A Static Analysis of Dynamic Fault Trees with Priority-AND Gates

Jianwen Xiang, Fumio Machida, Kumiko Tadano, Kazuo Yanoo, Wei Sun, Yoshiharu Maeno

Central Research Laboratories
NEC Corporation
Kawasaki, 211-8666 Japan

Email: {j-xiang@ah, f-machida@ab, k-tadano@bq, k-yanoo@ab, w-sun@ap, y-maeno@aj}.jp.nec.com

Abstract—A PAND gate is a special AND gate of Dynamic Fault Trees (DFTs) where the input events must occur in a specific order for the occurrence of its output event. We present a transformation from a PAND gate to an AND gate with some dependent conditioning events, called CAND gate, provided that the dynamic behavior of the system can be modeled by a (semi-)Markov process. With the transformation, a DFT with only static Boolean logic gates and PAND gates can be transformed into a static fault tree, which opens up the way to employ efficient combinatorial analysis for the DFT. In addition, the PAND gate cannot model the priority relations between the events whose occurrences are not necessary for the output event. The inability has not been addressed before and it can be overcome by the proposed CAND gate.

Keywords—fault tree; Markov property; Priority-AND; cut set; cut sequence

I. INTRODUCTION

Fault Tree Analysis (FTA) [1] is a traditional reliability and safety analysis technique. It is basically a deductive procedure for determining the various combinations of component failures that could result in the occurrence of an undesired Top Event (TE) at the system level. The applications of FTA are not limited to hardware systems, but also widely used in software (e.g., [2]–[4]) and integrated systems (e.g., [5], [6]).

Standard Boolean logic constructs, such as AND, OR, and Voting (k-out-of-n, k/n) gates, are used to decompose fault events and construct fault trees. This model is usually referred to as Static Fault Tree (SFT). SFTs assume that the interactions between fault events (including basic and intermediate ones) can be described in terms of Boolean logic gates, and thus only the smallest combinations of basic events (Minimal Cut Sets, MCSs) are relevant for the qualitative analysis of the Top Event (TE), not their sequences. In other words, SFTs do not consider potential (temporal) dependencies between fault events.

To remove this restriction, several dynamic gates have been proposed, such as Priority-AND (PAND), Functional Dependency (FDEP), Cold-Spare (CSP), and Sequential Enforcing (SEQ) gates. A fault tree with these dynamic gates is called a Dynamic Fault Tree (DFT) [7]. In this paper, we focus on a sub-class of DFTs, called Priority Fault Trees (PFTs), containing only static Boolean logic gates and dynamic PAND gates. A PAND gate is a special AND gate where the input events must occur in a specific order (typically from left to right) so as to result in the output event [1]. The PAND gate is used to represent the priority relation (sequential dependency) between the input events.

At the qualitative level, the MCSs play an important role in the analysis of a SFT, and they can be evaluated by applying the theorems of Boolean logic (e.g., [8]–[10]) or by resorting to Binary Decision Diagrams (BDDs) (e.g., [11]). In the case of PFTs, the concept of MCS must be refined to minimal cut sequence [12] representing the smallest sequence of primary component failures for the TE.

At the quantitative level, Markov analysis is usually carried out for DFTs including PAND gates (e.g., [7], [13], [14]). Unfortunately, the Markov analysis typically suffers from the state space explosion problem when the number of basic events increases. To solve the state space explosion problem, compositional approaches have been proposed, such as with input/output interactive Markov chains [15] and conversion from DFTs to Stochastic Petri Net by means of graph transformation [16]. Other approximation and/or exact solutions can be found in the literature such as [17]–[20].

A common obstacle in any quantitative analysis is the lack of accurate and reliable data on the failure distributions of the components. To overcome this deficiency, sometimes the qualitative analysis is the only valuable information on the system reliability. Nevertheless, the qualitative analysis of PFTs has not been well studied in most of the literature, especially the evaluation of minimal cut sequences. An alternative representation of PAND gate is presented in [1], in which the PAND gate is transformed into an AND gate with a conditioning event explicitly stating the temporal relation between the input events of the AND gate. Similar efforts by separating the logic AND part and the temporal (priority) part of the PAND gate can be found in [17] (by using INHIBIT-gate, a kind of AND gate with inhibit condition [1]) and [12]. However, they do not provide a method or algorithm to derive the minimal cut sequences of the PFT based on such transformations. The temporal semantics of PAND gate is formalized with Interval Logic (ITL) [21] in [3]. The formalization is used to derive safety requirements rather than minimal cut sequences from fault trees.

A notable exception is a most recent paper [20], in
which the proposed temporal algebraic framework for PFTs provides a sound theoretical basis for the determination of minimal cut sequences from a deduced sum-of-product canonical form of the TE. Each product term of the canonical form is called a Cut Sequence Set (CSS) consisting of basic fault events connected by Boolean AND and temporal BEFORE operators.

Even with the help of temporal logic and the CSS concept, the exhaustive search of all the minimal cut sequences from the CSSs could be very complex, especially when the number of components is not so small. This is because it is basically a permutation problem with factorial complexity. When the number of minimal cut sequences is huge (which could easily result from a not so large PFT), the quantitative analysis of the PFT based on the minimal cut sequences (e.g., [18]) also fails. In addition, two or more CSSs may overlap, i.e., they may contain repeating minimal cut sequences. These issues are not covered in [20].

Moreover, although the PAND gate provides a means to model the priority relations between a set of occurred input fault events, the priority relations between events whose occurrences are not required for the output event cannot be modeled by a PAND gate. In the logic sense, although a PAND gate can be always transformed into a set of CSSs, not every legitimate CSS can be modeled by a PAND gate (or a PFT). Such an inability of the PAND gate has not been addressed before, and we will review it later with a practical example.

This paper is a substantial extension of the work-in-progress reported in [22]. In this paper, we mainly focus on the qualitative analysis of PFTs. We notice an interesting phenomenon that, while Markov models are usually used for the quantitative analysis of PFTs, the fundamental Markov property is somehow overlooked in the qualitative analysis part. More specifically, if the dynamic behavior of the system satisfies the Markov property, the history should be irrelevant in determining the future (transitions), which in return somehow contradicts with the semantics of the PAND gate. Further investigation on some typical PAND applications (e.g., [7], [17]) discloses that such an “inconsistency” can be solved by identifying the “missing” dependent conditioning events in the original PAND gates.

With the proof of the existence of the dependent conditioning events, we propose a transformation from a dynamic PAND gate to a static AND gate with some dependent conditioning events, called a CAND gate. With the CAND transformation, we can reduce the permutation problem of a PFT into a combination problem of a SFT, in which the cost for the determination of the minimal cut sequences of the PFT can be saved. Moreover, the CAND gate can represent the priority relations that cannot be modeled by the PAND gate. The enhanced expressiveness is another advantage of the CAND gate.

The CAND solution is not limited to the systems in which the times to component failures are all exponentially distributed. Rather, the CAND solution can be applied to any system whose sequences of component failures to system failure can be modeled by a semi-Markov process (a generalization of Markov process) in which the times to component failures are not limited to exponential distributions.

This paper does not cover other dynamic gates of fault trees, such as functional dependency (FDEP), CSP (cold spare), and sequential enforcing (SEQ) gates [7]. Some solutions and discussions to these dynamic gates can be found in [19], [23], [24]. We have implemented the static analysis of PAND gate in an in-house tool, called CASSI (Computer Aided System model based SI environment) [6], [10]. Ongoing work is addressed to the possible static analysis of other dynamic gates and their implementations.

The rest of the paper is organized as follows. Section II presents the assumptions and some definitions as the preliminaries. Two typical PAND gate applications in fault-tolerant systems are presented in Section III. The formalization of PAND gate and its limitations are discussed in Section IV. The transformation from PAND to CAND gates and their comparison are presented in Section V, and a practical example of a fault-tolerant parallel processor configuration is used to demonstrate our method in Section VI. Finally, the concluding remarks are summarized in Section VII.

II. Preparation

A. Assumptions

1. The system is non-repairable.
2. The dynamic (failure) behavior of the system satisfies the (semi-)Markov property.
3. The basic events of fault trees (i.e., primary component failures) are s-independent.
4. Simultaneous occurrences of events are not considered (under the consideration of continuous failure time distribution [20]).

B. Definitions

Definition 1 (Conditioning Event). A conditioning event of fault tree refers to some specific conditions or restrictions that apply to any logic gate [1].

Like a basic event (typically a primary component failure [1]), a conditioning event is also a leaf (primary event [1]) of a fault tree. However, a conditioning event cannot be served as a direct input (fault event) of a logic gate such as an AND gate, because the conditioning event is not a fault but a “normal” event in general.

Unlike basic events which are typically assumed to be s-independent, the occurrence and non-occurrence (negation) of a conditioning event may depend on some specific situations, such as the failures of some components in some specific system states. For instance, the connection of a switch may change from a primary component to a spare
when the primary component fails and the switch is not failed. The connection to the primary component thus is a dependent conditioning event.

**Definition 2** (Markov Property). A stochastic process \( \{X(t); t \in T\} \) is said to possess Markov Property if for any \( t_0 < t_1 < \ldots < t_n < t_{n+1} \), the conditional distribution of \( X(t_{n+1}) \) for given values of \( X(t_0), \ldots, X(t_n) \), depends only on \( X(t_n) \) [25]:

\[
Pr\{X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n, \ldots, X(t_0) = x_0\} = Pr\{X(t_{n+1}) \leq x_{n+1} | X(t_n) = x_n\} \tag{1}
\]

The Markov property states that the next state to be visited just depends on the present state and not on the history, i.e., how the process arrived in the present state. A stochastic process with the Markov property is called a Markov process. A Markov process with a discrete state space is referred to as a Markov chain.

**Remark 1.** A distinction should be made between history and time. A Markov process may require the associated time to characterize the present state and is said to be non-homogeneous [26]. However, the (time dependent) non-homogeneous Markov process is still history independent. The time is different from the history, i.e., the sequence of past states and actions representing how the process arrived in the current state.

In some practical solutions, the Markov property may not be valid at all the time instants. A semi-Markov process is a generalization of a Markov process which moves from one state to another within a countable number of states. The successive states visited by the semi-Markov process forms a discrete-time Markov chain, where the Markov property is preserved at the state transition times and the distributions of sojourn times in states are relaxed to be general [25]. For the sake of convenience, we call the property possessed by a semi-Markov process as semi-Markov property.

**Definition 3** (Semi-Markov Property). A stochastic process \( \{Y(t); t \geq 0\} \) with a finite or countable set of states \( N = \{0, 1, 2, \ldots\} \), having stepwise trajectories with jumps (transitions) at times \( S_0 = 0 < S_1 < S_2 < \ldots \), is said to possess the Semi-Markov Property and is called a semi-Markov process if \( Y(S_n) \) satisfies the Markov Property and forms a Markov chain, i.e.,

\[
Pr\{Y(S_{n+1}) = j | Y(S_n) = i, \ldots, Y(S_0) = 0\} = Pr\{Y(S_{n+1}) = j | Y(S_n) = i\} \tag{2}
\]

In this paper we assume that that the dynamic (failure) behavior of the system satisfies (at least) the semi-Markov property (see Assumption (2)). With the assumption, the proposed CAND transformation of PAND gates can be applied to a wide range of applications including those that can be modeled by homogenous and non-homogenous Markov processes, and semi-Markov processes, and there is no specific restrictions on the distributions of the time to component failures.

**III. EXAMPLES OF PAND GATES**

In fault-tolerant systems, two typical applications of PAND gate are the switch control problem between a principal and a standby unit [17] and the competition of shared spares between primary components [7].

Fig. 1a shows the configuration of an electrical supply system with a switch control between a principal and a standby power supply. The switch is originally connected to the principal and will instantaneously connect to the standby when the principal fails. The system fails if 1) the principal and the standby both fail or 2) the switch fails first, and then the principal fails. In the second case, the standby cannot be used because the switch has failed before the failure of the principal. The fault tree with a PAND gate representing the sequence of the switch failure and then the principal failure is illustrated in Fig. 1b.

![Figure 1: Power supply system with standby unit](image-url)
the latter case, the spare cannot be used for PrimaryA since it has been used for PrimaryB. The fault tree of SystemA with a PAND gate representing the sequence of failures of first PrimaryB and then PrimaryA is presented in Fig. 2b.

Notice that in Figure 2, if we consider the failure of the whole system consisting of Spare, PrimaryA, and PrimaryB, and assume that the whole system fails if either PrimaryA or PrimaryB is down, then a PFT including two PAND gates can be constructed. Because the priority relations of the two PAND gates are “antisymmetric”, the PFT can be simplified into a SFT with only a 2-out-of-3 (voting) gate with inputs of the failures of the three components (PrimaryA, PrimaryB, and Spare), i.e., the system fails if any two of the three fail. Such a simplification, however, may not be applicable in some other cases due to different configurations between primaries and spares (e.g., [7]).

Notice that in Figure 2, if we consider the failure of the whole system consisting of Spare, PrimaryA, and PrimaryB, and assume that the whole system fails if either PrimaryA or PrimaryB is down, then a PFT including two PAND gates can be constructed. Because the priority relations of the two PAND gates are “antisymmetric”, the PFT can be simplified into a SFT with only a 2-out-of-3 (voting) gate with inputs of the failures of the three components (PrimaryA, PrimaryB, and Spare), i.e., the system fails if any two of the three fail. Such a simplification, however, may not be applicable in some other cases due to different configurations between primaries and spares (e.g., [7]).

![Fault Tree](image)

**Figure 2:** Competition of shared spare

IV. **FORMALIZATION AND LIMITATIONS OF PAND GATE**

For the purposes of identifying the problems of traditional PAND solutions and demonstrating our CAND transformation, we borrow the temporal semantics of PFTs defined in [20].

To model the temporal order relations, the events of FTs are considered as temporal functions, which are piecewise right-continuous on \( \mathbb{R}^+ \cup \{+\infty\} \) (we assume continuous time hereafter), and whose range is \( \mathbb{B} = \{0(false), 1(true)\} \). Let the set of such temporal events be \( \mathcal{E} \), the Boolean AND (\( \cdot \)) and OR (\( + \)) operators can be extended to \( \mathcal{E} \) with identity elements \( \bot \) and \( \top \), equivalent to 0 and 1 respectively. The occurrence time (date) of an event \( A \) is denoted by a function \( d(A) \) with:

\[
d(\bot) = +\infty, \quad d(\top) = 0
\]

A temporal non-inclusive operator BEFORE (BF, \(<\)) is introduced to represent the order (priority) relation between events. Let \( A \) and \( B \) be events, the formal semantics of \( A < B \) is defined below:

\[
A < B = \begin{cases} A & \text{if } d(A) < d(B) \\ \bot & \text{if } d(A) > d(B) \end{cases}
\]

For clarity, we omit the case of simultaneous occurrence based on the assumption of continuous time space in which the simultaneity can be neglected [20].

With the help of the temporal BF operator, it is now possible to derive the structure function of a PFT. For instance, let \( T, A, B, S \) be the top event and the failures of PrimaryA, PrimaryB, and Spare, respectively, the PFT of Figure 2b can be interpreted as:

\[
T = A \cdot S + B \cdot A \cdot (B < A)
\]

The structure function of a PFT can be reduced into a sum-of-product canonical form with the help of a set of theorems [20]. Let \( b_1, \ldots, b_n \) be basic events, a product term \( \prod b_i \prod (b_j < b_k) \) is called a Cut Sequence Set (CSS) [20]. A CSS is not a single cut sequence, but a temporal formula providing a sufficient condition on the order (priority) of basic events that leads to the TE which may contain a set of cut sequences. The determination of the included cut sequences from a CSS is generally done by exhaustive search of the cut sequences satisfying the order relation specified by the CSS from all the possible cut sequences. This is basically a permutation problem which could be very complex when the number of basic events is not so small.

One reason for the complexity of the evaluation of cut sequences is that, unlike the case of cut sets where redundant cut sets can be identified and removed in terms of subset relation, two cut sequences with different lengths are always mutually exclusive even if subset relation exist between the sets of their elements and the order of the shorter one is kept in the longer one. For instance, given a CSS \( b_1 \cdot b_2 \cdot (b_1 < b_2) \cdot (b_2 < b_3) \) (and assume that the PFT consists of only the three basic events \( b_1, b_2, \) and \( b_3 \)), the two eligible cut sequences \( [b_1, b_2] \) and \( [b_1, b_2, b_3] \) are mutually exclusive. This is because the shorter one \([b_1, b_2] \) implies the non-occurrence of \( b_3 \), and its equivalent logic expression is \( b_1 \cdot b_2 \cdot \neg b_3 \cdot (b_1 \cdot b_2) \).

Notice that \( b_1 \cdot (b_1 < b_j) \) (\( i \neq j \)) can be reduced into \( b_1 < b_j \) because \( b_i < b_j \rightarrow b_i \) [20]. In our discussion we keep \( b_i \) for better readability such that a CSS can be divided into two parts, one is the set of required basic events, and the other is set of order relations between the required basic events and possibly other basic events not specified in the first part.

Notice that \([b_1, b_2] \) is a shortened notation for a more correct expression \([b_1, b_2, \neg b_3] \) [20].
Another trouble of the evaluation of minimal cut sequences is that, two or more CSSs may contain repeated minimal cut sequences even when all the CSSs are minimal in the sense that CSS\textsubscript{i} \cdot CSS\textsubscript{j} \neq CSS\textsubscript{i} (i \neq j). In other words, the concept of minimal CSS just guarantees that no CSS is included in another CSS, but it does not guarantee that there is no overlap between two minimal CSSs in terms of their included minimal cut sequences.

In addition to the potential computation complexity of minimal cut sequences, the semantics of PAND gate imposes a constraint on the occurrences of the ordered events. Given a PAND gate with \( m \) input events \( E_1, \ldots, E_m \) and an output event \( T \), its semantics can be interpreted as:

\[
T = \prod_{i=1}^{m} E_i \cdot \prod_{j=1}^{m-1} (E_j \preceq E_{j+1}) \tag{4}
\]

Obviously, not every temporal product term connected by \( \cdot \) and \( \preceq \) can be represented by a PAND gate. This is different from the case of SFTs, in which any Boolean product term can be represented by an AND gate, and vice versa. The constraint of PAND gate makes it impossible to model some practical priority relations in which the occurrences of some ordered events are not required for the occurrence of the output event. For instance, return back to the shared spare example (Figure 2) and assume that there is another primary, PrimaryC, also supported by the Spare. Let \( C \) denote the failure of PrimaryC, the failure of SystemA then can be interpreted by the following temporal formula:

\[
T = A \cdot S + B \cdot A \cdot (B \preceq A) \cdot (B \preceq C) + C \cdot A \cdot (C \preceq A) \cdot (C \preceq B) \tag{5}
\]

In contrast to Eq. (3) whose corresponding PFT is given in Figure 2b, the corresponding PFT of Eq. (5) cannot be directly constructed. This is because the last temporal product terms of Eq. (5) cannot be represented by PAND gates directly. For instance, the last product term states that if \( C \) occurs first (before the occurrences of \( A \) and \( B \)), and then the occurrence of \( A \) will cause the occurrence of the TE \( T \) (because the shared Spare has been used for the replacement of PrimaryC and thus cannot be used for PrimaryA afterwards anymore). In this case, the occurrence of \( B \) is not required, and the only constraint of \( B \) is that it cannot occur before \( C \).

The inability of PAND gate thus somewhat restricts its applications in practice as demonstrated above, and this is another important motivation for us to propose an alternative combinatorial approach to overcome the inability.

V. PAND VS. CAND

A. Transformation from PAND to CAND

**Theorem 1.** Given a PAND gate and assuming that the dynamic behavior of the system (to reach system failure) satisfies (at least) the semi-Markov property, the PAND gate can be transformed into an AND gate with some additional conditioning events.

To be simple, the proof of Theorem 1 can be done in terms of the semi-Markov property, i.e., the occurrence of the output event of the PAND gate should be independent from the specific sequences of the occurrences of its input fault events. To prove Theorem 1, we first prove the existence of the conditioning event in a PAND gate with two input events [27], and then consider the PAND gate with more than two input events. Two lemmas are introduced below correspondingly.

**Lemma 1.** Given a PAND gate with an output event \( T \) and two input events \( E_1 \) and \( E_2 \) with \( T = E_1 \cdot E_2 \cdot (E_1 \preceq E_2) \), and assuming that the dynamic behavior of the system satisfies the semi-Markov property, then there must exist some dependent conditioning event, \( C \), which must be and can only be activated in the state where \( E_1 \cdot \neg E_2 \) (in the acceptable path) if it is initially false, or negated in the state where \( E_2 \cdot \neg E_1 \) (in the unacceptable path) if it is initially true.

**Proof Sketch:** The proof can be done by contradiction.

We first assume that \( C \) does not exist and a system state consists of the observations of the occurrences of only three events, namely \( E_1, E_2, \) and \( T \). With this assumption, both of the (left) acceptable path \( ([E_1, E_2]) \) and the (right) unacceptable path \( ([E_2, E_1]) \) of the two input events can reach the same state 110 (representing \( E_1 \cdot E_2 \cdot \neg T \) accounting for the occurrence of the top event as shown by Fig. 3a. Notice that the transition from the state 110 to 111 should only depend on 110 but not the history to 110 as stated by the semi-Markov property. Here, a contradiction arises in terms of the semantics of the PAND gate, i.e., only the acceptable path can result in the output event.

With the above contradiction, the existence of the additional event \( C \) can be proved. In addition, \( C \) must occur (or not occur) in only one sequence, otherwise the same contradiction occurs even with the introduction of \( C \). To this end, \( C \) is a dependent conditioning event rather than a fault event, because we assume that all the component failures are \( s \)-independent.

Unlike a fault event which is assumed to be false in the initial system state in FTA, a conditioning event could be initially true or false in different applications. If \( C \) is initially false, then it must only be activated in the state where \( E_1 \cdot \neg E_2 \) holds (i.e., \( E_1 \) just occurred under the non-occurrence of \( E_2 \)); If it is initially true, then it must only be negated in the state where \( \neg E_1 \cdot E_2 \) holds. Note that it is impossible for \( C \) to occur or be negated in the state with \( E_1 \cdot E_2 \), since such a state can be reached by both acceptable and unacceptable sequences of the two inputs.

The revised state transition diagram with the dependent
conditioning event $C$ for the case that it is initially false and is activated in the state 1000 is presented in Fig. 3b.

![State transition diagrams of PAND and CAND](image)

Figure 3: State transition diagrams of PAND and CAND

The equivalence between a CAND gate and its corresponding PAND gate can also be proved with the help of temporal logic. For instance, as for the case where $C$ is initially false as shown in Figure 3b, $C$ must (and can) only be activated in the state where $E_1 \cdot \neg E_2$ (and holds afterwards regardless of the subsequent occurrence of $E_2$). By considering the occurrence of $C$ as an instantaneous transition, we can derive $C = E_1 \prec E_2$, and then

$$E_1 \cdot E_2 \cdot C = E_1 \cdot E_2 \cdot (E_1 \prec E_2) \quad (6)$$

Similarly, if $C$ is initially true, then it must only be negated if $E_2$ occurs before $E1$, i.e., $C = \neg (E_2 \prec E_1) = (\neg E_1 \cdot \neg E_2) + (E_1 \prec E_2)$, and then

$$E_1 \cdot E_2 \cdot C = E_1 \cdot E_2 \cdot ((\neg E_1 \cdot \neg E_2) + (E_1 \prec E_2)) = E_1 \cdot E_2 \cdot (E_1 \prec E_2) \quad (7)$$

It is interesting to observe that while Markov models are widely used in the analysis of DFTs including PAND gates (e.g., [13]), the potential contradiction between the semantics of PAND gate and the fundamental (semi-)Markov property has seldom been noticed in such analysis. Lemma 1 removes the contradiction and opens up the way to employ combinatorial analysis for the DFTs.

Based on Lemma 1, we now review the two typical PAND applications of Section I and discuss how they can be transformed into static CAND gates. With respect to the switch example (Fig. 1b), the key issue is that if the principal supply fails under the non-occurrences of the switch failure, the connection of the switch will be instantaneously changed from the principal supply to the standby supply. But if the switch fails before the failure of the principal supply, it will be stuck to the principal supply regardless of the subsequent failure of the principal supply. Here, the dependent conditioning event can be defined as that the switch is connected to the principal supply which is initially true. Fig. 4 presents the corresponding CAND tree of the switch example.

![CAND tree of switch example](image)

Figure 4: CAND tree of switch example

With respect to the shared spare example (Fig. 2b), the point is that the spare will respond and replace only one of the primary components who first fails. In the case that PrimaryB fails first, the spare will replace PrimaryB and will not respond to the failure of PrimaryA anymore afterwards. Here, the dependent conditioning event could be defined as that the spare replaces PrimaryB which is initially false (assuming that the spare initially replaces nothing). Fig. 5 presents the corresponding CAND tree.

![CAND tree of shared spare example](image)

Figure 5: CAND tree of shared spare example

With the transformations of the above two example PAND applications, a critical issue has been disclosed, that is, whether the occurrence of the output event of a PAND gate is directly and solely controlled by the order of its input events. If the answer is YES, then there must exist some order control component in the system, which is actually not the case in the above two PAND applications. In contrast, the fundamental mechanical reason for the occurrence of the output events of the two example PAND gates is the combinations of the conditioning and input events. The order of the occurrences of the input events could be regarded as a kind of "syntactic sugar" for the "missing" conditioning event in such cases. However, the order solution (i.e., the PAND gate) may impose considerable cost on the determination of minimal cut sequences afterwards.

Lemma 1 only deals with the PAND gate with two inputs. As for a PAND gate with more than two inputs, another lemma can be introduced to transform the PAND gate into...
Lemma 2. Any PAND gate with more than two input events can be transformed into a cascade of PAND gates each with two input events, or a conjunction of a set of consecutive PAND gates each with two input events.

Proof Sketch: The equivalent cascaded and consecutive PAND trees of a PAND gate with three inputs is presented in Fig. 6. The proof of Lemma 2 can be carried out in terms of the temporal semantics of the PAND gate. For instance, the equivalence between the second and the first PAND gate of Fig. 6 can be proved below:

\[(E_1 \cdot E_2 \cdot (E_1 < E_2)) \cdot E_3 = (E_1 \cdot E_2 \cdot (E_1 < E_2) < E_3)\]

By combining Lemma 1 and 2, we can prove Theorem 1, and thus our static transformation can be applied to any PAND gate with any number of input events.

B. Expressiveness of CAND over PAND

As discussed in Section IV, the semantics of PAND gate (Eq. (4)) imposes a restriction that the order (priority) relation must be defined between the occurred (input) events (whose occurrences are required for the occurrence of the output event). This makes it impossible to express the order relations involving non-occurred events (whose occurrences are not required) with PAND gates, such as the last two CSSs of Eq. (5).

In contrast, the CAND gate can be used to cover such inability of PAND gate, provided that the system satisfies the (semi-)Markov property. For instance, as for the shared spare example among 3 primary components as interpreted by Eq. 5, let \(R_1 = rep(S, B)\) and \(R_2 = rep(S, C)\) be two dependent conditioning events representing the spare \(S\) replaces the primary \(B\) and \(C\), respectively. Notice that \(R_1\) can only occur if \(B\) fails first among the 3 primary components, likewise \(R_2\) can only occur if \(C\) fails first. The static fault tree with CAND gates corresponding to Eq. (5) can be interpreted by the following Boolean formula:

\[T = A \cdot S + B \cdot A \cdot R_1 + C \cdot A \cdot R_2\]  \(\text{(8)}\)

C. Reduction of MCSs of SFTs with CAND gates

By applying the CAND transformation, a PFT can be transformed into an equivalent static fault tree with CAND gates (simply called CFT). Unlike the PFT, the qualitative analysis of the CFT no more relies on the concept of minimal cut sequence, but MCS of traditional SFT. Existing efficient algorithms for the evaluation of MCSs of SFTs (e.g., [9], [10], [28], [29]) thus can be applied to CFTs, with a little more effort on the handling of inconsistent and redundant dependent conditioning events.

In SFTs, a MCS usually consists of only basic events (typically primary component failures [11]) which are assumed to be independent. The conditioning events, if any and included in a MCS, are also considered as independent in general for simplification. In the case of CFTs, the basic events are still assumed to be independent, but the conditioning events attached to the CAND gates are actually dependent events. A product of two or more dependent conditioning events could become inconsistent or redundant if the negation of an event or itself is implied by the product of other events.

In some cases, the inconsistency and redundancy are apparent such that they can be derived out directly without resorting to the temporal definitions of the dependent conditioning events. For instance, in the shared spare example, the inconsistency between dependent rep conditioning events can be concluded as that a spare cannot replace two primaries at the same time, likewise primary cannot be replaced by two spares. The replacement of a primary cannot be implied by other replacements, and thus there is no redundancy problem. Let \(P_1\) and \(P_2\) be distinct primaries and \(S_1\) and \(S_2\) be distinct spares, the following Boolean reduction rules can be introduced to solve the inconsistency problem:

\[\text{rep}(S_1, P_1) \cdot \text{rep}(S_1, P_2) = 0\]  \(\text{(9)}\)

\[\text{rep}(S_1, P_1) \cdot \text{rep}(S_2, P_1) = 0\]  \(\text{(10)}\)

In some other cases, we may have to resort to the temporal definitions of the dependent conditioning events. Although a conditioning event could be true in the initial system state, the initially true case (represented by the non-occurrences of all the components, i.e., \(\prod_{i=1}^{n} \neg b_i\)) can be removed from the temporal definition of the conditioning event in the reduction of MCSs of the fault tree, if we assume that the system and all the components are initially not failed. This is because any cut set of the fault tree must include at least one basic fault event, say \(b_j\) \((j \in [1..n])\), and thus no cut set will include the initially true case simply due to \(\prod_{i=1}^{n} \neg b_i \cdot b_j = 0\) as demonstrated earlier in Eq. (7). In other words, given a dependent conditioning event, only the encoded priority relation preserving its occurrence is necessary for the evaluation of the MCSs of the CFT.

Let \(b_i\) \((i = 1, \ldots, n)\) be independent basic fault events, and let \(c\) be a dependent conditioning event, the priority
The condition (if) parts of the above reduction rules depend on the temporal semantics of the dependent conditioning events, and thus in these cases the evaluation of MCSs of CFTs is similar to the evaluation of CSSs of the corresponding PFTs in terms of the set of temporal theorems [20]. Notice even in this case, the space complexity of the MCSs is generally (much) less than that of the CSSs, because a dependent conditioning event encoding a set of priority relations is required to be decomposed in the evaluation of the CSSs by applying the temporal distribution theorems such as \((b_1 + b_2) \leq b_3 = (b_1 \not\leq b_3) + (b_2 \not\leq b_3)\).

VI. FAULT-TOLERANT PARALLEL PROCESSOR EXAMPLE

To further demonstrate the proposed CAND gate and compare it with the traditional PAND gate, a Fault-tolerant Parallel Processor (FTPP) [30], [31] configuration is used. The FTPP configuration has been originally used to demonstrate the applications of DFTs in [7].

The configuration (Figure 7) consists of 16 Processing Elements (PEs), with 4 connected to each of 4 Network Elements (NEs). The NEs are fully connected. The 12 primary PEs are logically connected to form 4 triads \((T1, T2, T3, \text{ and } T4)\). For instance, \(T11, T12 \text{ and } T13\) form the triad \(T1\). The 4 spare PEs \((TS1, TS2, TS3, \text{ and } TS4)\) are distributed across the NEs, and the spare on each NE can support and replace for any failed PE connected to the same NE. For instance, \(TS1\) can replace \(T11, T22, \text{ or } T33\). The redundancy of each triad is set to 1, i.e., if 2-out-of-3 primary PEs are down, then the triad is failed. For simplification, we do not consider the failures of NEs. The system is failed if any of the triads is failed.

Let \(D11\) stands for the down event of \(T11\), an incomplete PFT of the configuration is presented in Figure 8. The temporal structure function of the (sub) PFT with root \(D11\) can be derived as:

\[
D11 = T11 \cdot TS1 + (T22 + T33) \cdot T11 \cdot ((T22 + T33) < T11) \\
= T11 \cdot TS1 \\
+ (T22 + T33) \cdot T11 \cdot ((T22 < T11) + (T33 < T11)) \\
= T11 \cdot TS1 \\
+ T22 \cdot T11 \cdot (T22 < T11) + T22 \cdot T11 \cdot (T33 < T11) \\
+ T33 \cdot T11 \cdot (T22 < T11) + T33 \cdot T11 \cdot (T33 < T11) \\
\]

\[
D11 = T11 \cdot TS1 \\
+ T22 \cdot T11 \cdot (T22 < T11) + T33 \cdot T11 \cdot (T33 < T11) \\
\]

where the last step reduction \((\equiv)\) is carried out in terms of the minimization algorithm of CSSs of [20].

Referring to Eq. (8), and let \(R22 = rep(TS1, T22)\) and \(R33 = rep(TS1, T33)\) be two dependent conditioning events representing that the spare \(TS1\) replaces \(T22\) and \(T33\), respectively, the Boolean structure function of the corresponding CFT of \(D11\) is:

\[
D11 = T11 \cdot TS1 + T22 \cdot T11 \cdot R22 + T33 \cdot T11 \cdot R33 \\
\]

According to Eq. (5), the equivalent temporal definition of the CFT of Eq. (14) is given below:

\[
D11 = T11 \cdot TS1 \\
+ T22 \cdot T11 \cdot ((T22 < T11) \cdot (T22 < T33)) \\
+ T33 \cdot T11 \cdot ((T33 < T11) \cdot (T33 < T22)) \\
\]

Some remarks about the comparison between the PFT and the CFT are made below.

**Remark 2.** Even without considering NE failures, the number of candidate minimal cut sequences for a given CSS of \(T\) is up to \(|e \times 16!|\). The PFT solution thus could impose heavy constraints in practice even when the scale of the system is not so large.
Remark 3. The reduction of the three MCSs of the CFT (Eq. (14)) can be carried out without referring to the temporal semantics of the dependent conditioning events, i.e., they can be calculated based on standard Boolean theorems together with the introduced reductions rules for solving inconsistent dependent conditioning events (Eq. (9–10)).

Remark 4. The reduction of the CSSs of the PFT (Eq. (13)) is more complex than the reduction of the MCSs of the CFT (Eq. (15)). Moreover, the last two CSSs are overlapped although neither of them is redundant. For instance, [T33, T22, T11] is a minimal cut sequence satisfying both T22 · T11 · (T22 < T11) and T33 · T11 · (T33 < T11).

Remark 5. The structure function of the PFT (Eq. (13)) and the temporal definition of the CFT (Eq. (15)) are equivalent although the proof (transformation) is not so apparent. Direct representation of Eq. (15) with a PFT is impossible because of the inability of the PAND gate. Therefore, Eq. (13) could be regarded as a “workaround” of Eq. (15) for the PFT representation. On one hand, the workaround may introduce some semantic confusions, such as that it may be difficult to interpret the causality between a minimal cut sequence of a CSS and the top event in terms of the priority relation of the CSS: On the other hand, such a workaround may not be available in some other cases where CFTs are the only solutions.

To illustrate the potential semantic confusions introduced by the PFT (see Eq. (13)), let us consider the CSS S1 = T22 · T11 · (T22 < T11). [T33, T22, T11] is a minimal cut sequence satisfying S1. However, the direct reason for [T33, T22, T11] resulting in D11 is not that T22 and T11 occur in sequence (i.e., the priority relation stated by S1), but the MCS M1 = T33 · T11 · R33 or the corresponding priority relation, T33 · T11 · ((T33 < T11) · (T33 < T22)), encoded by M1 of the CFT (see Eq. (14) and (15)).

VII. Conclusion

In this paper, we have presented a transformation from traditional dynamic PAND gates to static CAND gates, provided that the dynamic behavior (state change) of the system (to reach system failure) satisfies the semi-Markov property (in which the time to component failure can be generally distributed). With the CAND transformation, the qualitative analysis of a PFT can be reduced from a permutation problem into a combinatorial problem without resorting to the concept of minimal cut sequence. Moreover, the CAND gate can model the priorities relations between the events whose occurrences are not required (for the output event), in which the PAND gate may fail to model (directly). In addition to the theoretical contributions, the practical significance of the CAND solution is also important because it can remove the constraint of history dependency from the development (implementation) of the (fault-tolerant) system. For instance, the (automatic) recovery and reconfiguration mechanisms of the system can now rely on the current status of components rather than on the history of component failures.

ACKNOWLEDGMENT

The authors thank Prof. Guillaume Merle of Beihang University for insightful discussions on dynamic fault trees. The constructive and valuable comments of the anonymous reviewers are also gratefully acknowledged.

REFERENCES


