Journal of the Korean Society of Safety, Vol. 30, No. 3, pp. 100-106, June 2015
 http://dx.doi.org/10.14346/JKOSOS.2015.30.3.100

안전필수 계통의 리스크 평가를 위한 일회 순회 고장수목 모듈 검색 알고리즘

정우식†

세종대학교 원자력공학과

(2015. 3. 11. 접수 / 2015. 4. 7. 수정 / 2015. 5. 18. 채택)

One-time Traversal Algorithm to Search Modules in a Fault Tree for the Risk Analysis of Safety-critical Systems

Woo Sik Jung†

Department of Nuclear Engineering, Sejong University

(Received March 11, 2015 / Revised April 7, 2015 / Accepted May 18, 2015)

Abstract : A module or independent subtree is a part of a fault tree whose child gates or basic events are not repeated in the remaining part of the fault tree. Modules are necessarily employed in order to reduce the computational costs of fault tree quantification. This quantification generates fault tree solutions such as minimal cut sets, minimal path sets, or binary decision diagrams (BDDs), and then, calculates top event probability and importance measures. This paper presents a new linear time algorithm to detect modules of large fault trees. It is shown through benchmark tests that the new method proposed in this study can very quickly detect the modules of a huge fault tree. It is recommended that this method be implemented into fault tree solvers for efficient probabilistic safety assessment (PSA) of nuclear power plants.

Key Words : fault tree analysis, independent subtree, module

1. 서 론

1.1 고장수목과 모듈

고장수목분석(Fault Tree Analysis)은 원자력, 화공, 항공, 우주 시스템과 같은 안전필수 계통(Safety-Critical System)의 리스크 평가(Risk Analysis)에서 광범위하고 성공적으로 사용되어 왔다[1,2]. 원자력발전소 확률론적 안전성평가(Probabilistic Safety Assessment, PSA) 보고서인 WASH-1400 Report[3]이후 PSA에서 고장수목분석은 사건수목분석(Event Tree Analysis)과 함께 원자력발전소의 안전한 설계를 위하여 사용되어 왔다.

안전필수 계통 고장수목분석에 의하여 최소단절집합(Minimal Cut Set, 계통 고장을 유발하는 최소한의 기기고장들의 집합)들을 계산하고, 계통 고장수목 정점사건(Top Event)의 발생 확률을 계산하고, 기기가 계통 고장에 미치는 중요도척도(Importance Measure)를 계산한다. 이 최소단절집합은 최소절단집합으로 불리기도 한다.

안전필수 계통 고장수목의 최소단절집합들 숫자는 거의 무한개에 달하기 때문에, 전용 소프트웨어를 사용하여 최소단절집합을 생성한다. 이 때 최소단절집합 생성을 위한 계산양이 매우 많아 오랜 계산을 요구한다. 이 고장수목분석 계산의 어려움을 감소시키기 위해서 (1) 최소단절집합 절삭(Truncation), (2) 고장수목의 모듈(Module) 사용, (3) 고장수목의 재구조(Restructuring)를 사용한다.

첫 번째, 최소단절집합의 생성시 낮은 발생확률을 갖는 최소단절집합을 버리는 최소단절집합 절삭이 자주 사용된다. 그러나, 이 최소단절집합 절삭을 사용하여도, 정점사건 확률과 기기 중요도척도를 정확히 계산하기 위한 충분한 숫자의 최소단절집합들의 생성이 불가능할 때가 많다. 원자력 발전소의 확률론적안정성 평가에서 충분한 숫자의 최소단절집합들은 1,000부터 1,000,000까지의 범위에 있다.

† Corresponding Author : Woo Sik Jung, Tel : +82-2-3408-4413, E-mail : woosjung@sejong.ac.kr
Department of Nuclear Engineering, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul, Korea

두 번째, 이 고장수목분석 계산의 어려움을 줄이는 방법은 고장수목에서 모듈들을 찾아내고 인공적인 Super Components로 대체하는 것이다. 여기에서 고장수목의 모듈이란 모듈의 하부 Node(게이트와 기본사건)가 고장수목의 다른 부분에 나타나지 않는 독립된 하부수목을 말한다. 고장수목의 독립된 하부수목을 Super Component로 교체하면 최소단절집합의 크기를 대폭 줄일 수 있다. Chatterjee[4]와 Birnbaum[5]는 모듈의 특성들을 연구했고, 고장수목분석에서 모듈들을 사용할 수 있음을 보여주었다. Locks[6]은 모듈의 개념을 Non-coherent 고장수목까지 확장하였다.

세 번째, 고장수목 계산의 어려움을 감소시키는 방법으로 고장수목의 재구조(Fault Tree Restructuring)를 이용한다. 이 고장수목의 재구조는 가능한 한 큰 모듈을 최대한 많이 얻기 위해서 수행된다[7-16]. 고장수목의 재구조는 효율적인 고장수목 정량화를 위해 주어진 고장수목을 동일한 고장수목으로 변환하는 절차이다.

과거에는 독립 하부수목들은 수작업으로 찾았다[17]. 그러나 오늘날에는 고장수목 계산 소프트웨어가 자동으로 독립 하부수목들은 찾아준다[18].

Coherent 혹은 Non-coherent 고장수목의 모듈을 찾기 위해 Dutuit-Rauzy(DR) 알고리즘이 1996년에 제안되었다[19]. 선형시간 알고리즘의 특성 때문에, 이 알고리즘은 빠르게 모듈을 찾을 수 있다고 알려졌다.

원자력발전소의 PSA에서, 다음과 같은 계산 순서로 고장수목분석 계산의 어려움을 크게 감소시키고 있다. (1) 고장수목을 재구조 하고, (2) 모듈을 찾고, (3) 모듈의 최소단절집합을 계산하고, (4) 모듈의 최소단절집합의 확률을 Super Component에 할당하고, (5) 고장수목에서 모듈을 Super Component로 대체하고, (6) 고장수목의 최소단절집합을 생성한다. 모듈 사용은 매우 효율적이어서 고장수목 재구조화와 함께 고장수목 계산의 어려움을 크게 감소시킨다.

1.2 모듈의 정의

고장수목은 기본사건(Basic Event), 게이트(Gate), 정점사건으로 구성된다. Fig. 1의 고장수목은 {*e*5, *e*6, *e*8, *e*9, *e*12, *e*13, *e*14}의 기본사건들과 {*g*0, *g*1, *g*2, *g*3, *g*4, *g*7, *g*10, *g*11}의 게이트들로 이루어지고, 특별히 *g*0를 정점사건이라 부른다. 이 고장수목의 게이트들을 이용하여 기본사건들과 정점사건의 논리적인 상호관계를 표현한다. 기본사건을 Terminal Node로, 게이트를 Intermediate Node로, 정점사건을 Root Node로 명칭하기 때문에, 본 연구에서 Node는 기본사건, 게이트, 정점사건을 통칭한다. 기본사건은 기초적인 기기고장

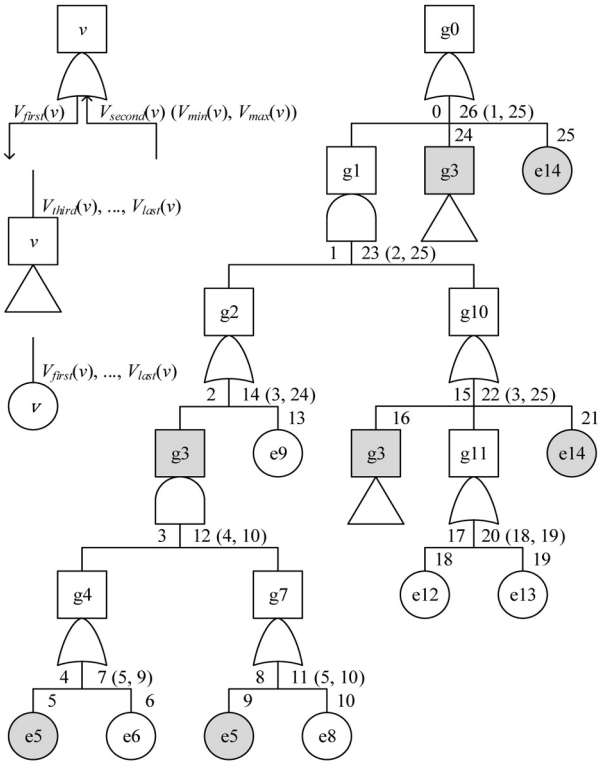


Fig. 1. DR method to find modules[19]

(Component Failure)이고, 정점사건은 시스템의 실패를 표현하는 사건이다. 고장수목의 기본적인 가정은 기본사건들 서로 상호 독립적이라는 것이다. 기기고장(기본사건)들은 중첩된 논리 게이트들을 통해 정점사건으로 논리적으로 전파된다.

고장수목은 Coherent 고장수목과 Non-coherent 고장수목으로 구분한다. Fig. 1의 고장수목과 같이 기기실패 기본사건들과 게이트들로 이루어진 고장수목을 Coherent 고장수목이라 부르고, 반대로 한 개 이상의 기기성공 기본사건($\overline{e5}$) 혹은 Negate($\overline{g2}$) 등을 가진 고장수목을 Non-coherent 고장수목이라 부른다.

$Event_{in}(v)$는 Node v로부터 아래방향으로 도달할 수 있는 Node들의 집합이고[19], $Event_{out}(v)$은 Node v를 지나지 않고 정점사건으로부터 도달할 수 있는 Node들의 집합으로 이 연구에서 정의하였다. 예를 들어, Node *g*3와 *e*14는 Node *g*10을 지나지 않고 정점사건으로부터 도달할 수 있기 때문에 이들은 $Event_{out}(g10)$와 $Event_{out}(g10)$의 멤버이다.

모듈은 모듈의 하부 Node들이 고장수목의 다른 곳에 존재하지 않는 게이트를 말한다[19]. 만약 Node v를 지나지 않고는 정점사건에서 $Event_{in}(v)$에 속한 Node를 방문할 방법이 없을 때 Node v는 모듈로 정의한다.

$Event_{in}(v) \cap Event_{out}(v) = \Phi$ (1)

식(1)의 모듈 정의에 의해, Fig. 1에서 정점사건 $g0$의 모듈은 아래와 같다.

$Module(g0) = \{g0, g3, g11\}$ (2)

예를 들어, $Event_{in}(g3)$와 $Event_{out}(g3)$는 다음과 같이 집합간에 중복되는 Node들이 없어 식 (1)을 만족하므로 $g3$는 모듈에 해당한다.

$Event_{in}(g3) = \{g4, e5, e6, g7, e8\}$
$Event_{out}(g3) = \{g1, g2, e9, g10, g11, e12, e13, e14\}$ (3)

또한, 정점사건 $g0$의 $Event_{out}(g0)$는 공집합이기 때문에 $Event_{in}(g0) \cap Event_{out}(g0)$는 역시 공집합으로 식 (1)을 만족하여 정점사건 $g0$는 모듈이다.

반면에, $Event_{in}(g10)$와 $Event_{out}(g10)$는 다음과 같이 중복되는 $Event_{in}(g3)$에 속한 Node들이 있어 식 (1)을 만족하지 않아 모듈이 아니다.

$Event_{in}(g10) = Event_{in}(g3) \cup \{g3, g11, e12, e13, e14\}$
$Event_{out}(g10) = Event_{in}(g3) \cup \{g1, g2, e9, e14\}$ (4)

본 논문에서 $Count(v)$는 고장수목에서 Node v의 반복 횟수를 가리킨다. Fig. 1에서의 중복 Node들의 반복 횟수는 다음과 같다.

$Count(g3) = 3$, $Count(e5) = 2$, $Count(e14) = 2$ (5)

고장 수목의 복잡성은 고장수목의 크기와 고장수목에서 반복되는 Node의 개수에 비례하여 증가한다. 또한, 반복 Node는 모듈의 개수를 줄이기 때문에 고장수목의 최소단절집합을 계산하는 것에 대한 어려움을 증가시킨다.

1.3 DR method to find modules

Dutuit와 Rauzy[19]는 깊이우선-좌측우선 순회에 기초한 효율적인 모듈 발견 알고리즘(DR 방법)을 제안하였다. Fig.1에 이 순회방법을 따른 Node들의 방문 순서를 Node들의 연결선에 따라 도식하였다.

Fig. 1에서 고장수목의 순회 순서는 {$g0$, $g1$, $g2$, $g3$, $g4$, $e5$, $e6$, $g4$, $g7$, $e5$, $e8$, $g7$, $g3$, $e9$, $g2$, $g10$, $g3$, $g11$, $e12$, $e13$, $g11$, $e14$, $g10$, $g1$, $g3$, $e14$, $g0$}이다. 여기에서 Node $g3$는 방문순서 3, 12, 16, 24에서 4번 방문되었다. 이를 다음과 같이 표시하였다.

$V_{first}(g3) = 3$, $V_{second}(g3) = 12$, $V_{last}(g3) = 24$ (6)

이 논문에서 Node 이름들에 나타나는 숫자는 각 Node의 처음 방문 순서를 의미한다.

추가된 정의로써 $V_{min}(v)$, $V_{max}(v)$, 그리고 $V_{last}(v)$를 다음과 같이 정의하였다.

$V_{min}(v) = \min_{w \in Event_{in}(v)} V_{first}(w)$ (7)
$V_{max}(v) = \max_{w \in Event_{in}(v)} V_{last}(w)$ (8)

DR방법에서, Node v의 방문순서가 다음의 부등식을 만족한다면 Node v는 모듈이다.

$V_{first}(v) < V_{min}(v) < V_{max}(v) < V_{second}(v)$ (9)

Node의 방문 순서들이 식 (9)의 부등식을 만족하면 식 (1)을 만족하기 때문에 이 Node는 모듈이다.

Fig. 1의 고장수목 Node들의 방문순서를 Table 1에서 요약하였다. 고장수목 Node들을 깊이우선-좌측우선 순회를 따라가 보면 방문순서들을 쉽게 찾을 수 있다. 고장수목에서 {$g0$, $g3$, $g11$}의 Node들의 방문 정보들은 식(9)의 부등식을 만족(식 (1)을 만족)하기 때문에, 이들은 모두 고장수목의 모듈들이다.

$Module(g0) = \{g0, g3, g11\}$ (10)

다른 노드들의 방문 정보들은 식(8)을 만족하지 않아 모듈에서 제외된다.

Table 1. DR method[19]

Node v	$V_{first}(v)$	$V_{second}(v)$	$V_{last}(v)$	$V_{min}(v)$	$V_{max}(v)$	Module
g0	0	26	26	1	25(e)	Yes
g1	1	23	23	2	25(e)	No
g2	2	14	14	3(a)	24(c)	No
g3	3(a)	12	24(c)	4	10	Yes
g4	4	7	7	5(b)	9(d)	No
e5	5(b)	5	9(d)			
e6	6	6	6			
g7	8	11	11	5(b)	10	No
e8	10	10	10			
e9	13	13	13			
g10	15	22	22	3(a)	25(e)	No
g11	17	20	20	18	19	Yes
e12	18	18	18			
e13	19	19	19			
e14	21	21	25(25)			

(a) $V_{first}(g3)$, (b) $V_{first}(e5)$, (c) $V_{last}(g3)$, (d) $V_{last}(e5)$, (e) $V_{last}(e14)$

DR방법은 고장수목의 이중 순회에 기초하고 있다. 첫 번째 순회에서, 각 Node들의 $V_{first}(v)$, $V_{second}(v)$, $V_{last}(v)$, $V_{min}(v)$들을 수집한다. 두 번째 순회에서, 각 Node들의 $V_{max}(v)$들을 수집하여 식 (9)에 의해 $V_{first}(v)$와 $V_{second}(v)$와 비교하여 각 Node들의 모듈 여부를 판단한다. 여기에서, 식 (7)의 $V_{min}(v)$은 $V_{first}(v)$에 기초하고 있기 때문에 고장수목의 처음 순회에서 각 Node v의 처음 방문에서 $V_{min}(v)$을 구할 수 있다.

1.4 본 연구의 목적

전술한 것처럼 1.3절의 DR방법은 고장수목의 이중 순회에 기초하고 있다. 반복하면 DR방법은 (1) 고장수목의 이중 순회가 필요하고, (2) Table 1의 방문순서를 저장하기 위한 추가 메모리들이 필요하고, (3) 저장된 방문순서를 비교하여 $V_{min}(v)$와 $V_{max}(v)$를 찾고 이들을 $V_{first}(v)$와 $V_{second}(v)$와 비교하기 위한 추가연산들이 요구된다.

이 연구의 목적은 (1) 추가 계산 메모리와 연산들을 최소화하거나 제거하는 일회 순회 방법을 찾고, (2) 새로운 방법으로 모듈 찾기를 가속화하고, (3) 새로운 방법의 효율성을 증명하기 위함이다. 이 목적 달성을 위해서, 본 연구에서 새로운 방법을 개발하였다. 이 새로운 알고리즘을 2장에 설명하였고, 이 새로운 방법의 증명을 3장에 추가하였고, 이 새로운 방법의 검증시험을 4장에 요약하였다. 이 새로운 방법의 장점을 4장과 5장에서 요약하였다.

2. 새로운 알고리즘

Fig. 2에서 모든 Node는 깊이우선-좌측우선 순회에 따라 방문되었다. Fig. 2에서 Node 이름들에 나타나는 숫자는 각 Node의 첫 번째 방문 순서 $V_{first}(v)$를 의미한다. 본 논문에서 효율적인 모듈 검색 방법 개발을 위해 다음 식 (11)과 (12)의 새로운 모듈척도를 도입하였다.

반복 Node v를 처음으로 방문하고 다음 Node w로 이동할 때, 모듈척도를 다음과 같이 단 한번 증가시킨다.

$$C_{in}(w)\ or\ C_{out}(w) = C_{out}(v) + (Count(v) - 1) \times \Delta_v \quad (11)$$

이 반복 Node v를 재방문하고 다음 Node w로 이동할 때마다, 모듈척도를 다음과 같이 반복적으로 감소시킨다.

$$C_{in}(w)\ or\ C_{out}(w) = C_{out}(v) - \Delta_v \quad (12)$$

식 (11)과 (12)에서, 모듈척도의 증감치 Δ_v는 다음과 같이 모든 Node에서 동일하게 적용한다.

$$\Delta_1 = \Delta_2 = \Delta_3 = \cdots = 1 \quad (13)$$

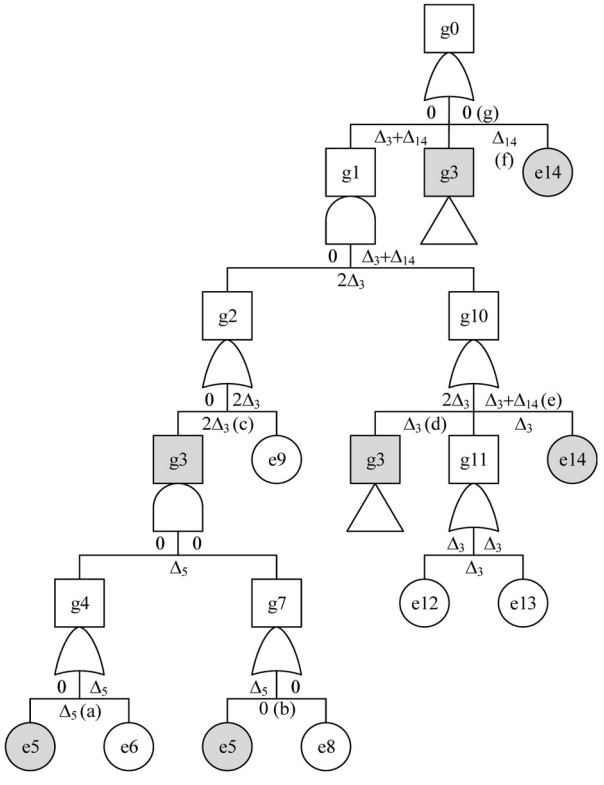
반복 Node $g0$, $g3$, $g11$들만이 식 (11)과 (12)의 모듈척도 변화를 유발하고, 고장수목의 순회를 마치면 모듈 척도의 총 변화량은 0이 되어야 한다. Fig. 2의 고장수목의 일회 순회를 따라 모듈척도 $C_{in}(v)$와 $C_{out}(v)$의 변화를 Table 2에 요약하였다.

만약 Node v를 진입할 때와 떠날 때 모듈척도의 변화가 없고

$$C_{out}(v) - C_{in}(v) = 0 \quad (14)$$

Node v의 방문순서가 다음 부등식을 만족하면,

$$V_{first}(v) < V_{min}(v) \quad (15)$$



(a) $C_{in}(e6) = C_{out}(e5) + (Count(e5) - 1)\Delta_5 = 0 + \Delta_5 = \Delta_5$
(b) $C_{in}(e8) = C_{out}(e5) - \Delta_5 = \Delta_5 - \Delta_5 = 0$
(c) $C_{in}(e9) = C_{out}(g3) + (Count(g3) - 1)\Delta_3 = 0 + 2\Delta_3 = 2\Delta_3$
(d) $C_{in}(g11) = C_{out}(g3) - \Delta_3 = 2\Delta_3 - \Delta_3 = \Delta_3$
(e) $C_{out}(g10) = C_{out}(e14) + (Count(e14) - 1)\Delta_{14} = \Delta_3 + \Delta_{14}$
(f) $C_{in}(e14) = C_{out}(g3) - \Delta_3 = (\Delta_3 + \Delta_{14}) - \Delta_3 = \Delta_{14}$
(g) $C_{out}(g0) = C_{out}(e14) - \Delta_{14} = \Delta_{14} - \Delta_{14} = 0$

Fig. 2. New method to find modules.

Node v는 고장수목의 모듈이다. 이 모듈 판단의 조건인 식 (14)과 (15)의 증명을 3장에 설명하였다.

Fig. 2와 Table 2에서 보이는 것처럼, Node $g0$, $g3$, $g11$의 모듈척도와 방문순서들은 식 (14)와 (15)를 만족한다. 그래서, 이 3개의 Node는 고장수목의 모듈들이다.

$$Module(g0) = \{g0, g3, g11\} \quad (16)$$

식 (11)에서 (14)까지의 모듈척도 $C_{in}(v)$와 $C_{out}(v)$ 그리고 식 (15)의 방문정보 $V_{first}(v)$와 $V_{min}(v)$들은 Node의 재방문 없이 Node의 한번 방문만으로 취득할 수 있는 정보들이다. 즉, 고장수목의 일회 순회만으로 모듈들을 찾아낼 수 있는 장점이 있다. 이 새로운 방법은 이중 순회에 기초한 DR 방법과 달리 일회 순회고, DR 방법의 Table 1에 나타난 추가적인 메모리나 비교연산을 요구하지 않는다.

3. 알고리즘의 증명

Table 2에서, 각 Node들이 모듈이 되지 못하는 경우들을 모아보면 아래와 같다. (Case 1) Node $g1$이 모듈이 되지 못하는 이유는 중복 Node $g3$와 $e14$가 Node $g1$의 방문 이후에 존재하기 때문이다. (Case 2) Node $g2$가 모듈이 되지 못하는 이유는 중복 Node $g3$가 Node $g2$의 방문 이후에 존재하기 때문이다. (Case 3) Node $g4$가 모듈이 되지 못하는 이유는 중복 Node $e5$가 Node $g4$의 방문 이후에 존재하기 때문이다. (Case 4) Node $g7$이 모듈이 되지 못하는 이유는 중복 Node $e5$가 Node $g7$의 방문 이전에 존재하기 때문이다. (Case 5) Node $g10$이 모듈이 되지 못하는 이유는 중복 Node $g3$가 Node $g10$의 방문 이전에 존재하기 때문이다. (Case 6) Node $g10$이 모듈이 되지 못하는 이유는 중복 Node $e14$가 Node $g10$의 방문 이후에 존재하기 때문이다.

Case 1, 2, 3, 6은 식 (14)를 이용하여 모듈이 아님을 판단하고, 반면에 Case 4와 5는 식 (15)를 이용하여 모듈이 아님을 판단한다. 다시 말해, 식 (14)로는 Node v의 방문 이후에 존재하는 중복 Node의 존재를 판단하여 Node v가 모듈이 아님을 판단하고, 식 (15)로는 Node v의 방문 이전에 이미 방문된 중복 Node의 존재를 판단하여 Node v가 모듈이 아님을 판단한다. 결국, 식(14)와 (15)는 모듈 여부를 판단하기 위한 필요충분 조건이다.

본 논문의 식 (13)처럼 증감치를 설정하면, Table 2의 $g10$ Node에서 $C_{in}(g10) = C_{out}(g10) = 2$로 같아지는 경우가 발생한다. 이 Node $g10$은 식 (15)의 조건에 위배되기 때문에 모듈에서 제외된다. 즉, 식(14)와 (15)는 동시에 모듈 여부를 판단하기 위한 필요충분 조건이다.

Table 2. New method

Node v	$C_{in}(v)$	$C_{out}(v)$	$C_{out}(v)$ - $C_{in}(v)$	$V_{first}(v)$	$V_{min}(v)$	Module
g0	0	0	0	0	1	Yes
g1	0	$\Delta_3+\Delta_{14}$	$\Delta_3+\Delta_{14}$	1	2	No
g2	0	$2\Delta_3$	$2\Delta_3$	2	3(a)	No
g3	0	0	0	3(a)	4	Yes
g4	0	Δ_5	Δ_5	4	5(b)	No
e5	0	0	0	5(b)		
e6	Δ_5	Δ_5	0	6		
g7	Δ_5	0	$-\Delta_5$	8	5(b)	No
e8	0	0	0	10		
e9	$2\Delta_3$	$2\Delta_3$	0	13		
g10	$2\Delta_3$	$\Delta_3+\Delta_{14}$	0	15	3(a)	No
g11	Δ_3	Δ_3	0	17	18	Yes
e12	Δ_3	Δ_3	0	18		
e13	Δ_3	Δ_3	0	19		
e14	Δ_3	$\Delta_3+\Delta_{14}$	Δ_{14}	21		

(a) $V_{first}(g3)$
(b) $V_{first}(e5)$

4. 알고리즘의 응용

이 연구에서 개발 한 새로운 알고리즘의 효율성을 증명하기 위하여, Table 3의 여섯 개 고장수목을 원자력발전소의 PSA모델로부터 선택하였다. 그리고 본 논문의 새로운 방법과 DR방법[19]을 이 고장수목들로 시험하였다.

Table 3에서 N1부터 N4는 Non-coherent 고장수목이고 C1과 C2는 Coherent 고장수목이다. 여기에서, Non-coherent 고장수목이란 고장수목 내부에 Negate나 성공사건들을 가진 고장수목을 말한다. Table 3의 게이트와 기본사건의 개수를 보면, 여섯 개 고장수목은 큰 모델이다.

Table 4에서 보이는 것처럼, 새로운 방법과 DR방법으로 동일한 모듈들을 찾아내었다. 여기에서, 두 방법으로 발견한 모듈들은 하나씩 비교하여 같음을 확인하였다. 새로운 방법은 고장수목에서 DR방법보다 적어

Table 3. Tested fault trees

Fault tree		Gates	Events	Successful gates (Negates)	Successful events
Non-coherent fault tree	N1	38,704	3,198	429	12
	N2	22,255	6,105	381	4
	N3	23,715	3,777	19	16
	N4	14,832	2,374	2,981	0
Coherent fault tree	C1	12,298	8,350	0	0
	C2	42,910	44,260	0	0

Table 4. Comparison of new method and DR method

Fault tree	Type 1 modules (a)	Type 2 modules (b)	DR Method (sec) (c)	New method (sec) (d)	(d)/(c)
N1	581	81	4.1	1.8	42.6%
N2	1,011	36	4.1	1.8	43.9%
N3	557	33	2.6	1.2	44.6%
N4	337	4	0.4	0.2	44.4%
C1	1,639	32	1.7	0.7	43.9%
C2	6,226	390	4.3	1.7	39.9%

No restructuring
Tested on CPU i7-4790 and RAM 32 GB
(a) Type 1 modules consist of basic events (no restructuring)
(b) Type 2 modules consist of gates and basic events (no restructuring)
(c) 1,000 calculations
(d) 1,000 calculations

도 2배 이상 빠른 모듈 발견시간을 보여주었다. 게다가, 새로운 방법은 DR방법보다 계산 메모리를 적게 사용한다.

고장수목 재구성 여부와 모듈 사용 여부에 따른 고장수목 계산시간을 Table 5에 비교하였다. Table 5는 (1) 모듈의 사용만으로는 고장수목 계산시간을 크게 단축시킬 수 없고, (2) 고장수목의 재구성과 함께 모듈을 사용해야 고장수목 계산시간을 크게 단축시킬 수 있음을 보여준다. 고장수목을 계산하기 전에 수행되는 대표적인 고장수목의 재구성은 아래와 같다.

$$g1 = (g2 \cup g3) \cap (g2 \cup g4) = g2 \cup (g3 \cap g4) \qquad (17)$$

이 변환을 통해 최소단절집합 생성을 위한 불리안 계산양을 많이 줄일 수 있고, 동시에 (g3 ∩ g4)가 모듈일 경우 고장수목 계산은 더욱 빨라진다. 1.1절에서 전술한 바와 같이 고장수목의 재구조는 가능한 큰 모듈을 최대한 많이 얻기 위해서 수행된다.

Table 5. Effects of fault tree restructuring and modules to cutset generation

Fault tree	Truncation limit	Cutsets	No restructuring		Restructuring	
			No modules (sec)	Modules (sec)	No modules (sec)	Modules (sec)
N1	1.0E-11	35,889	11.6	9.1	14.6	4.6
N2	1.0E-08	273	44.1	43.3	22.2	2.7
N3	1.0E-09	5,664	14.9	12.8	6.4	2.6
N4	1.0E-10	16,297	5.2	4.4	6.2	2.6
C1	1.0E-10	1,817	2.1	1.9	2.0	1.0
C2	1.0E-06	212	20.1	46.1	17.6	4.0

Tested on CPU i7-4790 and RAM 32 GB

Table 4와 Table 5의 계산에서 알 수 있듯이, 새로운 방법의 중요한 특징은

(1) 새로운 방법은 고장수목의 일회 순회를 기초로 한다. Node들의 재방문 없이 고장수목의 일회 순회만으로 모듈들을 찾아낼 수 있는 장점이 있다.

(2) 새로운 방법은 식 (11)에서 (14)까지의 휘발성 모듈척도 $C_{in}(v)$와 $C_{out}(v)$ 그리고 식 (15)의 노드 방문정보 $V_{first}(v)$와 $V_{min}(v)$의 값을 추적하고 비교한다.

(3) 새로운 방법은 어떠한 추가 메모리나 연산을 요구하지 않는다. DR 방법에서는 Table 1의 고장수목 순회 중에 Node들의 방문 정보를 저장하고, 고장수목 순회를 마친 마지막 단계에서 각 Node들의 모듈여부를 판단하게 된다. 새로운 방법에서는 각 Node들의 방문정보를 저장하고 재순회하는 과정이 필요 없다.

(4) 고장수목의 재구성과 함께 사용하면 계산시간을 크게 절감할 수 있다.

(5) 새로운 방법은 매우 간단하고 빠른 모듈 발견 방법이어서, 다양한 고장수목 소프트웨어에 쉽게 구현할 수 있다.

5. 결 론

3장의 새로운 알고리즘은 휘발성 정보인 모듈 척도와 방문순서를 추적하고 비교하여 일회 순회만으로 고장수목의 모듈들을 찾아낸다. 4장에서 모듈의 사용은 고장수목의 재구성과 함께 사용되어야 고장수목 계산시간을 크게 단축시킬 수 있음을 보여주었다.

4장의 계산과 같이, 제안된 방법은 대형 Coherent와 Non-coherent 고장수목의 모듈들을 빠르게 찾는다. 또한, 전통적인 Boolean 대수, 이진결정다이어그램(Binary Decision Diagram, BDD), 또는 Zero-suppressed BDD에 기초한 다양한 고장수목 계산 소프트웨어에 본 논문의 방법을 쉽게 구현할 수 있다. 원자력 발전소의 효율적인 확률론적안정성평가, 그리고 화공, 항공, 우주 시스템과 같은 안전필수 계통의 리스크 평가에 사용 중인 고장수목분석 소프트웨어에 이 방법을 구현하는 것을 추천한다.

감사의 글: 이 논문은 2014년도 세종대학교 교내연구비 지원에 의한 논문입니다. 또한, 본 연구는 원자력안전위원회와 한국방사선안전재단의 지원을 받아 수행한 원자력안전연구사업의 연구결과입니다(No. 1403011).

References

1) W. S. Lee, D. L. Grosh, F. A. Tillman and C. H. Lie, “Fault

Tree Analysis, Methods, and Applications - A Review", IEEE Transactions on Reliability, Vol. R-34, No. 3, pp. 194-203, 1985.
2) C. A. Ericson, "Fault Tree Analysis - A History", in Proceedings of the 17th International System Safety Conference, pp. 1-9, 1999.
3) U.S. NRC, Reactor Safety Study - an Assessment of Accident Risk in U.S. Commercial Nuclear Power Plants, WASH-1400, NUREG-75/014, 1975.
4) P. Chatterjee, "Modularization of Fault Trees: A Method to Reduce the Cost of Analysis", Reliability and Fault Tree Analysis, SIAM, pp. 101-137, 1975.
5) Z. W. Birnbaum and J. P. Esary, "Modules of Coherent Binary Systems", SIAM J Appl. Math., Vol. 13, pp. 442-462, 1965.
6) O. M. Locks, "Modularizing, Minimizing, and Interpreting the K & H Fault-Tree", IEEE Transactions on Reliability, Vol. R-30, No. 5, pp. 411-415, 1981.
7) A. Rosenthal, "Decomposition Methods for Fault Tree Analysis", IEEE Transactions on Reliability, Vol. R-29, No. 2, pp. 136-138, 1980.
8) L. Camarinopoulos and J. Yllera, "An Improved Top-down Algorithm Combined with Modularization as a Highly Efficient Method for Fault Tree Analysis", Reliability Engineering, Vol. 11, No. 2, pp. 93-108, 1985.
9) J. M. Wilson, "Modularizing and Minimizing Fault Trees", IEEE Transactions on Reliability, Vol. R-34, No. 4, pp. 320-322, 1985.
10) L. Camarinopoulos, "Advanced Concepts in Fault Tree Modularization", Nucl. Eng. Des., Vol. 91, pp. 79-91, 1986.
11) T. Kohda, E. J. Henley and K. Inoue, "Finding Modules in Fault Trees", IEEE Transactions on Reliability, Vol. 38, No. 2, pp. 165-176, 1989.
12) K. D. Russell and D. M. Rasmuson, "Fault Tree Reduction and Quantification-an Overview of IRRAS Algorithms", Reliability Engineering and System Safety, Vol. 40, No. 2, pp. 149-164, 1993.
13) I. Niemela, "On Simplification of Large Fault Trees", Reliability Engineering and System Safety, Vol. 44, No. 2, pp. 135-138, 1994.
14) S. Contini, "A New Hybrid Method for Fault Tree Analysis", Reliability Engineering and System Safety, Vol. 49, No. 1, pp. 13-21, 1995.
15) K. Reay and J. D. Andrews, "A Fault Tree Analysis Strategy using Binary Decision Diagrams", Reliability Engineering and System Safety, Vol. 78, No. 1, pp. 45-56, 2002.
16) H. Sun and J. D. Andrews, "Identification of Independent Modules in Fault Trees which Contain Dependent Basic Events", Reliability Engineering and System Safety, Vol. 86, pp. 285-296, 2004.
17) U.S. NRC, PRA Procedures Guide: A Guide to the Performance of Probabilistic Risk Assessments for Nuclear Power Plants, NUREG/CR-2300, 1983.
18) W. S. Jung, "ZBDD Algorithm Features for an Efficient Probabilistic Safety Assessment", Nuclear Engineering and Design, Vol. 239, pp. 2085-2092, 2009.
19) Y. Dutuit and A. Rauzy, "A Linear Time Algorithm to Find Modules of Fault Trees", IEEE Transactions on Reliability, Vol. 45, No. 3, pp. 422-425, 1996.