# On Qualitative Analysis of Fault Trees Using Structurally Persistent Nets

Ricardo J. Rodríguez *Member, IEEE*

***Abstract*—A Fault Tree (FT) defines an undesired top event, characterizing it using logic combinations of lower-level undesired events. In this paper, we focus on coherent FTs, i.e., the logic is restricted to AND/OR formulae. Fault Tree analysis is used to identify and assess the Minimal Cut Sets (MCS) of an FT, which define the minimal set of events leading to the undesired state. The dual of MCS is Minimal Path Set (MPS). MCS and MPS are commonly used for qualitative evaluation of FTs in safety and reliability engineering. This paper explores computation of the MCS/MPS of an FT by means of structural analysis (namely, computation of minimal p-semiflows) of a Petri net that represents the FT. To this end, we propose a formal definition of a coherent FT and a transformation from this model to a Petri net subclass (namely, structurally persistent nets). We also prove the relationship between minimal p-semiflows and MCS/MPS in an FT. In addition, we propose an algorithm that uses linear programming techniques to compute the MCS/MPS in an FT. Finally, we put our findings into practice by qualitatively evaluating the FT of a pressure tank system.**

***Index Terms*—Fault Trees, Petri nets, Linear Programming, qualitative evaluation, cut sets**

## ACRONYMS

| | |
|---|---|
| **FT** | Fault Tree |
| **FTA** | Fault Tree Analysis |
| **MCS** | Minimal Cut Set(s) |
| **MPS** | Minimal Path Set(s) |
| **TE** | Top Event |
| **LP** | Linear Programming |
| **LPP** | Linear Programming Problem |
| **PN** | Petri net |

## I. INTRODUCTION

A FAULT Tree (FT) of a system represents an event-driven failure logic that describes how an undesired state of the system is reached, located at the root of the tree – termed as the Top Event (TE). In this paper, we focus on *coherent* Fault Trees, where the failure logic is restricted to the use of AND/OR gates [1].

A Fault Tree Analysis (FTA) [2] identifies the event combinations that lead to an undesired state. FTA is a top-down deductive analysis technique developed in the early 1960s and commonly used in safety and reliability engineering. An FTA obtains two different sets of events: the set of events whose occurrence leads to a system failure (*cut sets*); and its dual, i.e., the set of events whose non-occurrence guarantees that the system does not reach the failure state represented by its TE (*path sets*). These sets are named *Minimal Cut Sets* (MCS) and *Minimal Path Sets* (MPS), respectively[1], when they cannot be further reduced.

MCS and MPS are used to qualitatively evaluate an FT in a safety and reliability analysis [3], [4]. The goal of safety analysis is to ensure that an undesired state is not reached, or that there is a very low probability of its being reached. A complete review (unfortunately not updated) of FT analysis, methods, and applications can be found in [5]. In summary, a system safety assessment must compute the MCS and MPS of an FT once undesired events and their underlying logic have been defined. However, this task incurs high time and memory costs. In this paper, we focus on how computation of the MCS and MPS of coherent Fault Trees can be improved. In particular, the aim of this work was to develop an efficient algorithm to compute these minimal sets.

To fulfill this goal, we first propose the transformation of a formally defined, coherent Fault Tree into structurally persistent nets, a subclass of Petri nets. Then, we prove some net properties in the obtained net. We also state the relationship between its minimal p-semiflows and the MCS and MPS of a coherent FT. Therefore, we convert the problem of computing the MCS/MPS a coherent FT to the computation of minimal p-semiflows in a bounded structurally persistent net. Finally, we propose an algorithm that uses Linear Programming (LP) techniques to compute the minimal p-semiflows of the Petri net obtained by model transformation. Thus, the computation of the MCS/MPS of a coherent FT can be performed in polynomial time. We asses in evidence our findings by qualitatively evaluating an FT of a pressure tank system.

This paper is organized as follows. Section II briefly reviews related work. Section III provides the background required to understand the rest of the paper. Then, Section IV introduces the model transformation to obtain a structurally persistent net from an FT. Section V proves the relationship between p-semiflows and the MCS/MPS of an FT, and introduces an algorithm that uses LP techniques to compute the MCS/MPS of a given FT. Then, in Section VI we evaluate our algorithm in an FT of a pressure tank system. Finally, Section VII concludes the paper and states some future research plans.

## II. RELATED WORK

Computation of the MCS/MPS of an FT entails high time and memory costs, given that logic equations can produce

Ricardo J. Rodríguez is with the Research Institute of Applied Sciences in Cybersecurity, University of León, Spain (e-mail: rj.rodriguez@unileon.es).

[1]In this paper, we use MCS and MPS interchangeably as singular and plural acronyms.

many cut/path sets. Indeed, their computation is generally an NP-hard problem [6]. Some approaches have attempted to produce the MCS most likely to occur (called *culling* [7]) or to compute the FT modules (i.e., a subtree whose terminal events do not appear anywhere else) [8]. MCS/MPS computation is also based on Bryant's Binary Decision Diagrams (BDD) [6], [9], [10], where a conversion from FT to a BDD is performed to overcome these computation limitations. However, the main drawback is the effort required to perform such a conversion. Furthermore, BDD computation may fail and does not avoid the exponential problem [11]. In [12], a BDD-based algorithm is presented for large coherent FT that uses truncating and modularizing. Other works use different formal models for reliability analysis in large cases, such as Multivalued Decision Diagrams [13] or Dynamic Bayesian Networks [14].

There are two main approaches to MCS and MPS computation, divided into top-down and bottom-up approaches [15]. One of the best-known top-down methods is MOCUS [16], [17], outperformed by CARA [18]. In [19], Garriba et al. introduce DICOMICS, an algorithm that provides an efficient construction of MCS for large fault trees and whose time complexity depends on the logic structure of the tree. Some other well-known methods are FATRAM [20] (also top-down) and MICSUP [21] (bottom-up). An extensive review of MCS/MPS algorithms can be found in [2]. Note that all these algorithms can be applied to coherent FTs. Petri nets (PNs) have also been used for MCS/MPS computation, as reported in [22], where a fault tree is transformed into a reverse PN, and its reachability graph is used to obtain the MCS of the FT. However, the problem of determining the reachability graph of a PN is NP-hard and has exponential space requirements [23].

Petri nets are a formal modeling language widely used in the industry for analyzing large complex systems, such as manufacturing systems [24]–[27] and intelligent mobile robots [28]–[30]. The safety of such systems is usually a system requirement, and the use of Petri nets to analyze a fault tree of the system has also been explored in the safety research community. Some of these are related to timing analysis in safety-critical systems by exploring the reachability state space [1], [31], [32] or the reachability markings [33], [34]. Other works focus on computation of the underlying Markov Chain [35] or on the steady-state simulation analysis [36], [37]. The work reported in [38] presents an algorithm to compute the MCS of a coherent FT by means of transforming it into a Petri net, and computing its reachability set. In [30], system reliability is analyzed using reliability block diagrams instead of fault trees, and a conversion algorithm to colored Petri nets [39] is also presented. Here, we use structural analysis of PNs (namely, minimal p-semiflows) to qualitatively evaluate a fault tree (i.e., to compute the MCS and MPS). The most closely related work is [40], where p-semiflows are used to obtain the frequency of each end-state of an event tree with dependencies.

The qualitative analysis of fault trees has received wide research attention [6], [7], [9], [41]–[45]. However, unlike our work, none of these makes use of linear programming techniques or provides an algorithm to the compute MCS/MPS of a coherent FT in polynomial time. In contrast, Linear Programming techniques have been applied to Petri net models for computing system performance by means of throughput bounds [25], [46], [47].

With respect to the aforementioned works, the contribution of this paper is twofold. First, we define a model transformation from coherent Fault Trees to Petri nets (namely, a subclass known as structurally persistent nets). Second, we prove that the minimal p-semiflows of these nets have a correspondence with the cut sets of the fault tree. Given that p-semiflows can be computed using linear programming techniques, we also provide an algorithm to efficiently compute the MCS and MPS of a coherent fault tree.

## III. Definitions

This section details some of the definitions required to understand the rest of the paper. First, we introduce some concepts related to Petri nets. We assume that the reader is familiar with the fundamentals of a Petri net, such as its structure and firing rules. A gentle introduction to Petri nets is given in [48]. Lastly, we introduce fault trees and formally define a coherent fault tree. The reader is referred to [3] for a gentle introduction to Fault Trees.

### A. Petri Nets

**Definition 1.** *A Petri net [48] (PN) is a 4–tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post}\rangle$, where:*

- *$P$ and $T$ are disjoint non-empty sets of* places *and* transitions *($|P| = n$, $|T| = m$) and*
- *$\mathbf{Pre}$ ($\mathbf{Post}$) are the pre–(post–)incidence non-negative integer matrices of size $|P| \times |T|$.*

The *pre-* and *post-sets* of a node $v \in P \cup T$ are respectively defined as ${}^\bullet v = \{u \in P \cup T | (u, v) \in F\}$ and $v^\bullet = \{u \in P \cup T | (v, u) \in F\}$, where $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs. A Petri net is said to be *self-loop free* if $\forall p \in P, t \in T$ $t \in {}^\bullet p$ implies $t \notin p^\bullet$. The *incidence matrix* of a Petri net is defined as $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$.

A vector $\mathbf{m} \in \mathbb{Z}_{\geq 0}^{|P|}$ which assigns a non-negative integer to each place is called a *marking vector* or *marking*.

**Definition 2.** *A Petri net system, or marked Petri net $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$, is a Petri net $\mathcal{N}$ with an initial marking $\mathbf{m_0}$.*

A transition $t \in T$ is *enabled* at marking $\mathbf{m}$ if $\mathbf{m} \geq \mathbf{Pre}(\cdot, t)$, where $\mathbf{Pre}(\cdot, t)$ is the column of $\mathbf{Pre}$ corresponding to transition $t$. A transition $t$ enabled at $\mathbf{m}$ can *fire*, yielding a new marking $\mathbf{m}' = \mathbf{m} + \mathbf{C}(\cdot, t)$ (*reached* marking). This is denoted by $\mathbf{m} \xrightarrow{t} \mathbf{m}'$. A sequence of transitions $\sigma = \{t_i\}_{i=1}^n$ is a *firing sequence* in $\mathcal{S}$ if there exists a sequence of markings such that $\mathbf{m_0} \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \mathbf{m}_2 \ldots \xrightarrow{t_n} \mathbf{m}_n$. In this case, marking $\mathbf{m}_n$ is said to be *reachable* from $\mathbf{m_0}$ by firing $\sigma$, and this is denoted by $\mathbf{m}_0 \xrightarrow{\sigma} \mathbf{m}_n$. The *firing count vector* $\boldsymbol{\sigma} \in \mathbb{Z}_{\geq 0}^{|T|}$ of the firable sequence $\sigma$ is a vector such that $\boldsymbol{\sigma}(t)$ represents the number of occurrences of $t \in T$ in $\sigma$. If $\mathbf{m_0} \xrightarrow{\sigma} \mathbf{m}$, then we can write in vector form $\mathbf{m} = \mathbf{m_0} + \mathbf{C} \cdot \boldsymbol{\sigma}$, which is referred to as the *linear* (or *fundamental*) *state equation* of the net.

The set of markings *reachable* from $\mathbf{m_0}$ in $\mathcal{N}$ is denoted as $RS(\mathcal{N}, \mathbf{m_0})$ and is called the *reachability set*.

A place $p \in P$ is $k-bounded$ if $\forall \mathbf{m} \in RS(\mathcal{N}, \mathbf{m_0}), \mathbf{m}(p) \leq k$. A net system $\mathcal{S}$ is *k-bounded* if each place is $k$-bounded. A net system is *bounded* if there exists some $k$ for which it is $k$-bounded. A net $\mathcal{N}$ is *structurally bounded* if it is bounded no matter which $\mathbf{m_0}$ is the initial marking.

A place $p$ is *identical* to a place $p' \neq p$ if $\mathbf{m_0}(p) = \mathbf{m_0}(p'), \mathbf{Pre}(p, \cdot) = \mathbf{Pre}(p', \cdot)$, and $\mathbf{Post}(p, \cdot) = \mathbf{Post}(p', \cdot)$, where $\mathbf{Pre}(p, \cdot)$, $\mathbf{Post}(p, \cdot)$, is the row of $\mathbf{Pre}$, $\mathbf{Post}$, corresponding to a place $p$. Places $p, p' \neq p$, are *series* places if $\mathbf{Pre}(p, \cdot) = \mathbf{Post}(p', \cdot)$.

Two transitions $t$, $t'$ are said to be in *structural conflict* if they share at least one input place, i.e., ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$. Two transitions $t$, $t'$ are said to be in *effective conflict for a marking* $\mathbf{m}$ if they are in structural conflict and they are both enabled at $\mathbf{m}$.

A *p-semiflow* is a non-negative vector $\mathbf{y} \geq \mathbf{0}$ such that it is a left anuller of the net's incidence matrix, i.e., $\mathbf{y}^\top \cdot \mathbf{C} = \mathbf{0}$. A p-semiflow implies a token conservation law independent of any firing of transitions. A p-semiflow $\mathbf{v}$ is *minimal* when its support, $\|\mathbf{v}\| = \{i | \mathbf{v}(i) \neq 0\}$, is not a proper superset of the support of any other p-semiflow, and the greatest common divisor of its elements is one. A Petri net is said to be *conservative* if there exists a p-semiflow which contains all places in its support.

**Definition 3.** *[49] A marked PN $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0} \rangle$ is said to be* persistent *if for any reachable marking $\mathbf{m}$ and for all transitions $t_i, t_j, t_i \neq t_j$, enabled in $\mathbf{m}$, the sequence $t_i, t_j$ is firable from $\mathbf{m}$.*

**Definition 4.** *A net $\mathcal{N}$ is said to be a* structurally persistent net *if $\langle \mathcal{N}, \mathbf{m_0} \rangle$ is persistent for all finite initial markings $\mathbf{m_0}$.*

Note that structurally persistent nets are totally conflict-free, i.e., no pair of transitions is in structural or effective conflict. That is, $\forall p \in P, |p^\bullet| \leq 1$.

## B. Fault Trees

A fault tree defines a top event, located at the root of the tree, and a set of children connected through logic gates indicating how the combination of the lower level events must happen. The logic gates express the failure logic of the FT under analysis. In this paper, we focus on coherent fault trees, a type of FT where the logic formulae are restricted to AND/OR gates. An AND gate (Figure 1(a)) indicates that the output event occurs when all inputs are present. In contrast, an OR gate (Figure 1(b)) triggers the output event when at least one of the inputs occurs.

An event of an FT can be of several kinds. A *basic event* is a component or human fault for which statistical failure and repair data is available. A *conditioning event* specifies conditions or constraints to trigger a gate. An *external event* (or *house event*) specifies an event normally expected to occur. An *undeveloped event* defines an event that is developed no further, either because there is a lack of information or because there is no consequence. Finally, an *intermediate event* represents a middle (and the top) event generated as a combination of other types of events. An FT can also have the *in/out transfer symbols*, which are used to divide large fault trees into smaller ones in order to perform an analysis by parts, and also to reduce the duplication of fault trees. Figure 1 shows the graphical symbols of all these FT elements.

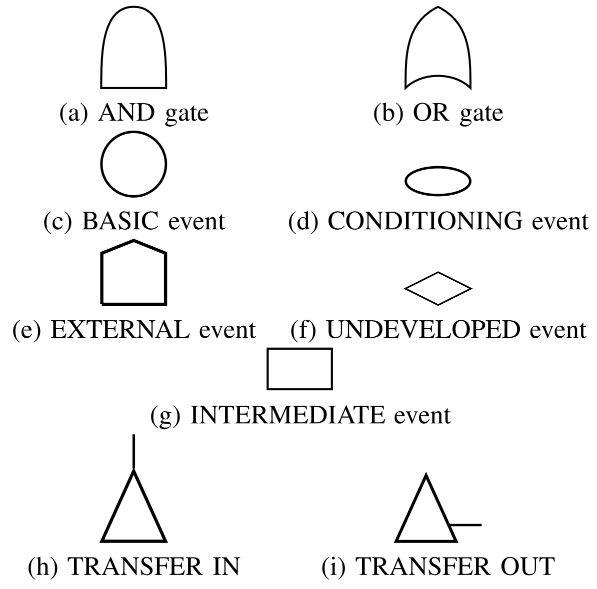


Figure 1. Fault tree gates, events, and transfer graphical symbols.

An FTA is used to identify and assess all combinations of the undesired events in the context of a system that lead to an undesired state, expressed by the TE [50]. A *cut set* is a set of basic events whose occurrence causes a system to fail (that is, the TE occurs) [3]. A cut set is said to be a *minimal cut set* when it cannot be further reduced and its events still lead the system to an undesired state (i.e., it contains the minimum events that cause the TE to occur). A *path set* is a dual set of the cut set. It contains the basic events of the FT whose non-occurrence assures the non-occurrence of the TE. A path set is said to be a *minimal path set* when, having removed any basic event from this set, the remaining events no longer constitute a path set [3]. The MCS and MPS are related: for a given coherent FT, its MPS are equal to the MCS of the dual FT. A dual FT is obtained by transforming AND gates of such an FT into OR gates, and *vice versa*. We formally define a coherent fault tree as follows:

**Definition 5.** *A coherent fault tree (FT) is a 5–tuple $\mathcal{F} = \langle \mathcal{E}, \mathcal{G}, \mathcal{G}^+, \mathcal{G}^*, \mathcal{T} \rangle$, where:*

- $\mathcal{E}, |\mathcal{E}| \geq 1$, *is a non-empty set of* basic, undeveloped, or external events*;*
- $\mathcal{G}, |\mathcal{G}| \geq 1, \mathcal{G} \cap \mathcal{E} = \emptyset$, *is a non-empty set of* intermediate events*;*
- $\mathcal{G}^+ : \mathcal{G} \times (\mathcal{E} \cup \mathcal{G}) \to \{0, 1\}$ *relates, for each intermediate event $g \in \mathcal{G}$, the events $e_1, \ldots, e_n \in \mathcal{E} \cup \mathcal{G}, n > 1$, such that at least one of them must occur (value 1) to trigger the output event $g$;*
- $\mathcal{G}^* : \mathcal{G} \times (\mathcal{E} \cup \mathcal{G}) \to \{0, 1\}$ *relates, for each intermediate event $g \in \mathcal{G}$, the events $e_1, \ldots, e_n \in \mathcal{E} \cup \mathcal{G}, n > 1$, such that all of them must occur (value 1) to trigger the output event $g$; and*
- $\mathcal{T} = \{g\}, g \in \mathcal{G}$ *is the* top event *of the FT $\mathcal{F}$.*

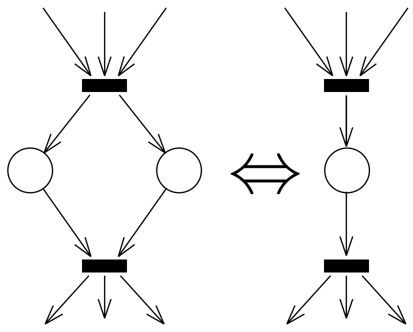
For convenience, in the sequel we denote $\mathcal{G}^+, \mathcal{G}^*$, in matrix form, i.e., $\mathcal{G}^+, \mathcal{G}^* \in \{0,1\}^{|\mathcal{G}|\times(|\mathcal{E}|+|\mathcal{G}|)}$. An event $g \in \mathcal{G}$ has only non-null components in either $\mathcal{G}^+$ or $\mathcal{G}^*$, and not both. That is, let $g \in \mathcal{G}$ be an intermediate event and $\mathcal{G}^+(g,\cdot)$, $\mathcal{G}^*(g,\cdot)$, its corresponding row of $\mathcal{G}^+$, $\mathcal{G}^*$, respectively. Then, $\forall g \in \mathcal{G}, \mathcal{G}^+(g,\cdot)\cdot\mathbf{1} > 0 \Leftrightarrow \mathcal{G}^*(g,\cdot)\cdot\mathbf{1} = 0$, and vice versa. Self-feedback is not allowed in intermediate events, i.e., $\forall g \in \mathcal{G}, \mathcal{G}^+(g,g) = \mathcal{G}^*(g,g) = 0$.

Note that here, we only consider the structure and not the probabilities of failure of basic events, normally used for the quantitative evaluation of an FT.

## IV. Model Transformation: From an FT to a Structurally Persistent Net

In this section, we present a model transformation from a fault tree $\mathcal{F}$ to a Petri net system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$, where $\mathcal{N}$ is indeed a bounded, structurally persistent net. The set of places $P$ of the obtained Petri net is divided into three disjoint sets $P_E, P_{EG}$, and $P_G$, i.e., $P = P_E \cup P_{EG} \cup P_G, P_E \cap P_G = \emptyset, P_{EG} \cap P_G = \emptyset, P_E \cap P_{EG} = \emptyset$. The model transformation steps are as follows:

1) First, each event $e \in \mathcal{E}$ is transformed into a place $p_e \in P_E \subset P$ and into a transition $t_e$, where $p_e$ is a place that represents the event $e$, and $t_e$ is connected as output to $p_e$, i.e., $p_e^\bullet = \{t_e\}$. The initial marking of $p_e$ is set to one, i.e., $\mathbf{m_0}(p_e) = 1$. To relate a Petri net place $p_e \in P_E$ to a fault tree event $e \in \mathcal{E}$, we define a set $\mathcal{R} : P_E \times \mathcal{E}$, and add the tuple $\langle p_e, e\rangle$ to $\mathcal{R}$ (i.e., $\mathcal{R} = \mathcal{R} \cup \langle p_e, e\rangle$).
2) Then, each intermediate event $g \in \mathcal{G}$ is transformed into a place $p_g \in P_G$ and into a transition $t_g$, where $p_g$ is a place that represents the event $g$, and $p_g^\bullet = \{t_g\}$. The initial marking of $p_g$ is set to zero, i.e., $\mathbf{m_0}(p_g) = 0$.
3) Now, the transformations of the logical OR and AND relations are carried out, following the guidelines given in [51]. First, we describe the transformation of OR gates. Let $\mathcal{A}$ be the set of tuple events $\langle g, e\rangle$, such that $\forall g \in \mathcal{G}, e \in \mathcal{E}\cup\mathcal{G}, \mathcal{G}^+(g,e) = 1$. Then, for each $\langle g, e\rangle \in \mathcal{A}$, a place $p \in P_{EG}$ and a transition $t$ are created. Such a place $p$ is connected as an input place to $t$, and as an output place to $t_e$, where $t_e$ is the output transition of $p_e$, and $p_e$ is the place that represents an event $e \in \mathcal{E}\cup\mathcal{G}$ (created in previous steps), i.e., ${}^\bullet t = \{p\}, t_e^\bullet = \{p\}$. Finally, the transition $t$ is connected to place $p_g$, i.e., $t^\bullet = \{p_g\}$.
Now, we describe the transformation of AND gates. Let $\mathcal{A}$ be the set of tuple events $\langle g, e\rangle$, such that $\forall g \in \mathcal{G}, e \in \mathcal{E}\cup\mathcal{G}, \mathcal{G}^*(g,e) = 1$. In this case, a single transition $t$ is created first, and $t^\bullet = \{p_g\}$. Then, for each $\langle g, e\rangle \in \mathcal{A}$, a place $p \in P_{EG}$ is created, and $p^\bullet = \{t\}, t_e^\bullet = \{p\}$. Figures 2(a,b) and 2(c,d) graphically describe an example of this step performed with an OR and AND gate, respectively.
4) As the last step, the transition $t_g$ of place $p_g$ that represents the top event of $\mathcal{F}$, i.e., $p_g \in \mathcal{T}$, is removed. As we will see below, this ensures, the conservativeness of the obtained Petri net model.
5) As an optional step, place reduction rules such as the fusion of series places and the elimination of identical places (see Figure 3) can be applied to minimize the number of places in the PN [52]. Note that PN properties remain in spite of the application of these reductions.


(a) Elimination of identical places

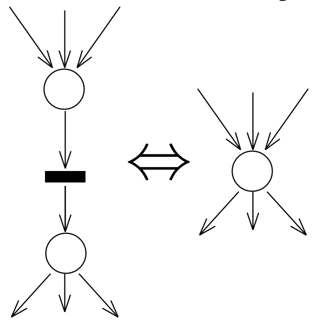

(b) Fusion of series places

Figure 3. Particular cases of PN reduction rules.

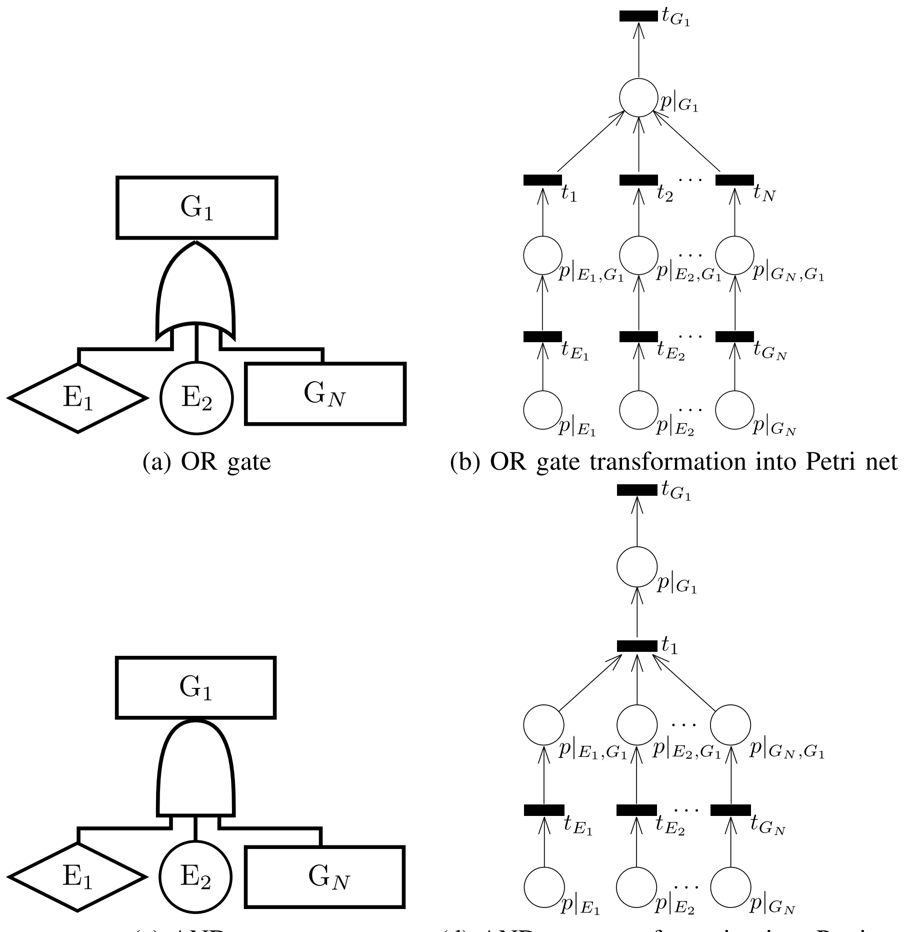
The Petri net $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$ obtained after implementing the described model transformation steps in a fault tree is a *structurally persistent net* (see Section III-A). In the sequel, we termed as FT-SPN $\mathcal{S}_\mathcal{F} = \langle \mathcal{N}, \mathcal{R}, \mathbf{m_0}\rangle$ to the structurally persistent net $\mathcal{S}$ obtained as a result of the above model transformation and the relation between Petri net places and FT events $\mathcal{R}$.

Note that as a FT is acyclic, an FT-SPN is acyclic as well. Furthermore, the model transformation does not create any transition in the FT-SPN without input places (i.e., $\forall t \in T, |{}^\bullet t| \geq 1$) and thus, the net is bounded.

## V. Fault Tree Analysis Using P-Semiflows

In this section, we firstly prove the conservativeness of FT-SPNs and the relationship between a minimal p-semiflow in an FT-SPN and MCS and MPS of an FT. Then, we present an LP problem to compute the minimal p-semiflows of an FT-SPN, which is integrated into an algorithm to efficiently compute the MCS/MPS of a coherent FT.

As described in Section III, a p-semiflow of a PN is a non-negative vector $\mathbf{y} \geq \mathbf{0}$ such that it is a left anuller of the net's incidence matrix, i.e., $\mathbf{y}^\top \cdot \mathbf{C} = \mathbf{0}$. Recall that a p-semiflow implies a token conservation law independent of any firing of transitions, i.e., for a given Petri net $\mathcal{N}, \forall \mathbf{m_0}, \forall \mathbf{m} \in RS(\mathcal{N}, \mathbf{m_0}), \mathbf{y}^\top \cdot \mathbf{m} = \mathbf{y}^\top \cdot \mathbf{m_0}$. Given the model transformation presented in Section IV, the number of places in an FT-SPN with an initial marking other than zero matches the number of events in the corresponding FT, i.e., $\|\mathbf{m_0}\| = \|\mathcal{E}\|$. As the top event $g \in \mathcal{T}$ is represented by a



(a) OR gate (b) OR gate transformation into Petri net

(c) AND gate (d) AND gate transformation into Petri net

Figure 2. Transformation of OR and AND gates into Petri nets.

place $p_g$ in which the net ends, it is intuitively clear to state that $p_g$ will appear in all the p-semiflows of an FT-SPN.

**Theorem 1.** *An FT-SPN is conservative.*

*Proof:* Let us consider firstly the top event $\mathcal{T} = \{g\}, g \in \mathcal{G}$, of a coherent FT represented by a place $p_\mathcal{T}$ and a transition $t_\mathcal{T}$ in the FT-SPN. Suppose that $\mathcal{G}^+(g,:) \cdot \mathbf{1} > 0$, that is, the lower events leading to $g$ are joined by an OR gate. Recall that with the model transformation, an OR gate having $\mathcal{X} = \{e_1, \ldots, e_n\}$ input events creates $t_1, \ldots, t_n$ transitions and $p_1, \ldots, p_n$ places in the FT-SPN. These places and transitions are interconnected as follows: $p_i^\bullet = \{t_i\}, t_i^\bullet = \{p_\mathcal{T}\}, i = 1, \ldots, n$. According to the token conservation law, the sum of tokens at such places and $p_\mathcal{T}$ remains constant, i.e., they belong to the same p-semiflow. Suppose now that $\mathcal{G}^*(g,:) \cdot \mathbf{1} > 0$, that is, the lower events leading to $g$ are joined by an AND gate. Again, with the model transformation there will be only one transition and so many places as input events to the AND gate in the FT. According to the token conservation law, the sum of tokens at every place and $p_\mathcal{T}$ remains constant, i.e., they belong to the same p-semiflow.

The same reasoning can be recursively applied to each input event of $g$, and so on and so forth until all basic events are reached. Thus, all places of an FT-SPN belong to some p-semiflow, which ensures the conservativeness of the net. ■

Since an FT-SPN is conservative, we can state that a basic event $e \in \mathcal{E}$ of an FT will be included in at least one of the p-semiflows of an FT-SPN. Indeed, a minimal p-semiflow of an FT-SPN relates the events that lead to the top event $t \in \mathcal{T}$. That is, the minimal p-semiflows of an FT-SPN of a given FT provide the set of minimal path sets, i.e., the set of events such that when none of them occur, the top event does not occur either.

**Theorem 2.** *Let $\mathcal{F} = \langle \mathcal{E}, \mathcal{G}, \mathcal{G}^+, \mathcal{G}^*, \mathcal{T} \rangle$ be a coherent FT transformed into an FT-SPN $\mathcal{S}_\mathcal{F} = \langle \mathcal{N}, \mathcal{R}, \mathbf{m_0} \rangle$. The set of places $p \in P_E$ contained in the support of a minimal p-semiflow of $\mathcal{N}$ representing events $e \in \mathcal{E}$ defines a path set of $\mathcal{F}$.*

*Proof:* Suppose an intermediate event $g \in \mathcal{G}$ such that $\mathcal{X} = \{e_1, \ldots, e_n\}, e_i \in \mathcal{E} \cup \mathcal{G}, \mathcal{G}^+(g, e_i) = 1, i = 1 \ldots n$, is the set of events that at least one must occur to trigger $g$. Let $p_g, t_g$ be the place, transition representing the event $g$ in the FT-SPN, respectively. Then, $\forall t \in {}^\bullet p_g, p_{e_i}^\bullet = \{t\}$, where place $p_{e_i}$ represents the event $e_i \in \mathcal{X}$. According to the token conservation law, any p-semiflow $\mathbf{y}$ that contains $p_g$ in its support must also contain the set of places $\{p_{e_i}, \forall e_i \in \mathcal{X}\}$, that is, $\{p_g\} \cup \{p_{e_i}, \forall e_i \in \mathcal{X}\} \in \|\mathbf{y}\|$. Therefore, the p-semiflow $\mathbf{y}$ relates the events $e_1, \ldots, e_n$ that must not occur in order not to trigger the event $g$.

Suppose now an intermediate event $g \in \mathcal{G}$ such that $\mathcal{X} = \{e_1, \dots e_n\}, e_i \in \mathcal{E} \cup \mathcal{G}, \mathcal{G}^*(g, e_i) = 1, i = 1 \dots n$, is the set of events which all must occur to trigger $g$. Let $p_g, t_g$ be the place, transition representing the event $g$ in the FT-SPN, respectively. Then, $\exists t \in T, {}^\bullet p_g = p_{e_i}^\bullet = t$, where place $p_{e_i}$ represents the event $e_i \in \mathcal{X}$. Thus, according to the token conservation law, there exists a minimal p-semiflow $\mathbf{y}$ that contains $p_g$ in its support and also contains one of the places $p \in \mathcal{A} = \{p_{e_i}, \forall e_i \in \mathcal{X}\}$, that is, $\{p_g\} \cup \{p \in \mathcal{A}\} \in \|\mathbf{y}\|$. As before, such p-semiflows relate the events $e_1, \dots, e_n$ that must not occur in order not to trigger the event $g$. ■

Note that if an FT $\mathcal{F}$ is transformed into its dual $\mathcal{F}'$ (i.e., matrices $\mathcal{G}^+, \mathcal{G}^*$ are interchanged), the p-semiflows of $\mathcal{F}'$ relate the cut set of events whose occurrence will lead the top event to occur.

Therefore, the set of minimal p-semiflows of an FT-SPN representing a (dual) FT contains all the (cut sets) path sets of such an FT. After computing these sets, they must be reduced to obtain the MCS and MPS of the FT. Computation of the set of minimal p-semiflow $\mathcal{Y}$ of a Petri net is generally a computationally expensive task (a EXPSPACE-hard problem [53]). The number of minimal p-semiflows can be upper bounded to the number of incomparable vectors of dimension $n = |P|$, i.e., $|\mathcal{Y}| \leq \binom{n}{\lceil n/2 \rceil}$ [54]. Thus, it is desirable to reduce the FT-SPN size by applying the place reduction rules previously described in Section IV.

A review of p-semiflow computation in Petri nets using mathematical programming can be found in [54], where Colom and Silva propose improved algorithms to compute minimal p-semiflows using variants of the Fourier-Motzkin Method [55]. It is also worth to mention the work reported in [56], where a symbolic method for p-semiflow computation based on zero-suppressed decision diagrams is presented. The results outperform classic p-semiflow computation in cases where the number of p-semiflows is intractable. Any of these p-semiflow computation methods [54], [56], [57] can be used to compute the minimal p-semiflows of an FT-SPN.

In this paper, we propose a Linear Programming Problem (LPP) that computes the minimal p-semiflow of an FT-SPN, after applying reduction rules, having a given place in its support. The application of reduction rules in the FT-SPN becomes a mandatory step, as we will show in the sequel.

**Theorem 3.** *A minimal p-semiflow* $\mathbf{y}$ *of a FT-SPN, after applying reduction rules, that includes* $p \in P_E$ *in its support, i.e.,* $p \in \|\mathbf{y}\|$*, can be computed by the following Linear Programming problem:*

$$\begin{aligned} &maximize\ \mathbf{y}(p) \\ &subject\ to \\ &\quad \mathbf{y}^\top \cdot \mathbf{C} = \mathbf{0} \\ &\quad \mathbf{y}^\top \cdot \mathbf{m_0} = 1 \\ &\quad \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{1}$$

*Proof:* Suppose that $\mathbf{y}$ is a linear combination of p-semiflows, i.e., $\mathbf{y} = \sum_{i=1}^{n} \alpha_i \cdot \mathbf{y}_i, \alpha_i > 0$. The second constraint of the LPP (1) implies that $\mathbf{y} \cdot \mathbf{m_0} = 1$, i.e., $\sum_{i=1}^{n} \alpha_i \cdot \mathbf{y}_i \cdot \mathbf{m_0} = \alpha_1 \cdot \mathbf{y}_1 \cdot \mathbf{m_0} + \alpha_2 \cdot \mathbf{y}_2 \cdot \mathbf{m_0} + \dots + \alpha_n \cdot \mathbf{y}_n \cdot \mathbf{m_0} = 1, \alpha_i > 0$. Recall that in the model transformation, only places $p \in P_E$ are initially marked, i.e., $\mathbf{m_0}(p) = 1, \forall p \in P_E, \mathbf{m_0}(p') = 0, \forall p' \in P \setminus P_E$. Thus, $\mathbf{y}_i \cdot \mathbf{m_0} = \sum \mathbf{y}_i(p), p \in P_E, p \in \|\mathbf{y}_i\|$. Therefore, the second constraint becomes $\alpha_1 \cdot \sum \mathbf{y}_1(p) + \alpha_2 \cdot \sum \mathbf{y}_2(p) + \dots + \alpha_n \cdot \sum \mathbf{y}_n(p) = 1, \alpha_i > 0$, where $p \in P_E, p \in \|\mathbf{y}_i\|, i = 1 \dots n$. Since the support of $\mathbf{y}$ covers more places than any support of individual p-semiflows $\mathbf{y}_i$, in order to fulfill the aforementioned constraint $\mathbf{y}(p)$ for a given $p \in P_E$, the value of $\mathbf{y}(p)$ is not maximum, which contradicts the optimization function of LPP (1). Therefore, $\mathbf{y}$ cannot be a linear combination of p-semiflows, that is, $\mathbf{y}$ is a minimal p-semiflow. ■

Note that a minimal p-semiflow of a FT-SPN (dual FT-SPN) defines a path set (cut set, respectively) of the original coherent fault tree. Recall that computing the minimal p-semiflows of a Petri net is known to be equal to computing the extreme points of a polyhedral cone [54]. Thus, since an LPP can be solvable in polynomial time [58], we have shown that the computation of the MCS/MPS in a coherent FT is not a NP-hard problem, as it is in a general case [6]. To the best of our knowledge, this result has not been explicitly discussed before.

**Corollary 1.** *The computation of the minimal cut sets and minimal path sets of a coherent Fault Tree are solvable in polynomial time.*

Note that there may exist many p-semiflows, whose number may not be polynomially with respect to the structural size of a Petri net. The key point is that, by definition, the structure of an FT-SPN is never going to be so complex that the number of minimal p-semiflows would grow exponentially. A particular case of a Petri net structure that leads to an exponential number of p-semiflows is depicted in Figure 4.

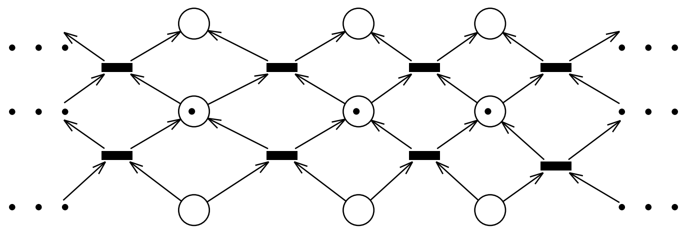


Figure 4. A particular Petri net structure that leads to an exponential growth of minimal p-semiflows.

## *A. An Algorithm to Compute the MCS and MPS of a Coherent FT*

The LPP (1) can be used to iteratively compute the cut sets and path sets of an FT transformed into an FT-SPN, obtaining the MCS/MPS after reducing supersets. In what follows, we develop an algorithm that makes use of LPP (1).

Recall that some reduction rules can be applied in an FT-SPN in order to minimize its size, i.e., to potentially minimize the number of places. Places representing events are also candidates for elimination when applying reduction and LPP (1) should consider each one of these places independently, computing its associated minimal p-semiflow.

However, according to the token conservation law, the p-semiflows of identical places will be the same, that is, let $p$ and $p'$ be a pair of identical places that represent events (i.e., $p, p' \in P_E$) and $\mathbf{y}_p, \mathbf{y}'_p$ their corresponding p-semiflows. The support of both p-semiflows $\mathbf{y}_p, \mathbf{y}'_p$ removing $p, p'$ is the same, i.e., $\|\mathbf{y}_p\| \setminus \{p\} = \|\mathbf{y}'_p\| \setminus \{p'\}$. Therefore, we can minimize the number of computations of LPP (1) by tracking the identical places in $P_E$ eliminated when applying the reduction rules. We have termed this phase the *folding* step, as a single place represents a set of removed places. In this phase, we define $P_f : P_E \times P_E$ to relate the places that are identical. Note that in a set of identical places, we can arbitrarily choose one of these places as being representative of the others. In the example explained above, $\langle p, p' \rangle \in P_f$.

Similarly, after computing the minimal p-semiflow of a folded place, an *unfolding* step must be performed. That is, those places that represent several events are considered separately, with the rest of the places belonging to the support of the obtained minimal p-semiflow.

Let us illustrate how these folding/unfolding steps work by means of a naïf example. Suppose an FT-SPN in which, after folding, places $p_1$, $p_2$, and $p_3$ are fused because they are identical places, i.e., $P_f = \{\langle p_1, p_2 \rangle, \langle p_1, p_3 \rangle\}$. Let $\mathbf{y} = \{p_1, p_4, p_5, p_6\}$ be the minimal p-semiflow that contains $p_1$ in its support, obtained as a solution of LPP (1). In this case, the unfolding step over $\mathbf{y}$ will obtain, in this case, three minimal p-semiflows: $\mathbf{y}_1 = \{p_1, p_4, p_5, p_6\}$, $\mathbf{y}_2 = \{p_2, p_4, p_5, p_6\}$, and $\mathbf{y}_3 = \{p_3, p_4, p_5, p_6\}$.

This is the key reason why these folding/unfolding steps become mandatory, since otherwise we might "lose" one or another minimal p-semiflow: the solution of LPP (1) is a single $\mathbf{y}$; thus, in the case of a place $p$ related to some other places and sharing several minimal p-semiflows with the same value of $\mathbf{y}(p)$, only one of these minimal p-semiflows will be obtained as the solution of LPP (1).

Algorithm 1 describes the steps to compute the MCS and MPS of a coherent FT. As input, it takes a coherent FT $\mathcal{F}$ formally defined as previously described in Section III. As output, it returns $\mathcal{M}, \mathcal{M}'$, which respectively represent the set of MCS and MPS of $\mathcal{F}$.

Step 1 initializes the sets $\mathcal{M}, \mathcal{M}'$, which represent the minimal cut sets and the minimal path sets of $\mathcal{F}$, respectively. Steps 2–9 compute $\mathcal{M}'$. Firstly, the transformation from $\mathcal{F}$ to an FT-SPN $\mathcal{S}_{\mathcal{F}}$ is performed in step 2. Then, a folding step occurs as previously described. After that, in steps 4–8, an iteration process occurs for each place that represents an event $e \in \mathcal{E}$: First of all, its associated minimal p-semiflow $\mathbf{y}$ is computed using LPP (1); then, a set of minimal p-semiflows $\mathcal{Z}$ is obtained as an unfolding step over $\mathbf{y}$; and finally, the set of places contained in the support of the minimal p-semiflows of $\mathcal{Z}$ and initially marked (i.e., they represent an event $e \in \mathcal{E}$) are added as a new Path Set to $\mathcal{M}'$. Note that $\mathcal{R}$ is used when building these sets. Finally, step 9 removes supersets contained in $\mathcal{M}'$, thus obtaining the MPS of $\mathcal{F}$. The set of MCS is obtained in a similar way (steps 11-18), after a transformation of $\mathcal{F}$ into its dual $\mathcal{F}'$ (step 10). Recall that reduction rules must be applied after obtaining $\mathcal{F}'$. Note that after the solution of LPP (1) is obtained, it is unnecessary to transform back from Petri nets to FT since $\mathcal{R}$ maintains the relationship between places and events.

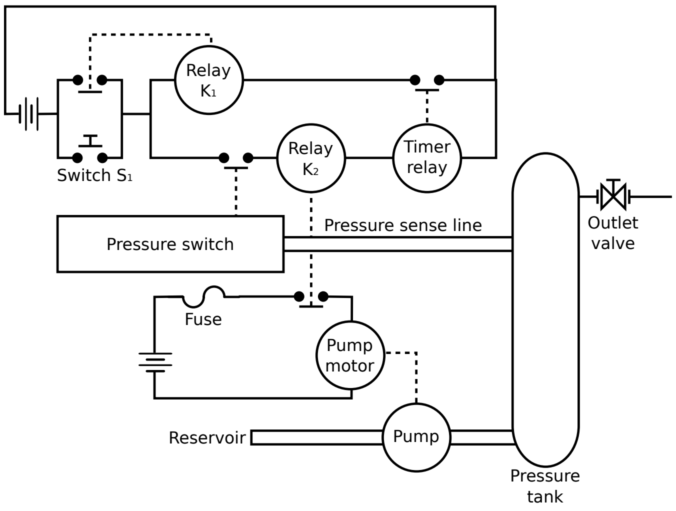


Figure 5. A pressure tank system.

The computational complexity of Algorithm 1 is polynomial, since all the steps have a polynomial-time order complexity. Let a coherent FT $\mathcal{F} = \langle \mathcal{E}, \mathcal{G}, \mathcal{G}^+, \mathcal{G}^*, \mathcal{T} \rangle$ be the input. The model transformation iterates on events, and on the AND and OR relationships. Thus, its cost in a worst-case scenario can be approximated to $O(2 \cdot |\mathcal{G}| \cdot (|\mathcal{E}| + |\mathcal{G}| + 1))$. The folding step iterates on the places $p \in {}^\bullet t, |{}^\bullet t| > 1, t \in T$, which can be roughly upper bounded by $O(|T| \cdot |P|)$. Similarly, the unfolding step depends on the total number of groups $g$ of places folded, and the number of places $l$ within this group. Thus, the unfolding process complexity time can be upper bounded to $O(g \cdot l)$, where $g \cdot l \leq |P_E|$. The cost of solving the LPP (1) depends on the method used [58]. Although the well-known simplex method was discovered to have a worst-case exponential performance [59], there are methods (e.g., interior-point methods [60]) that have a worst-case polynomial performance [61]. Finally, the computation effort needed to eliminate supersets can be roughly upper bounded by $O(|\mathcal{E}|^{3/2})$.

In the next section we present a qualitative analysis of an industrial case study, namely, a pressure tank system. We apply Algorithm 1 to directly compute the MCS and MPS of the system, describing the algorithm steps in more detail.

## VI. Case Study: A Pressure Tank System

Consider a pressure tank system (taken from [4]) depicted in Figure 5. The system has three working modes: operational mode, shutdown mode, and emergency shutdown mode. In the operational mode, the reset switch $S_1$ is closed and immediately opened, allowing the current to flow in the control branch circuit. Then, the relay coils $K_1$ and $K_2$ are activated. Relay $K_1$ contacts are closed and latched, while the relay $K_2$ contacts are closed and then the pump motor starts working. When the pressure switch detects an excess pressure, the shutdown mode starts. First, the control circuit is deactivated, de-energizing the $K_2$ coil and opening its contacts, thus

**Algorithm 1:** Computation of the Minimal Cut Sets and Minimal Path Sets of a coherent Fault Tree.

```
Input: $\mathcal{F} = \langle \mathcal{E}, \mathcal{G}, \mathcal{G}^+, \mathcal{G}^*, \mathcal{T} \rangle$
Output: The minimal cut sets $\mathcal{M}$ and the minimal path sets $\mathcal{M}'$ of $\mathcal{F}$
1  $\mathcal{M} = \mathcal{M}' = \emptyset$
2  Transform $\mathcal{F}$ into an FT-SPN $\mathcal{S}_\mathcal{F} = \langle \mathcal{N}_\mathcal{F}, \mathcal{R}, \mathbf{m_0} \rangle$ (see Section IV)
3  Build $P_f$, folding step over $\mathcal{S}_\mathcal{F}$
4  foreach $p \in P_E$ do
5      Compute the minimal p-semiflow $\mathbf{y}$ that contains $p$ in its support, using LPP (1)
6      Unfolding step over $\mathbf{y}$ taking $P_f$ into account, obtaining a set of minimal p-semiflows $\mathcal{Z}$
7      $\forall \mathbf{z} \in \mathcal{Z}, \mathcal{M}' = \mathcal{M}' \bigcup \{e | \langle p, e \rangle \in \mathcal{R}, p \in \|\mathbf{z}\| \bigcap \|\mathbf{m_0}\|\}$
8  end
9  Remove the supersets from $\mathcal{M}$
10 Transform $\mathcal{F}$ into its dual $\mathcal{F}'$
11 Transform $\mathcal{F}'$ into an FT-SPN $\mathcal{S}_{\mathcal{F}'} = \langle \mathcal{N}_{\mathcal{F}'}, \mathcal{R}, \mathbf{m_0} \rangle$ (see Section IV)
12 Build $P_f$, folding step over $\mathcal{S}_{\mathcal{F}'}$
13 foreach $p \in P_E$ do
14     Compute the minimal p-semiflow $\mathbf{y}$ that contains $p$ in its support, using LPP (1)
15     Unfolding step over $\mathbf{y}$ taking $P_f$ into account, obtaining a set of minimal p-semiflows $\mathcal{Z}$
16     $\forall \mathbf{z} \in \mathcal{Z}, \mathcal{M} = \mathcal{M} \bigcup \{e | \langle p, e \rangle \in \mathcal{R}, p \in \|\mathbf{z}\| \bigcap \|\mathbf{m_0}\|\}$
17 end
18 Remove the supersets from $\mathcal{M}$
```

shutting the motor down. Finally, an emergency shutdown mode starts when the pressure switch hangs up: the timer relay contacts should open after a timeout of 60 seconds, de-energizing $K_1$ and $K_2$ coils, thus shutting the pump motor off. The timer resets itself automatically after each cycle.

Figure 6 shows the FT of the pressure tank system. The *Top Event* represents a pressure tank rupture. For the sake of clarity, we summarize all the events of the FT (taken from [4]) in Table I.

## A. Qualitative Evaluation

Recall that a qualitative evaluation of an FT renders it possible to obtain the logic equations that describe each gate of the FT. Here, we qualitatively evaluate an FT, obtaining the logic equation associated with the top event of the FT by applying Algorithm 1.

The FT illustrated in Figure 6 is given as input. The FT-SPN obtained after steps 2 and 3 is depicted in Figure 7(a). In this case, step 3 has fused the series places, but no identical place has been eliminated. Let us now iterate through basic event places by order. For place $p|_{E_1}$, step 6 computes the minimal p-semiflow $\mathbf{y}_1 = \{p|_{TopEvent}, p|_{E_1}, p|_{G_1}, p|_{E_2}, p|_{G_2}, p|_{E_3}, p|_{E_4}, p|_{E_5}, p|_{G_3}, p|_{G_4}, p|_{G_6}, p|_{G_7}, p|_{E_9}, p|_{E_{10}}, p|_{E_{11}}, p|_{E_{12}}, p|_{E_{13}}, p|_{G_8}, p|_{E_{14}}, p|_{E_{15}}, p|_{E_{16}}\}$. This minimal p-semiflow conforms the path set $\mathcal{PS}_1 = \{E_1, E_2, E_3, E_4, E_5, E_9, E_{10}, E_{11}, E_{12}, E_{13}, E_{14}, E_{15}, E_{16}\}$. The minimal p-semiflow $\mathbf{y}_1$ will also be the solution for places $p|_{E_2}, p|_{E_3}, p|_{E_4}$, and $p|_{E_5}$. However, for the place $p|_{E_6}$ (till the place $p|_{E_8}$), a new minimal p-semiflow $\mathbf{y}_2 = \{p|_{TopEvent}, p|_{E_1}, p|_{G_1}, p|_{E_2}, p|_{G_2}, p|_{E_3}, p|_{E_4}, p|_{E_5}, p|_{G_3}, p|_{G_5}, p|_{E_6}, p|_{E_7}, p|_{E_8}\}$ appears, conforming a new path set $\mathcal{PS}_2 = \{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8\}$. Finally, from the place $p|_{E_9}$ to the place $p|_{E_{12}}$ the same $\mathbf{y}_1$ is given as the

Table I
DESCRIPTION OF BASIC (E) AND INTERMEDIATE (G) EVENTS OF THE FT DEPICTED IN FIGURE 6.

| Event | Description |
|---|---|
| Top Event | Pressure tank rupture. |
| $E_1$ | Pressure tank ruptures under load. |
| $E_2$ | Tank ruptures due to improper installation. |
| $G_1$ | Secondary failure of ruptured pressure tank. |
| $E_3$ | Secondary failure of tank from some other out of tolerance conditions (e.g., mechanical, thermal). |
| $G_2$ | $K_2$ relay contacts remain closed for a time $T > 60$ seconds. |
| $E_4$ | $K_2$ relay contacts fail to open. |
| $E_5$ | $K_2$ relay secondary failure. |
| $G_3$ | EMF to $K_2$ relay coil for a time $T > 60$ seconds. |
| $G_4$ | EMF remains on pressure switch (P/S) contacts when P/S contacts closed for a time $T > 60$ seconds. |
| $G_5$ | P/S contacts closed, $T > 60$ seconds. |
| $G_6$ | EMF through $S_1$ switch contacts when P/S contacts closed, $T > 60$ seconds. |
| $G_7$ | EMF through $K_1$ relay contacts when P/S contacts closed, $T > 60$ seconds. |
| $E_6$ | Pressure switch secondary failure. |
| $E_7$ | Pressure switch contacts fail to open. |
| $E_8$ | Excess pressure not sensed by pressure-activated switch. |
| $E_9$ | $S_1$ switch secondary failure. |
| $E_{10}$ | $S_1$ switch contacts fail to open. |
| $E_{11}$ | External reset activation force remains on switch $S_1$. |
| $E_{12}$ | $K_1$ relay contacts fail to open. |
| $E_{13}$ | $K_1$ relay secondary failure. |
| $G_8$ | Timer relay contacts fail to open when P/S contacts closed, $T > 60$ seconds. |
| $E_{14}$ | Timer does not timeout due to improper setting installation. |
| $E_{15}$ | Timer relay contacts fail to open. |
| $E_{16}$ | Timer relay secondary failure. |

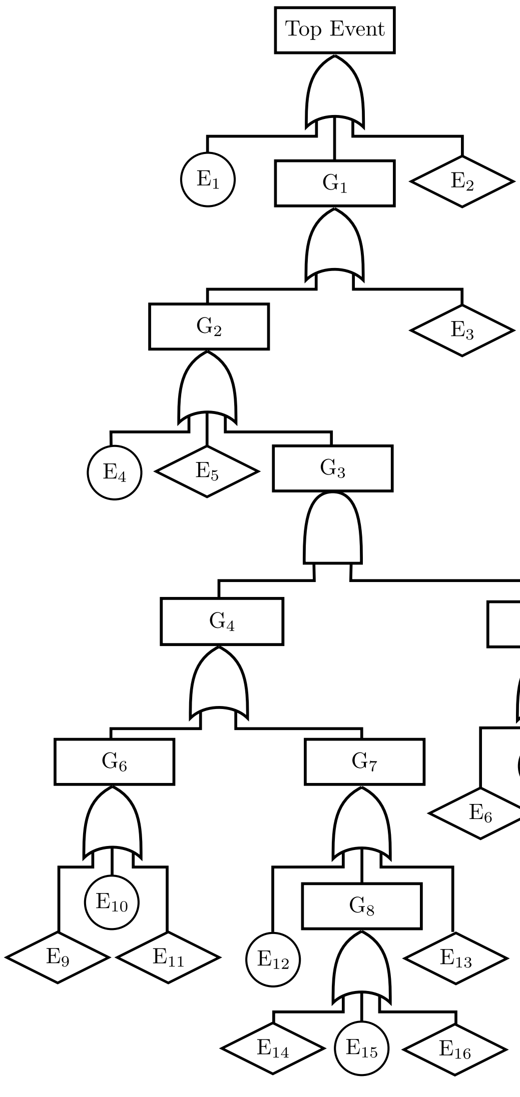


Figure 6. A fault tree of the pressure tank system depicted in Figure 5.

solution. Step 9 maintains both $\mathcal{PS}_1$ and $\mathcal{PS}_2$, since they are independent sets. Thus, they individually conform a Minimal Path Set. Recall that an MPS defines the set of events whose non-occurrence assures the non-occurrence of the TE. Note that Algorithm 1 is not optimal, as several iteration steps are repeated giving no new solutions. Additional conditions could be added to the iteration loop to prevent the selection of candidate places that have already been selected and cannot form part of some other p-semiflow.

The dual FT of $\mathcal{F}$ is computed in step 10, and its FT-SPN $\mathcal{S}_{\mathcal{F}'}$ in step 11. Let us consider that we skip step 12, i.e., we do not apply any reduction rule. Figure 8 shows an example of the place $p|_{G_8}$ in the dual FT-SPN, without applying any reduction rules. If we iterate over the LPP at step 14, and compute the minimal p-semiflows for each place initially marked, we obtain the minimal p-semiflows and MCS for each place shown in Table II. However, if we compute the minimal p-semiflows in the net using, for instance, the algorithm given in [54], we obtain a total of 29 minimal p-semiflows. This demonstrates the need to apply reduction rules in the FT-SPN, since they prevent this problem from happening.

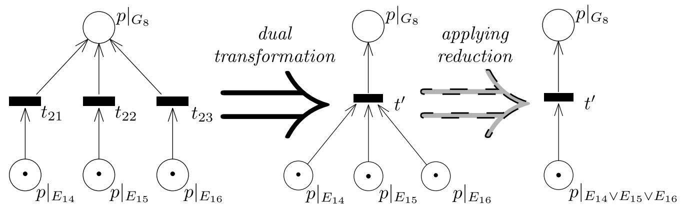


Figure 8. An excerpt of dual conversion in the FT-SPN and reduction processes.

Table III
MINIMAL CUT SETS OF THE FT ILLUSTRATED IN FIGURE 6.

| | | |
|---|---|---|
| $\mathcal{MCS}_1 = \{E_6, E_9\}$ | $\mathcal{MCS}_{11} = \{E_6, E_{15}\}$ | $\mathcal{MCS}_{21} = \{E_7, E_{12}\}$ |
| $\mathcal{MCS}_2 = \{E_6, E_{10}\}$ | $\mathcal{MCS}_{12} = \{E_6, E_{16}\}$ | $\mathcal{MCS}_{22} = \{E_7, E_{13}\}$ |
| $\mathcal{MCS}_3 = \{E_6, E_{11}\}$ | $\mathcal{MCS}_{13} = \{E_7, E_{14}\}$ | $\mathcal{MCS}_{23} = \{E_8, E_{12}\}$ |
| $\mathcal{MCS}_4 = \{E_7, E_9\}$ | $\mathcal{MCS}_{14} = \{E_7, E_{15}\}$ | $\mathcal{MCS}_{24} = \{E_8, E_{13}\}$ |
| $\mathcal{MCS}_5 = \{E_7, E_{10}\}$ | $\mathcal{MCS}_{15} = \{E_7, E_{16}\}$ | $\mathcal{MCS}_{25} = \{E_4\}$ |
| $\mathcal{MCS}_6 = \{E_7, E_{11}\}$ | $\mathcal{MCS}_{16} = \{E_8, E_{14}\}$ | $\mathcal{MCS}_{26} = \{E_5\}$ |
| $\mathcal{MCS}_7 = \{E_8, E_9\}$ | $\mathcal{MCS}_{17} = \{E_8, E_{15}\}$ | $\mathcal{MCS}_{27} = \{E_3\}$ |
| $\mathcal{MCS}_8 = \{E_8, E_{10}\}$ | $\mathcal{MCS}_{18} = \{E_8, E_{16}\}$ | $\mathcal{MCS}_{28} = \{E_1\}$ |
| $\mathcal{MCS}_9 = \{E_8, E_{11}\}$ | $\mathcal{MCS}_{19} = \{E_6, E_{12}\}$ | $\mathcal{MCS}_{29} = \{E_2\}$ |
| $\mathcal{MCS}_{10} = \{E_6, E_{14}\}$ | $\mathcal{MCS}_{20} = \{E_6, E_{13}\}$ | |

Let us now consider that we apply step 12 eliminating the identical places of basic events (e.g., places $p|_{E_{14}}, p|_{E_{15}}$, and $p|_{E_{16}}$ are fused in Figure 8), thus obtaining the FT-SPN depicted in Figure 7(b). Namely, the set of fused places is $P_f = \{\langle p|_{E_1}, p|_{E_2}\rangle, \langle p|_{E_4}, p|_{E_5}\rangle, \langle p|_{E_6}, p|_{E_7}\rangle, \langle p|_{E_6}, p|_{E_8}\rangle, \langle p|_{E_9}, p|_{E_{10}}\rangle, \langle p|_{E_9}, p|_{E_{11}}\rangle, \langle p|_{E_{12}}, p|_{E_{13}}\rangle, \langle p|_{E_{14}}, p|_{E_{15}}\rangle, \langle p|_{E_{14}}, p|_{E_{16}}\rangle\}$. In this case, the iteration loop at step 14 computes a total of six minimal p-semiflows:

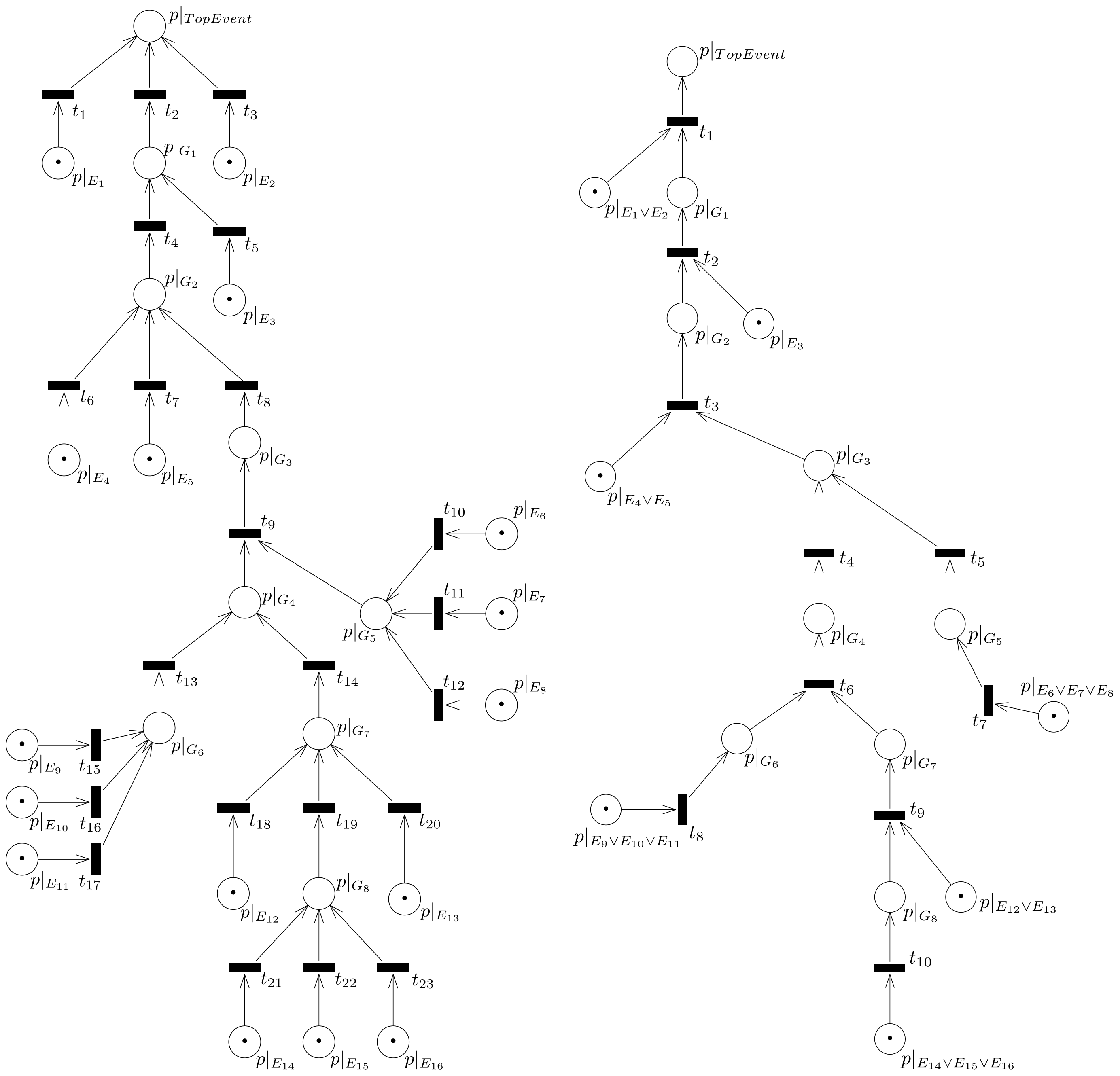
$$\begin{aligned}
\mathbf{y}_1 =& \{p|_{TopEvent}, p|_{G_1}, p|_{G_2}, p|_{G_3}, p|_{G_4}, p|_{G_5}, p|_{E_6\vee E_7\vee E_8},\\
& p|_{G_6}, p|_{E_9\vee E_{10}\vee E_{11}}\}\\
\mathbf{y}_2 =& \{p|_{TopEvent}, p|_{G_1}, p|_{G_2}, p|_{G_3}, p|_{G_4}, p|_{G_5}, p|_{E_6\vee E_7\vee E_8},\\
& p|_{G_7}, p|_{G_8}, p|_{E_{14}\vee E_{15}\vee E_{16}}\}\\
\mathbf{y}_3 =& \{p|_{TopEvent}, p|_{G_1}, p|_{G_2}, p|_{G_3}, p|_{G_4}, p|_{G_5}, p|_{E_6\vee E_7\vee E_8},\\
& p|_{G_7}, p|_{E_{12}\vee E_{13}}\}\\
\mathbf{y}_4 =& \{p|_{TopEvent}, p|_{G_1}, p|_{G_2}, p|_{E_4\vee E_5}\}\\
\mathbf{y}_5 =& \{p|_{TopEvent}, p|_{G_1}, p|_{E_3}\}\\
\mathbf{y}_6 =& \{p|_{TopEvent}, p|_{E_1\vee E_2}\}
\end{aligned}$$

Finally, unfolding the fused places, extracting the places initially marked and removing the linear combinations – similarly to the previous case, none is a superset – we obtain the Minimal Cuts Sets of $\mathcal{F}$ (steps 16 to 18). These MCS are listed in Table III. Note that the extracted MCS matches the number of minimal p-semiflows previously computed in the FT-SPN of the dual FT without applying any reduction rule.

Recall that the top event occurs iff at least all of the basic events in one MCS occur simultaneously, i.e. $P(TopEvent) = P(\bigcup_{i=1}^{n} \mathcal{MCS}_i)$, where $n$ is the number of MCS. Therefore, once the MCS are known, they can be used for quantitative analysis.

## VII. CONCLUSIONS

A coherent Fault Tree (FT) defines a Top Event (TE), representing an undesired state, and the combinations through



(a) FT-SPN of the FT

(b) FT-SPN of the dual FT, after applying reduction rules

Figure 7. The FT-SPNs of the pressure tank system example (FT depicted in Figure 6).

AND and OR logic gates of undesired events that can lead a system to the TE. The Minimal Cut Set (MCS) of an FT defines the minimal set of events that should any of them occur will cause the TE to occur. A Minimal Path Set (MPS) is the dual of a MCS. The MCS and MPS are used in safety and reliability engineering to qualitatively and quantitatively evaluate an FT.

In this paper, we formally define a coherent FT and propose its transformation into a subclass of Petri nets, namely, structurally persistent nets. We prove some interesting properties of the obtained net (named FT-SPN), such as boundedness and conservativeness. We also prove that the computation of minimal p-semiflows in these nets is equivalent to the computation of MCS and MPS in the original FT. Although the computation of the minimal p-semiflows is generally a computationally expensive task, the FT-SPN structure ensures that this extreme case is never reached. We propose an algorithm that uses linear programming techniques to compute the MCS/MPS in an FT-SPN, after applying some reduction rules. Therefore, the computation of the MCS/MPS of a coherent FT is solvable in polynomial time. We put our algorithm into practice by evaluating the FT of a pressure tank system.

At the moment, we are implementing our approach as a module in the PeabraiN [62] tool, taking fault trees described

Table II
MINIMAL P-SEMIFLOWS OF THE DUAL FT-SPN WITHOUT APPLYING REDUCTION RULES, AND THE ASSOCIATED MINIMAL CUT SETS.

| **Place** $p$ | **Minimal p-semiflow** | **MCS** |
|---|---|---|
| $p\vert_{E_1}$ | $\mathbf{y}_1 = \{p\vert_{TopEvent}, p\vert_{E_1}\}$ | $\{E_1\}$ |
| $p\vert_{E_2}$ | $\mathbf{y}_2 = \{p\vert_{TopEvent}, p\vert_{E_2}\}$ | $\{E_2\}$ |
| $p\vert_{E_3}$ | $\mathbf{y}_3 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{E_3}\}$ | $\{E_3\}$ |
| $p\vert_{E_4}$ | $\mathbf{y}_4 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{E_4}\}$ | $\{E_4\}$ |
| $p\vert_{E_5}$ | $\mathbf{y}_5 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{E_5}\}$ | $\{E_5\}$ |
| $p\vert_{E_6}$ | $\mathbf{y}_6 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_6}, p\vert_{E_9}\}$ | $\{E_6, E_9\}$ |
| $p\vert_{E_7}$ | $\mathbf{y}_7 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_7}, p\vert_{G_6}, p\vert_{E_9}\}$ | $\{E_7, E_9\}$ |
| $p\vert_{E_8}$ | $\mathbf{y}_8 = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_8}, p\vert_{G_6}, p\vert_{E_9}\}$ | $\{E_8, E_9\}$ |
| $p\vert_{E_9}$ | $\mathbf{y}_9 = \mathbf{y}_6$ | $\{E_6, E_9\}$ |
| $p\vert_{E_{10}}$ | $\mathbf{y}_{10} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_6}, p\vert_{E_{10}}\}$ | $\{E_6, E_{10}\}$ |
| $p\vert_{E_{11}}$ | $\mathbf{y}_{11} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_6}, p\vert_{E_{11}}\}$ | $\{E_6, E_{11}\}$ |
| $p\vert_{E_{12}}$ | $\mathbf{y}_{12} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_7}, p\vert_{E_{12}}\}$ | $\{E_6, E_{12}\}$ |
| $p\vert_{E_{13}}$ | $\mathbf{y}_{13} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_7}, p\vert_{E_{13}}\}$ | $\{E_6, E_{13}\}$ |
| $p\vert_{E_{14}}$ | $\mathbf{y}_{14} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_7}, p\vert_{E_{14}}\}$ | $\{E_6, E_{14}\}$ |
| $p\vert_{E_{14}}$ | $\mathbf{y}_{15} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_7}, p\vert_{E_{15}}\}$ | $\{E_6, E_{15}\}$ |
| $p\vert_{E_{16}}$ | $\mathbf{y}_{16} = \{p\vert_{TopEvent}, p\vert_{G_1}, p\vert_{G_2}, p\vert_{G_3}, p\vert_{G_4}, p\vert_{G_5}, p\vert_{E_6}, p\vert_{G_7}, p\vert_{E_{16}}\}$ | $\{E_6, E_{16}\}$ |

by Open-PSA XML format as input. In the future, we aim to better characterize coherent FT whose MCS/MPS are solvable in polynomial time. We also intend to compare our approach with other techniques for computing the cut sets of a coherent FT.

## ACKNOWLEDGEMENTS

This work was partially supported by the Spanish National Institute of Cybersecurity (INCIBE) according to rule 19 of the Digital Confidence Plan (Digital Agency of Spain), the University of León under contract X43, the ARTEMIS Joint Undertaking nSafeCer under grant agreement no. 295373 and by National funding. Part of this study was performed while Dr. Rodríguez was a visiting researcher at Mälardalen University, Västerås (Sweden). The author is grateful to Javier Campos for his useful comments on a draft of this paper and to anonymous referees for providing constructive comments and helping to improve the contents of this manuscript.

## REFERENCES

[1] G. Hura and J. Atwood, "The Use of Petri Nets to Analyze Coherent Fault Trees," *IEEE Trans. Rel.*, vol. 37, no. 5, pp. 469–474, 1988.

[2] W. E. Vesely, F. Goldberg, N. Roberts, and D. Haasl, *Fault Tree Handbook*. United States Government Printing Office, 1981.

[3] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback, "Fault Tree Handbook with Aerospace Applications," National Aeronautics and Space Administration, Washington, DC, Handbook, 2002.

[4] D. Kececioglu, *Reliability Engineering Handbook*, 1st ed. Prentice Hall, 1991, vol. 2.

[5] W. S. Lee, D. Grosh, F. Tillman, and C. Lie, "Fault Tree Analysis, Methods, and Applications – A Review," *IEEE Trans. Rel.*, vol. R-34, no. 3, pp. 194–203, 1985.

[6] A. Rauzy, "New algorithms for fault trees analysis," *Reliab. Eng. Syst. Safe.*, vol. 40, no. 3, pp. 203–211, 1993.

[7] R. Sinnamon and J. Andrews, "New approaches to evaluating fault trees," *Reliab. Eng. Syst. Safe.*, vol. 58, no. 2, pp. 89–96, 1997.

[8] Y. Dutuit and A. Rauzy, "A Linear-Time Algorithm to Find Modules of Fault Trees," *IEEE Trans. Rel.*, vol. 45, no. 3, pp. 422–425, 1996.

[9] R. Sinnamon and J. Andrews, "Fault Tree Analysis and Binary Decision Diagrams," in *Proceedings of the Annual Reliability and Maintainability Symposium*, 1996, pp. 215–222.

[10] Y. Ren and J. Dugan, "Design of Reliable Systems Using Static and Dynamic Fault Trees," *IEEE Trans. Rel.*, vol. 47, no. 3, pp. 234–244, 1998.

[11] A. Rauzy, "Mathematical Foundations of Minimal Cutsets," *IEEE Trans. Rel.*, vol. 50, no. 4, pp. 389–396, 2001.

[12] W. S. Jung, S. H. Han, and J. Ha, "A fast BDD algorithm for large coherent fault trees analysis ," *Reliab. Eng. Syst. Safe.*, vol. 83, no. 3, pp. 369–374, 2004.

[13] Y. Mo, L. Xing, and J. Dugan, "MDD-Based Method for Efficient Analysis on Phased-Mission Systems With Multimode Failures," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 6, pp. 757–769, June 2014.

[14] D. Codetta-Raiteri and L. Portinale, "Dynamic Bayesian Networks for Fault Detection, Identification, and Recovery in Autonomous Spacecraft," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 13–24, Jan 2015.

[15] J. Carrasco and V. Sune, "An Algorithm to Find Minimal Cuts of Coherent Fault-trees with Event-Classes, Using a Decision Tree," *IEEE Trans. Rel.*, vol. 48, no. 1, pp. 31–41, March 1999.

[16] J. Fussell and W. Vesely, "A New Methodology for Obtaining Cut Sets for Fault Trees," *T. Am. Nucl. Soc.*, vol. 15, pp. 262—-263, 1972.

[17] A. Rauzy, "Toward an Efficient Implementation of the MOCUS Algorithm," *IEEE Trans. Rel.*, vol. 52, no. 2, pp. 175–180, 2003.

[18] L. Rosenberg, "Algorithm for finding minimal cut sets in a fault tree," *Reliab. Eng. Syst. Safe.*, vol. 53, no. 1, pp. 67–71, 1996.

[19] S. Garribba, P. Mussio, F. Naldi, G. Reina, and G. Volta, "Efficient Construction of Minimal Cut Sets from Fault Trees," *IEEE Trans. Rel.*, vol. R-26, no. 2, pp. 88–94, 1977.

[20] D. M. Rasmuson and N. H. Marshall, "FATRAM – A Core Efficient Cut-Set Algorithm," *IEEE Trans. Rel.*, vol. R-27, no. 4, pp. 250–253, Oct 1978.

[21] P. K. Pande, M. E. Spector, and P. Chatterjee, "Computerized Fault Tree Analysis: TREEL and MICSUP," Operational Research Center, University of Berkeley, California, Tech. Rep. ORC 75-3, 1975.

[22] D. Makajić-Nikolić, M. Vujošević, and N. Nikolić, "Minimal cut sets of a coherent fault tree generation using reverse Petri nets," *Optimization*, vol. 62, no. 8, pp. 1069–1087, 2013.

[23] J. Esparza and M. Nielsen, "Decidability Issues for Petri Nets - a survey," *Bulletin of the EATCS*, vol. 52, pp. 244–262, 1994.

[24] Y. Chen, Z. Li, and M. Zhou, "Behaviorally optimal and structurally simple liveness-enforcing supervisors of flexible manufacturing systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, pp. 615–629, May 2012.

[25] R. J. Rodríguez, J. Júlvez, and J. Merseguer, "On the Performance Estimation and Resource Optimisation in Process Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 6, pp. 1385–1398, 2013.

[26] Y. Chen, Z. Li, K. Barkaoui, and M. Uzam, "New Petri net structure and its application to optimal supervisory control: Interval inhibitor arcs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 10, pp. 1384–1400, Oct 2014.

[27] O. Baruwa, M. Piera, and A. Guasch, "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored Petri nets and anytime heuristic search," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 5, pp. 831–846, May 2015.

[28] Y. Du, C. Jiang, and M. Zhou, "Modeling and analysis of real-time cooperative systems using Petri nets," *IEEE Transactions on Systems,*

*Man and Cybernetics, Part A: Systems and Humans*, vol. 37, no. 5, pp. 643–654, Sept 2007.

[29] L. Ma and J.-P. Tsai, "Formal modeling and analysis of a secure mobile-agent system," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 1, pp. 180–196, January 2008.

[30] R. Robidoux, H. Xu, L. Xing, and M. Zhou, "Automated modeling of dynamic reliability block diagrams using colored Petri nets," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 40, no. 2, pp. 337–351, March 2010.

[31] N. Leveson and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 3, pp. 386–397, 1987.

[32] J. Magott and P. Skrobanek, "Method of time Petri net analysis for analysis of fault trees with time dependencies," *IEE P-Comput. Dig. T.*, vol. 149, no. 6, pp. 257–271, 2002.

[33] W. Huanqiu, G. Jinzhong, and X. Fengzhang, "Dynamic analysis of coherent fault trees," *Journal of Quality in Maintenance Engineering*, vol. 4, no. 2, pp. 122–130, 1998.

[34] A. Adamyan and D. He, "Failure and Safety Assessment of Systems using Petri Nets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2002, pp. 1919–1924.

[35] M. Malhotra and K. Trivedi, "Dependability Modeling Using Petri-Nets," *IEEE Trans. Rel.*, vol. 44, no. 3, pp. 428–440, 1995.

[36] B. Kaiser, C. Gramlich, and M. Förster, "State/event fault trees – A safety analysis model for software-controlled systems ," *Reliab. Eng. Syst. Safe.*, vol. 92, no. 11, pp. 1521–1537, 2007.

[37] S. Bernardi, F. Flammini, S. Marrone, N. Mazzocca, J. Merseguer, R. Nardone, and V. Vittorini, "Enabling the usage of UML in the verification of railway systems: The DAM-rail approach," *Reliab. Eng. Syst. Safe.*, vol. 120, no. 0, pp. 112–126, 2013.

[38] Z. Rochdi, B. Driss, and T. Mohamed, "Industrial systems maintenance modelling using Petri nets," *Reliab. Eng. Syst. Safe.*, vol. 65, no. 2, pp. 119–124, 1999.

[39] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, 2nd ed., ser. Monographs in Theoretical Computer Science. Springer, 1997.

[40] O. Nývlt and M. Rausand, "Dependencies in event trees analyzed by Petri nets," *Reliab. Eng. Syst. Safe.*, vol. 104, no. 0, pp. 45–57, 2012.

[41] N. Leveson, S. Cha, and T. Shimeall, "Safety Verification of Ada Programs Using Software Fault Trees," *IEEE Softw.*, vol. 8, no. 4, pp. 48–59, 1991.

[42] J. Bechta Dugan, S. J. Bavuso, and M. Boyd, "Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems," *IEEE Trans. Rel.*, vol. 41, no. 3, pp. 363–377, 1992.

[43] J. Vaurio, "Fault tree analysis of phased mission systems with repairable and non-repairable components," *Reliab. Eng. Syst. Safe.*, vol. 74, no. 2, pp. 169–180, 2001.

[44] C. Ibáñez-Llano, A. Rauzy, E. Meléndez, and F. Nieto, "A reduction approach to improve the quantification of linked fault trees through binary decision diagrams," *Reliab. Eng. Syst. Safe.*, vol. 95, no. 12, pp. 1314–1323, 2010.

[45] S. Bernardi, F. Flammini, S. Marrone, J. Merseguer, C. Papa, and V. Vittorini, "Model-Driven Availability Evaluation of Railway Control Systems," in *Proceedings of the 30th International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, 2011, pp. 15–28.

[46] G. Chiola, C. Anglano, J. Campos, J. Colom, and M. Silva, "Operational Analysis of Timed Petri Nets and Application to the Computation of Performance Bounds," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models (PNPM)*. Toulouse, France: IEEE Computer Society Press, October 1993, pp. 128–137.

[47] S. Bernardi and J. Campos, "A min-max problem for the computation of the cycle time lower bound in interval-based time Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1167–1181, 2013.

[48] T. Murata, "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE*, vol. 77, no. 4, April 1989, pp. 541–580.

[49] J. Campos, G. Chiola, and M. Silva, "Ergodicity and Throughput Bounds of Petri Nets with Unique Consistent Firing Count Vector," *IEEE Trans. Softw. Eng.*, vol. 17, no. 2, pp. 117–125, February 1991.

[50] M. Čepin and B. Mavko, "A dynamic fault tree," *Reliab. Eng. Syst. Safe.*, vol. 75, no. 1, pp. 83–91, 2002.

[51] T. Liu and S. Chiou, "The application of Petri nets to failure analysis," *Reliab. Eng. Syst. Safe.*, vol. 57, no. 2, pp. 129–142, 1997.

[52] M. Silva and J. Colom, "On the computation of structural synchronic invariants in P/T nets," in *Advances in Petri Nets 1988*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin Heidelberg, 1988, vol. 340, pp. 386–417.

[53] S. Soliman, "Invariants and Other Structural Properties of Biochemical Models as a Constraint Satisfaction Problem," *Algorithm. Mol. Biol.*, vol. 7, no. 1, p. 15, 2012.

[54] J. Colom and M. Silva, "Convex geometry and semiflows in P/T nets. A comparative study of algorithms for computation of minimal p-semiflows," in *Advances in Petri Nets 1990*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1991, vol. 483, pp. 79–112.

[55] J. Fourier, "Solution d'une question particulière du calcul des inégalités," *Nouveau Bulletin des sciences par la Société philomathique de Paris*, vol. 99, pp. 317–319, 1826.

[56] G. Ciardo, G. Mecham, E. Paviot-Adet, and M. Wan, "P-Semiflow Computation with Decision Diagrams," in *Applications and Theory of Petri Nets*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5606, pp. 143–162.

[57] J. Martínez and M. Silva, "A simple and Fast Algorithm to Obtain all Invariants of a Generalised Petri Net," in *Application and Theory of Petri Nets*, ser. Informatik-Fachberichte, C. Girault and W. Reisig, Eds. Springer Berlin Heidelberg, 1982, vol. 52, pp. 301–310.

[58] N. Megiddo, "On the complexity of linear programming," in *Advances in Economic Theory: Fifth world congress*. Cambridge University Press, 1987, pp. 225–268.

[59] V. Klee and G. Minty, "How good is the simplex algorithm?" in *Inequalities, III, Proceedings of the Third Symposium on Inequalities*. Academic Press, 1972, pp. 159–175.

[60] R. G. Bland, D. Goldfarb, and M. J. Todd, "The Ellipsoid Method: A Survey," *Operations Research*, vol. 29, no. 6, pp. 1039–1091, 1981.

[61] B. Yamnitsky and L. A. Levin, "An Old Linear Programming Algorithm Runs in Polynomial Time," in *23rd Annual Symposium on Foundations of Computer Science, 1982*, Nov 1982, pp. 327–328.

[62] R. J. Rodríguez, J. Júlvez, and J. Merseguer, "PeabraiN: A PIPE Extension for Performance Estimation and Resource Optimisation," in *Proceedings of the 12th International Conference on Application of Concurrency to System Designs (ACSD)*. IEEE, 2012, pp. 142–147.

**Ricardo J. Rodríguez** (M'13) received M.S. and Ph.D. degrees in computer science from the University of Zaragoza, Spain, in 2010 and 2013, respectively. His Ph.D. dissertation was focused on performance analysis and resource optimization in critical systems, with special interest in Petri net modeling techniques. He was a Visiting Researcher at the School of Computer Science and Informatics, Cardiff University, Cardiff, U.K., in 2011 (July to September) and 2012 (May to July), and at the School of Innovation, Design and Engineering, Mälardalen University (Västeras, Sweden), in 2014 (mid January to mid March).

He is currently a senior/postdoctoral researcher at the Research Institute of Applied Sciences in Cybersecurity (RIASC), University of León, Spain. His research interests include performance analysis and optimization of large and distributed systems, computer software security, and dependability. He has been involved in reviewing tasks for international conferences and journals. Dr. Rodríguez is a member of the IEEE Computer Society.