MDD を用いた多状態システムの解析に関する一考察

広島大学・大学院先進理工系科学研究科 *
岡村　寛之，鄭　俊俊，土肥　正
Hiroyuki Okamura, Junjun Zheng, Tadashi Dohi
Graduate School of Advanced Science and Engineering,
Hiroshima University

1　はじめに

近年，IoT や AI を活用したシステムが多岐にわたる分野で応用されている．特に，自動運転のようなセーフティクリティカルシステムにおいて，複雑化が進む一方で，その信頼性を確保することが喫緊の課題となっている．このような状況において，システムの信頼性を定性的・定量的に解析することは，製品開発の全ての段階で欠かせないものである．

信頼性解析には，大きく分けて二つのアプローチが存在する．一つは，故障木を用いたシステムの故障原因を分析する静的アプローチであり，もう一つは，マルコフモデルを用いて時間の経過による故障発生を考慮する動的アプローチである．本稿では，静的な信頼性解析を中心に議論を進める．

静的な信頼性解析において，故障木はシステムの故障が起こる原因を分析するための有効な手法である．故障木は，システムの故障が起こる原因を AND/OR ゲートを用いて表現する木構造であり，システムの信頼性を評価するための基本的なモデルとして広く用いられている．故障木を用いた信頼性解析は，故障木の構造関数を多項式に変換することで，トップ事象の発生確率を求めることができる．しかしながら，故障木には繰り返し事象や k-out-of-n ゲートを含む複雑な構造が存在し，これらの構造を多項式に変換するこ

〒739-8527 東広島市鏡山 1-4-1

とは困難である．

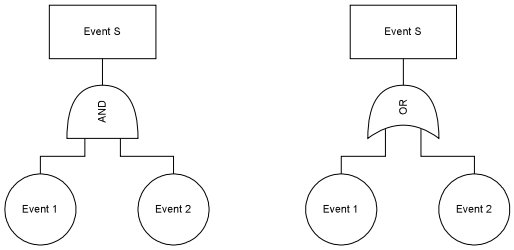
故障木の構造関数表現に対して BDD (Binary Decision Diagram) がよく用いられる．これを用いることで，繰り返し事象や k-out-of-n ゲートを含む故障木に対しても正確に多項式を求めることができる．BDD は，論理関数をグラフ構造で表現する手法であり，論理関数の簡約化や最適化に有効である．

本稿では多状態システムに対する静的解析手法に関する議論を行う．故障木に基づいたシステム表現は故障と正常の二つの状態をとるため，2 状態システムとも呼ばれる．しかしながら，多くのシステムは故障だけでなく，劣化や異常などの多様な状態を持つ．2 状態システムに対する BDD と同じく，多状態システムでは BDD を拡張した MDD (Multi-valued Decision Diagram) が用いられる [1, 2]．MDD は，多状態システムの信頼性解析において，故障木の構造関数を効率的に表現することができる．

2　2 状態システム

2.1　モデル表現

2 状態システムとは，システムを構成する要素が稼働状態，故障状態などの 2 状態，かつシステム全体の状態も 2 状態で表現されるシステムであり，信頼性工学の分野でよく用いられる．代表的な表現として故障木がある．故障木（以下，FT）はシステム障害とその原因（部品の故障など）の関係を表すモデルとして利用され，システム故障に至る原因解析などを事前に抽出するために利用される．基本的な FT では原因と事象を AND ゲート（すべての原因が発生したら該当する事象が発生）, OR ゲート（少なくとも一つの原因が発生したら該当する事象が発生）あるいは k-out-of-n ゲート（n 個の原因のうち k 個が発生したら該当する事象が発生）でモデル化していく．FT は何らかの事象を表すノードと AND/OR/k-out-of-n ゲートから構成される木構造となっており，親ノードに対応する事象の発生条件をゲートと子ノードで表現する．例えば，図 1a は AND ゲートを用いて Event 1 と Event 2 が発生した時に Event S が発生する関係を表しており，図 1b は OR ゲートを用いて Event 1 と Event 2 の少なくともどちらか一方が発生した時に Event S が発生する関係を表している．実際の FT では複数の階層で構成されトップ事象（木の根ノードに対応する事象）の発生条件を原因事象（木の葉ノードに対応する事象）を用いて記述する．信頼性分析で FT を用いる場合，トップ事象を「システムの故障」とする場合と「システムが稼働」とする場合の二通りが考えられる．以降では，原因事象を「該当する部品が稼働」，トップ事象を「システムが稼働」を表すものとして考える．



(a) AND gate (b) OR gate

図 1: FT の例.

FT の解析を行うために構造関数（structure function）を定義する．いま FT における原因事象を表す変数 x_i, $i = 1, \ldots, n$ を

$$x_i = \begin{cases} 1 & (\text{原因 } i \text{ が発生している}) \\ 0 & (\text{原因 } i \text{ が発生していない}) \end{cases} \tag{1}$$

と定義する．この時，トップ事象の発生を表す変数 x_S が以下の式で与えられるものとする．

$$x_S = \varphi(x_1, x_2, \ldots, x_n). \tag{2}$$

ここで，$\varphi(x_1, x_2, \ldots, x_n)$ は構造関数と呼ばれる論理関数であり，各原因事象の状態 x_i に応じてトップ事象の状態（0 または 1）を出力する．例えば，図 1a で，Event 1, Event 2 の状態（0 または 1）を表す変数 x_1, x_2 が与えられた時，Event S の状態 x_S は

$$x_S = x_1 \wedge x_2 \tag{3}$$

として与えられる．ここで $\wedge$ は論理積を表す．また，図 1b の場合は

$$x_S = x_1 \vee x_2 \tag{4}$$

となる．ここで，$\vee$ は論理和を表す．一般に AND ゲートは信頼性では直列構造を表現し，どちらか一方の部品が壊れるとシステム全体が壊れる（どちらも稼働していないとシステムが稼働しない）ような事象を表す．一方，OR ゲートは並列構造と呼ばれ，複数の部品が同時に壊れない限りシステム全体が壊れないような構造を表す（いずれかが稼働していればシステムも稼働）．k-out-of-n ゲートは AND ゲート，OR ゲートの中間に位置

するゲートであり，n 個の部品のうち k 個以上が稼働していればシステムが稼働する構造を表す．例えば，$k = 2, n = 3$ の場合，その構造関数は

$$x_S = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3) \tag{5}$$

となる．構造関数の論理演算による表記は簡単ではあるが実際のトップ事象の発生確率を求めるためには構造関数の多項式表現あるいは，後述する BDD (Binary Decision Diagram) を用いた手法が必要となる．

2.2 BDD による解析

FT では，各原因事象の発生確率からトップ事象の発生確率を算出することが必要となる．トップ事象の計算には構造関数の多項式表現が必要となる．これは，積事象を $x \wedge y \to xy$，和事象を $x \vee y \to 1 - (1 - x)(1 - y)$ と置き換えることで得られる．いま，構造関数 $\varphi(x_1, \ldots, x_n)$ から対応する多項式関数 $f_\varphi(x_1, \ldots, x_n)$ が得られたとする．いま，$X_1, \ldots, X_n$ を原因事象の発生を表す指標確率変数，X_S をトップ事象の発生を表す指標確率変数とすると，トップ事象の発生確率は

$$P(X_S = 1) = f_\varphi(p_1, p_2, \ldots, p_n) \tag{6}$$

となる[*1]．ここで，$p_i = P(X_i = 1)$ は原因事象 X_i の発生確率である．しかしながら，繰り返し事象や k-out-of-n ゲートを含む FT では上記の単純な置き換えでは計算できない．繰り返し事象とは，同じ原因事象が葉ノードに複数回現れることを表す．図 2 は繰り返し事象を含む FT の例を示している．この場合，Event 1 が葉ノードに複数回現れている．この FT に対する構造関数は

$$\varphi(x_1, x_2) = (x_1 \vee x_2) \wedge x_1 \tag{7}$$

となる．論理積，論理和を単純に $x \wedge y \to xy$，$x \vee y \to 1 - (1 - x)(1 - y)$ で置き換えた多項式を作ると

$$f_\varphi(x_1, x_2) = x_1^2 + x_1 x_2 - x_1^2 x_2 \tag{8}$$

となる．実際は式 (7) は $\varphi(x_1, x_2) = x_1$ と論理的に等価であるため，求めたい多項式は $f_\varphi(x_1, x_2) = x_1$ である．このように繰返し事象を含む FT に対しては，単純な置き換えで正しい多項式を得ることができない．

[*1] ここでは原因事象が独立を仮定している．

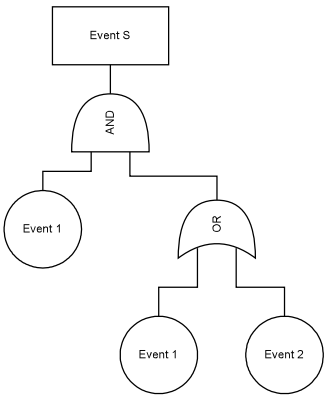


図 2: 繰り返し事象を含む FT の例.

このような問題から，FT では構造関数を BDD (Binary Decision Diagram) で表現する手法が行われる [3]．BDD とは論理関数を非循環グラフにより表す手法であり，同じ論理値を出力するサブグラフを共有することで論理関数をコンパクトに表現することができる [4]．いま，論理関数 $\varphi(x_1, \ldots, x_n)$ が与えられたとき，原因事象 1 の状態に着目するとシャノン分解から以下のように記述できる．

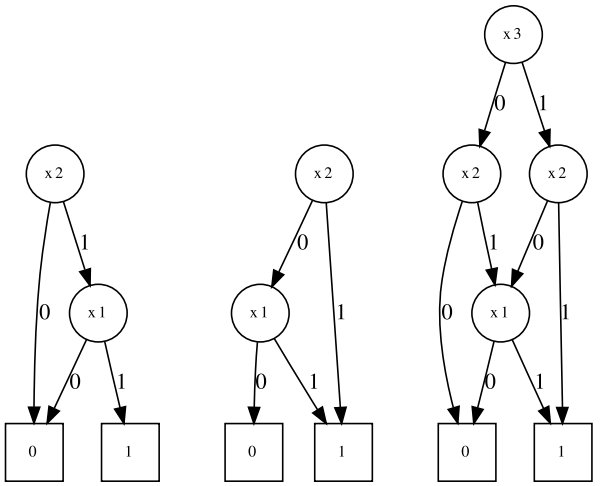
$$\varphi(x_1, \ldots, x_n) = (x_1 \wedge \varphi(1, x_2, \ldots, x_n)) \vee (\overline{x_1} \wedge \varphi(0, x_2, \ldots, x_n)). \tag{9}$$

同様に $\varphi(1, x_2, \ldots, x_n)$, $\varphi(0, x_2, \ldots, x_n)$ の $x_2, \ldots, x_n$ についてもさらにシャノン分解を行うことができ，最終的には $\varphi(1, 0, \ldots, 1)$ のように具体的な値が与えられた時の論理値まで分解できる．BDD では，上記の部分的に値が与えられた $\varphi(1, x_2, \ldots, x_n)$ や $\varphi(0, x_2, \ldots, x_n)$ をノードとする二分木を構成することで論理関数を表現する．図 3a，図 3b，図 3c はそれぞれ式 (3)，式 (4)，式 (5) の構造関数に対応する BDD 表現を示している．一番上にあるノードから各変数が取る値の枝を辿ることで最終的な構造関数の出力として 0 または 1 が得られることがわかる．

また，構造関数の多項式に対するシャノン分解は以下の再帰式となる．

$$f_\varphi(x_1, \ldots, x_n) = x_1 f_\varphi(1, x_2, \ldots, x_n) + (1 - x_1) f_\varphi(0, x_2, \ldots, x_n). \tag{10}$$

これと，式 (6) の性質を利用すると BDD 表現からトップ事象の発生確率（故障確率や信頼度）を計算するアルゴリズムを構築することができる．Algorithm 1 は BDD で表現された構造関数をつかってトップ事象の発生確率を計算する手続きを示している．`visited`, `haskey` はノードをキーとしたハッシュテーブルとハッシュテーブルに特定のキーがある



(a) AND ゲートの BDD　(b) OR ゲートの BDD　(c) 2-out-of-3 ゲートの BDD

図 3: 構造関数の BDD 表現.

Algorithm 1 Computation Top Event Probability.

```
function prob(node)
  return visited[node] if haskey(visited, node)
  return 0 if is_zero(node)
  return 1 if is_one(node)
  p = get_prob(node)
  result = p * prob(get_one(node)) + (1-p) * prob(get_zero(node))
  visited[node] = result
  return result
end
```

かどうか確認をする関数を表している．これは，一度計算した確率を何回も計算しないようにするためのキャッシュとして機能する．関数 `get_prob` は BDD ノードに対応する変数の発生確率を取得するための関数．`get_one`, `get_zero` はそれぞれ，1-枝，0-枝の子ノードを取得するための関数である．

静的な構造分析では，極小パスベクトル（Minimal Path Vector）の導出が重要な役割を果たす．いま，構造関数 $\varphi(x_1, \ldots, x_n)$ の入力値 $\boldsymbol{x} = (x_1, \ldots, x_n)$ を状態ベクトルと呼ぶことにすると，極小パスベクトルとは，$\varphi(\boldsymbol{x}) = 1$ となる状態ベクトルのうち，任意

の x_i の値が 1 から 0 に変化した新しい状態ベクトル $\boldsymbol{x}'$ が $\varphi(\boldsymbol{x}')=0$ となるような状態ベクトル $\boldsymbol{x}$ を示す．また，極小パスベクトルの集合を極小カット集合（MPS: Minimal Path Set）と呼ぶ．これは，現在正常に稼働している部品の一つでも故障すると，システムが故障するような部品の状態を意味している．FT が AND, OR, k-out-of-n ゲートで構成されるとき，このシステムは単調，つまり，部品の故障によってシステム状態がより故障しにくい状態には移行しない性質を持つ．システムが単調であるとき，MPS と構造関数は 1 対 1 に対応するため，通常の FT に対して MPS は一意に決定される．例えば，式 (3) のシステムの場合，MPS は $\{(1,1)\}$ となり，式 (4) の場合は $\{(1,0),(0,1)\}$ となる．また，式 (5) の場合は，$\{(1,1,0),(0,1,1),(1,0,1)\}$ となる．このような，MPS の導出は信頼性の分析において重要な役割を果たす．

FT が繰り返し事象を含む場合，MPS の導出は容易ではない．ここでは，BDD を用いた MPS の導出アルゴリズムの紹介を行う [3]．基本的なアイデアは構造関数を表す BDD から，MPS の集合を表す BDD を再構成する．いま，シャノン分解によって構造関数が

$$\varphi(x_1,\ldots,x_n)=(x_1\wedge\varphi(1,x_2,\ldots,x_n))\vee(\overline{x_1}\wedge\varphi(0,x_2,\ldots,x_n)) \tag{11}$$

のように分解されることを思い出す．システムが単調であることを仮定すると，$\varphi(0,x_2,\ldots,x_n)$ の $(x_2,\ldots,x_n)$ に対する MPS は $x_1=0$ とすることで $\varphi(x_1,\ldots,x_n)$ の $(x_1,\ldots,x_n)$ の MPS になっていることが容易にわかる．一方，$\varphi(1,x_2,\ldots,x_n)$ の $(x_2,\ldots,x_n)$ に対する MPS を求めたとき，$\varphi(0,x_2,\ldots,x_n)$ の $(x_2,\ldots,x_n)$ に対する MPS と同じ集合があった場合，明らかに MPS ではない，そのため，$\varphi(1,x_2,\ldots,x_n)$ の MPS から $\varphi(0,x_2,\ldots,x_n)$ の MPS を除いた集合を求め，その集合要素に対して $x_1=1$ を付け加えることで $\varphi(1,\ldots,x_n)$ の MPS を求めることができる．

Algorithm 2 は，上記の考えに基づいて構造関数の BDD 表現から MPS を導出するアルゴリズムである [3]．関数 `minsol` は再帰的に BDD の 0-枝と 1-枝を辿るようにして実行される．まず 0-枝のノードに対して `minsol` を実行し，MPS を得る．一方，1-枝については 0-枝に含まれる MPS を除いたノードに対して `minsol` を実行して MCS を取得し，最終的に，ぞれぞれ 0,1 の値を決定した際に得られた MPS を子ノードとするノードを `create_node` 関数を使って作成している．アルゴリズム内の `is_zero`, `is_one` は，それぞれ 0-終端，1-終端ノードかどうかを判別する関数．関数 `get_one` は 1-枝の子ノードを取得する．また演算子-は BDD 上の集合の差を行う．

Algorithm 2 MPS.

```
function minsol(node)
  return node if is_zero(node) || is_one(node)
  x = minsol(get_zero(node))
  z = get_one(node) - x # set difference
  y = minsol(z)
  return create_node(x, y)
end
```

3 多状態システム

多状態システムは 2 状態システムの拡張であり，構成要素（コンポーネント）ならびにシステムが複数の状態をとることを許す．信頼性分野において多状態システムは，故障だけでなく，劣化や異常などの多様な状態を持つシステムを表現するために用いられる．一方で，整数表現によるモデル化であるため，より広範囲のシステムを表現することができる．

以下の定義を与える．

- n 個の互いに独立なコンポーネント $C = \{1, 2, \ldots, n\}$ がある．
- コンポーネント i は状態 $\Omega_i = \{0, 1, \ldots, m_i\}$ を持つ．ここで，状態 0 は故障状態，状態 m_i は正常状態，他の状態は劣化状態（度合い）を表し，順序関係 $0 < 1 < \ldots < m_i$ がある．
- システムは状態 $S = \{0, 1, \ldots, m\}$ を持つ．ここで，状態 0 は故障状態，状態 m は正常状態，他の状態は劣化状態（度合い）を表し，順序関係 $0 < 1 < \ldots < m$ がある．
- 各コンポーネントの状態を表すベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ が $\boldsymbol{x} \in \Omega_C = \Omega_1 \otimes \cdots \otimes \Omega_n$ となる．
- 構造関数 $\varphi : \Omega_C \to S$ は Ω_C から S への全射とする．

多状態システムは構造関数を与えることでその特徴が決定されるが複数のコンポーネントに対して直接構造関数を与えることは難しい．2 状態システム同様に AND, OR などの演算を定義することが重要となる．2 状態とは異なり，複数の状態をとるため，整数の

四則演算，min/max 演算，比較演算子，if-then-else 演算を用いて構造関数を表現することが必要となる．

例えば，2 状態システムにおける k-out-of-n を四則演算等を用いた多状態表現すると，

$$x_S = \begin{cases} 1 & \text{if } x_1 + x_2 + \ldots + x_n \geq k \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

となる．また AND ゲートには min 演算，OR ゲートは max 演算が相当する．

上記で見るように整数演算の表現を用いたとしても構造関数を単純な多項式で表現することは容易ではない．そのため，2 状態システムに対する BDD を拡張した MDD (Multi-valued Decision Diagram) を用いることを考える．MDD は，多状態システムの信頼性解析において，構造関数を効率的に表現することができる．

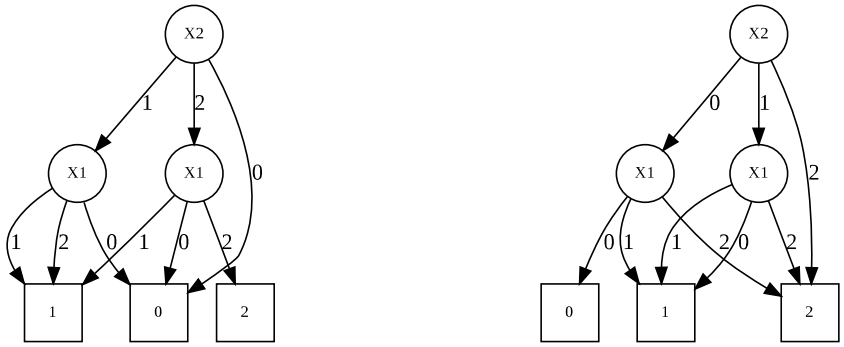
BDD 同様に MDD もシャノン分解を利用した木の表現となる．多状態システムにおける構造関数 $\varphi(x_1, \ldots, x_n)$ は，以下のように分解される．

$$\varphi(x_1, \ldots, x_n) = \sum_{v=0}^{m_1} I(x_1 = v)\varphi(v, x_2, \ldots, x_n). \tag{13}$$

ここで，$I(x_1 = v)$ は $x_1 = v$ の条件を表す指示関数である．同様に，$\varphi(v, x_2, \ldots, x_n)$ は $\varphi(x_1, \ldots, x_n)$ の $x_1 = v$ の条件下での部分構造関数である．部分構造関数を木のノードとするとき，BDD に類似した木構造が得られる．図 4a および図 4b は，多状態システムにおける MIN および MAX 演算の MDD 表現を示している．ここでは各変数 X1, X2 は 0, 1, 2 の 3 状態をとるものとしている．BDD と同じであるが，各ノードからの枝が各変数のドメイン数（3 本）になっていることがわかる．また，終端ノードについても 2 値から多値に変更されている．

MDD による構造関数表現からトップ事象の確率を計算するためには，BDD の場合と同様に，再帰的なアルゴリズムで実現できる．一方で，多状態システムでは，2 状態システムの MPS に対応する極小ベクトル集合（Minimal Vector Set）を求めることが重要となる．極小ベクトル集合とは，いずれか一つの部品の状態が低下したらシステムの状態が低下する状態ベクトルの集合のことであり，システムの状態毎に求めることができる．つまり，システム状態 2 に対する極小ベクトル集合は，いずれか一つの部品の状態が低下したらシステム状態が 1 以下になるような状態ベクトルの集合である．

ここでは，構造関数の単調性を仮定し，構造関数の MDD 表現を用いて極小ベクトル集合を求めるアルゴリズムを考える．基本的なアイデアは 2 状態システムに対する MPS のアルゴリズムと同様であり，構造関数の MDD 表現から極小ベクトルを表す MDD を再



(a) MIN 演算の MDD　　(b) MAX 演算の MDD

図 4: 構造関数の MDD 表現（MIN/MAX）.

Algorithm 3 MDD の極小ベクトル集合.

```
function minsol(x)
  return x if x が終端ノード
  u[0] = minsol(get(node,0))
  for i = 1:m
    tmp = minsol(get(node,i))
    u[i] = tmp - get(node,i-1)
  end
  return create_node(u)
end
```

構成する．いま，極小ベクトル集合を表す MDD では，終端 x ノードに到達するパスが状態 x に対する極小ベクトル集合の要素となるようにする．いま，$x_1 = s$ に固定して，残りの変数に対する極小ベクトル集合 V_s が求まったものとする．このとき $s = 0$ ならば，$(0, y)$, $y \in V_0$ は明らかに極小ベクトル集合となる．一方，V_i, V_j, $i < j$ に対して同じ要素が存在する場合，それは極小ベクトル集合ではない．そこで，V_j から $V_0, \ldots, V_{j-1}$ の要素を除いた集合 $V_j \setminus \cup_{k=0}^{j-1} V_k$ を求めることで，V_j の極小ベクトル集合を求めることができる．単調システムの場合 $V_j \setminus \cup_{k=0}^{j-1} V_k = V_j \setminus V_{j-1}$ となる．

Algorithm 3 は構造関数の MDD 表現から極小ベクトル集合を導出するアルゴリズム

である．関数 `get` はドメインの値に対応した枝に接続する下位のノードを取得する関数である．まず，0-枝に対して最小ベクトル集合を求めるアルゴリズムを実行し，その後，順に各枝に対する実行と集合差によって重複する集合を取り除く作業を繰り返す．最終的に作成された子ノード `u` を持つ新たなノードを `create_node` 関数で作成している．

4　まとめ

本章では，多状態システムに対する構造関数の MDD 表現とそれを用いた解析の紹介を行った．特に，極小ベクトル集合の導出アルゴリズムについて言及した．本アルゴリズムは特定の値（特定のシステム状態）に対する極小ベクトル集合だけでなく，全体の極小ベクトル集合を同時に求めるものであり，多状態システムの信頼性解析において重要な役割を果たす．今後は EvMDD (Edge-valued MDD) や MDD の高速化手法など，MDD を用いた多状態システムの信頼性解析手法のさらなる発展が期待される．

参考文献

[1] T. Akiba, H. Nagatsuka, H. Yamamoto, and M. J. Zuo, “Efficient analysis of multi-state k-out-of-n systems,” *Reliability Engineering & System Safety*, vol. 133, pp. 95–105, 2015.

[2] X. Song, X. Jia, and N. Chen, “Sensitivity analysis of multi-state social network system based on MDD method,” *IEEE Access*, vol. 7, pp. 167 714–167 725, 2019.

[3] A. Rauzy, “New algorithms for fault trees analysis,” *Reliability Engineering & System Safety*, vol. 40, no. 3, pp. 203–211, 1993.

[4] R. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.