

Fig. 3. ZTDD for $/G_2$
$/G_2 = < b, < c, 0, < d, 0, 1, 0 >, 0 >, 1, 0 >$
Identical to $/G_2 = /b + /c/d$ in Eq. (3).

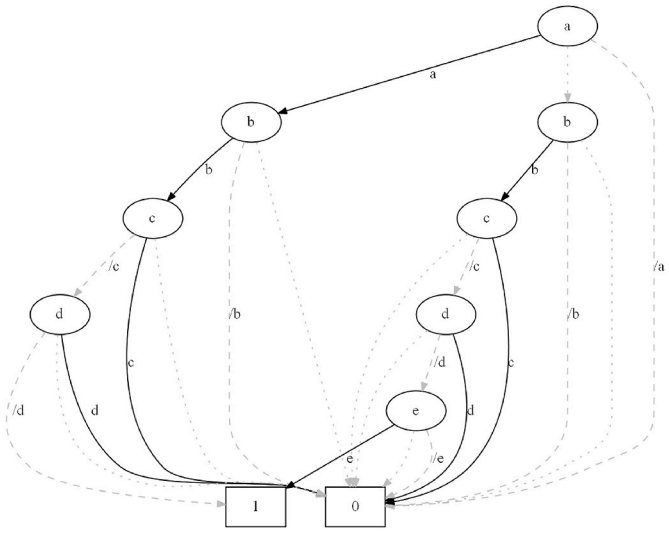

Fig. 4. ZTDD without DTA for G_1/G_2
Identical to $G_1/G_2 = ab/c/d + b/c/de$ in Eq. (3).

BDD that is converted from the ZTDD is identical to the BDD that is directly calculated from G_1/G_2.

4.2. ZTDD algorithm with DTA

The ZTDD that is generated from G_1/G_2 by the ZTDD algorithm with DTA in Eqs. (10) and (13) is depicted in Fig. 6. It can be confirmed that the ZTDD in Fig. 6 is identical to the Boolean equation in Eq. (2).

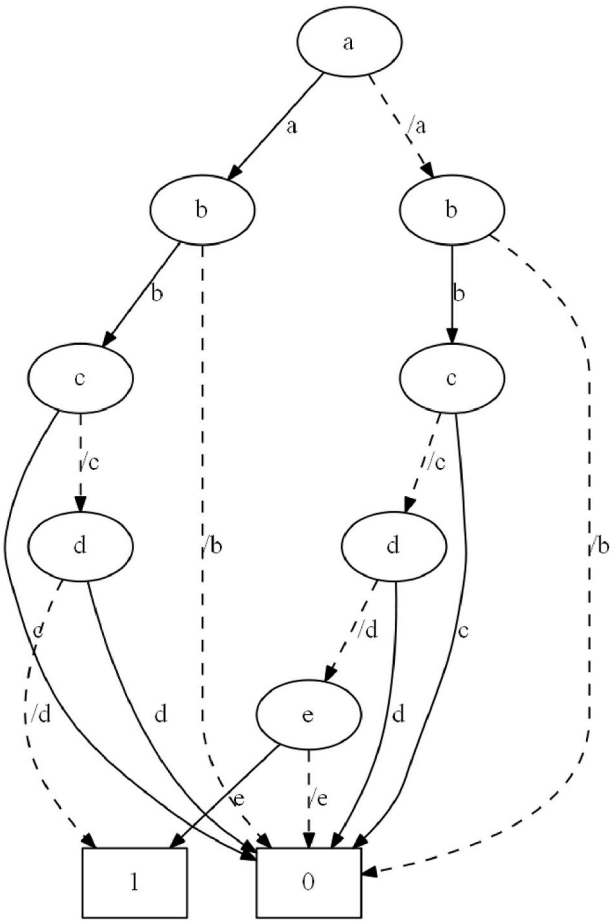

Fig. 5. BDD for G_1/G_2
Identical to $G_1/G_2 = ab/c/d + /ab/c/de$ in Eq. (4).

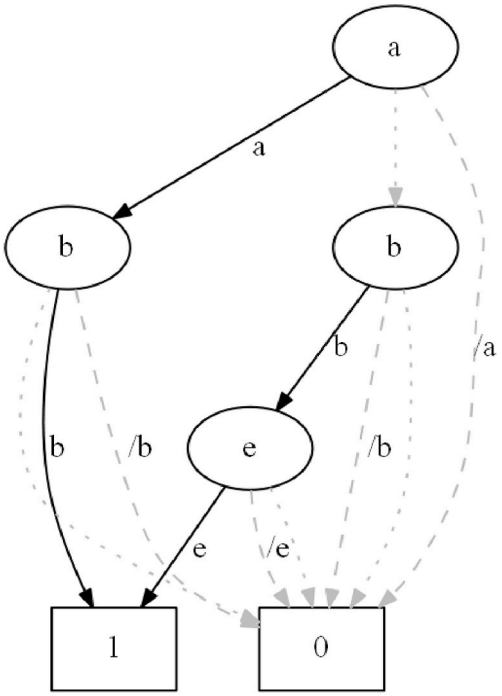

Fig. 6. ZTDD with DTA for $Delterm(G_1, G_2)$
Identical to $Delterm(G_1, G_2) = ab + be$ in Eq. (2).

5. ZTDD application to PSA fault trees

The ZTDD algorithm in Section 2 was implemented in a new tool named ZEBRA (ZTDD Equation Based Risk Analyzer), and its performance was compared with that of the fault tree solver FTREX (Fault Tree Reliability Evaluation eXpert), which employs the ZBDD algorithm [3, 4]. With the large fault trees in Table 6, calculations were performed using FTREX and ZEBRA with DTA by employing the calculation options in Table 7. The calculation results presented in Tables 8–11 reveal that ZEBRA outperforms FTREX in solving large fault trees.

Table 6
PSA fault trees.

Fault trees	Gates	Events	/Gates[a]	/Events[b]	Initiators
Loss of offsite power multi-unit PSA	138,381	24,843	316	9	95
Seismic multi-unit PSA	67,131	19,096	195	0	13
Low power and shutdown PSA	50,220	11,801	4,475	2	45
Internal PSA	8,175	3,982	172	13	17

[a] Complemented gates.
[b] Complemented events.

Table 7
Calculation options.

Calculations	Parallel computing	Local DTA[a]	Local mutually exclusive event deletion[b]
C1			
C2		O	O
C3	O		
C4	O	O	O

[a] Although the DTA is always performed one time with the MCSs of a top event by $G_1/G_2 \cong Delterm(G_1, G_2)$, it can be applied at any child logic gate by $G_1/G_2 \cong Delterm(G_1, G_2)/G_2$.
[b] The complemented gate $/G_2$ of G_1/G_2 for mutually exclusive event deletion can be additionally connected to the child gates of G_1 before solving a fault tree.

Table 8
Calculation of loss of offsite power multi-unit PSA (seconds).

Truncation limit		1.00E-11	1.00E-12	1.00E-13
MCSs		154,937	674,332	2,734,715
FTREX 2.0	C1	28	224	1,972
	C2	34	145	2,250
ZEBRA 1.0	C1	14	45	199
	C2	12	29	88

Table 9
Calculation of seismic multi-unit PSA (seconds).

Truncation limit		1.00E-11	1.00E-12	1.00E-13
MCSs		18,173	85,133	410,261
FTREX 2.0	C1	9	45	303
	C2	5	20	107
ZEBRA 1.0	C1	5	18	112
	C2	4	6	15

Table 10
Calculation of low power and shutdown PSA (seconds).

Truncation limit		1.00E-09	1.00E-10	1.00E-11
MCSs		3,100	25,174	170,963
FTREX 2.0	C1	16	49	630
	C2	15	43	341
ZEBRA 1.0	C1	36	150	581
	C2	24	74	218
	C3	18	84	357
	C4	9	28	86

Table 11
Calculation of internal PSA (seconds).

Truncation limit		1.00E-15	1.00E-16	1.00E-17
MCSs		1,359,385	4,494,871	14,139,375
FTREX 2.0	C1	23	55	127
	C2	33	76	180
ZEBRA 1.0	C1	34	71	173
	C2	22	65	157
	C3	20	46	114
	C4	18	43	105

6. Conclusions

In this paper, a novel algorithm, the ZTDD algorithm, and its features are introduced. The algorithm was developed to avert the CDF overestimation caused by DTA. Further, the solving of noncoherent fault trees with the ZTDD algorithm is elucidated. The use of the ZTDD algorithm for the PSA of nuclear power plants is strongly recommended. The results of this study can be summarized as follows.

(1) The ZTDD algorithm can calculate a solution of G_1/G_2 without DTA (Sections 2.2 and 2.3). If this ZTDD is converted into BDD, it is identical to the BDD that is directly calculated from G_1/G_2. Thus, the extreme overestimation of $p(G_1/G_2)$ can be avoided using the proposed ZTDD algorithm.
(2) The ZTDD algorithm can calculate the approximate solution of G_1/G_2 with DTA (Sections 2.2 and 2.5). This ZTDD is identical to the ZBDD that is directly calculated from G_1/G_2 with DTA. In order to calculate the solution of a noncoherent fault tree using the ZBDD algorithm, $/x$ needs to be replaced with a temporary basic event. Hence, the size of ZBDD exponentially increases when many completed basic events are present (Section 5).
(3) The ZTDD algorithm can solve a noncoherent fault tree comprising many complemented basic events with or without DTA, which is advantageous. Furthermore, the ZTDDs of logic gates maintain a minimal size when a fault tree is solved in a bottom-up way since a ZTDD simultaneously encodes three Boolean logics that have x, $/x$, and nothing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the Korea Foundation Of Nuclear Safety (KOFONS) grant funded by the Nuclear Safety and Security Commission

(NSSC), Republic of Korea (Nos. 2106062-0323-SB110 and 2204017- 0223-SB110).

Appendix A. ZBDD algorithm

The Boolean combination of $G(x) =< x, L_1, R_1 >$ and $H(y) =< y, L_2, R_2 >$ is formulated as Eq. (A.1) [3,4]. Here, $x < y$, $g = G(x)$ and $h = H(y)$. That is, the variable x is located at a higher position in ZBDD than variable y.

$$\begin{aligned} &G(x)\cdot H(x) =< x, (L_1L_2 + L_1R_2 + R_1L_2), R_1R_2 > \\ &G(x) + H(x) =< x, (L_1 + L_2), (R_1 + R_2) > \\ &G(x)\cdot H(y) =< x, L_1h, R_1h > \\ &G(x) + H(y) =< x, L_1, (R_1 + h) > \end{aligned} \tag{A.1}$$

When $ab + abc$ is located in a ZBDD, abc is a subset of ab. Specifically, ab and abc are minimal and nonminimal solutions, respectively. Therefore, abc should be deleted (subsumed). When a fault tree is solved in a bottom-up way with Eq. (A.1), nonminimal solutions in the ZBDD of each logic gate should be deleted.

When a ZBDD has $< \alpha, G(x), H(y) >$, the nonminimal solutions exist in $G(x)$ since it is $\alpha G(x) + H(y)$. The subsuming operation is derived as Eq. (A.2).

$$\text{Subsume}(G,H) = G\backslash H = \begin{cases} G\backslash R_2 & , x > y \\ < x, L_1\backslash H, R_1\backslash H > & , x < y \\ < x, L_1\backslash (L_2 \text{ or } R_2), R_1\backslash R_2 > & , x = y \end{cases} \tag{A.2}$$

$$G(x) =< x, L_1, R_1 >= xL_1 + R_1$$

$$H(y) =< y, L_2, R_2 >= yL_2 + R_2$$

where the term $L_1\backslash(L_2 \text{ or } R_2)$ in the last case indicates that each solution in L_1 is tested and deleted if L_2 or R_2 has its minimal solution.

If a fault tree has a logical combination of G_1/G_2, it is approximated by DTA. Nonminimal solutions in G_1 are deleted if minimal solutions are in G_2. Then, the remaining solutions of G_1 become the final solutions.

$$G_1 \,/\, G_2 \approx \text{Subsume}(G_1, G_2) \tag{A.3}$$

References

[1] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasl, Fault Tree Handbook, U.S. Nuclear Regulatory Commission, NUREG-0492, 1981.

[2] S. Minato, Zero-suppressed BDDs for set manipulation in combinatorial problems, in: Proceedings of the 30th International Conference on Design Automation, 1993, pp. 272–277.

[3] W.S. Jung, S.H. Han, J.J. Ha, A fast BDD algorithm for the reliability analysis of large coherent systems, Reliab. Eng. Syst. Saf. 83 (2004) 369–374.

[4] W.S. Jung, ZBDD algorithm features for an efficient probabilistic safety assessment, Nucl. Eng. Des. 239 (2009) 2085–2092.

[5] C.Y. Lee, Representation of switching circuits by binary-decision programs, Bell System Technical Journal 38 (1959) 985–999.

[6] B. Akers, Binary decision diagrams, IEEE Trans. Comput. C-27 (6) (1978) 509–516.

[7] R. Bryant, Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput. C-35 (8) (1986) 677–691.

[8] R. Bryant, Symbolic Boolean manipulation with ordered binary decision diagrams, ACM Comput. Surv. 24 (1992) 293–318.

[9] A. Rauzy, New algorithms for fault trees analysis, Reliab. Eng. Syst. Saf. 5 (59) (1993) 203–211.

[10] Y. Duituit, A. Rauzy, Efficient algorithms to assess component and gate importance in fault tree analysis, Reliab. Eng. Syst. Saf. 72 (2001) 213–222.

[11] A. Rauzy, BDD for reliability studies, in: K.B. Misra (Ed.), Handbook of Performability Engineering, Elsevier, Amsterdam, The Netherlands, 2008, pp. 381–396.

[12] R. Remenyte-Prescott, J. Andrews, An enhanced component connection method BDD, Reliab. Eng. Syst. Saf. 93 (2008) 1543–1550.

[13] O. Nusbaumer, A. Rauzy, Fault tree linking versus event tree linking approaches - a reasoned comparison, Journal of Risk and Reliability 227 (2013) 315–326.

[14] A. Rauzy, L. Yang, Decision diagram algorithms to extract minimal cutsets of finite degradation models, Information 10 (2019) 368.

[15] W.S. Jung, S.H. Han, J.E. Yang, Fast BDD truncation method for efficient top event probability calculation, Nucl. Eng. Technol. 40 (7) (2008) 571–580.

[16] T. Luo, K.S. Trivedi, An improved algorithm for coherent-system reliability, IEEE Trans. Reliab. 47 (1998) 73–78.

[17] T. Sasao, Ternary Decision Diagrams and Their Applications, Representations of Discrete Functions, 1996, pp. pp269–292 (Chapter 12).

[18] T. Sasao, Arithmetic ternary decision diagrams applications and complexity. Proceedings of the Fourth International Workshop on Applications of the Reed-Muller Expansion in Circuit Design, 1999.

[19] R. Remenyte-Prescott, J. Andrews, Analysis of non-coherent fault trees using ternary decision diagrams, Proc. Inst. Mech. Eng. O J. Risk Reliab. 222 (2) (2008).

[20] W.S. Jung, A new zero-suppressed ternary decision diagram algorithm, in: Probabilistic Safety Assessment and Management (PSAM) Topical, October 23-25, 2023. Virtual meeting.